Robert Peiternigl, BSc

Modellierung und optimale Steuerung eines Slot-Cars

Masterarbeit

Technische Universität Graz

Institut für Regelungs- und Automatisierungstechnik
Institutsvorstand: Univ.-Prof. Dipl.-Ing. Dr.techn. Martin Horn

Betreuer: Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. tit.Univ.-Prof. Anton Hofer

Wien, September 2014

This document is set in Palatino, compiled with pdfLaTeX2e and `Biber`.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, _____ _____

Date Signature

Eidesstattliche Erklärung[1]

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am _____ _____

Datum Unterschrift

[1] Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Abstract

Although the fastest possible lap on a Carrera track with the corresponding manual controller can be great fun, a time-optimal lap is for an inexperienced user mostly far away. The purpose of this thesis is to provide a time optimal control with the given control technology. In this thesis the Carrera-track, as well as the vehicles are descriped by mathematical models and a time optimal control is designed, as far as possible within the given restrictions. The modelling of the track is done in a modular system of separate sections by B-Splines. The vehicles are modelled as a two-mass-system based on the linear one-track-model. A system of appropriate differential equations is derived using the famous Lagrange formalisms. Based on the mathematical models a suitable solution for time optimal control with respect to the restrictions is found in acceptable calculation time and with restricted memory by the differential evolution-algorithm. A comparison with manual control shows that the result of optimal control works clearly better and thus fulfills its purpose.

Erzielen möglichst kleiner Rundenzeiten auf einer Carrera-Bahn mit dem zugehörigen, manuellen Controller kann zwar viel Spaß machen, von einer zeitoptimalen Runde ist ein normaler Benutzer aber meist weit entfernt. Mit regelungstechnischen Mitteln wird es möglich gemacht, eine zeitoptimale Steuerung zur Verfügung zu stellen. In dieser Arbeit werden die Carrera-Bahn, sowie die Fahrzeuge als Modell abgebildet und dafür eine zeitoptimale Steuerung entworfen, soweit es mit den vorhandenen Beschränkungen möglich ist. Die Modellierung der Strecke erfolgt in einem Baukastensystem der einzelnen Abschnitte per B-Splines. Die Fahrzeuge sind als Zwei-Massen-Systeme an ein lineares Einspurmodell angelehnt und mit den Lagrange-Gleichungen 2. Art als Differentialgleichungen abgebildet. Mit den gefundenen Differentialgleichungen wurde aufgrund der Komplexität der Problemstellung mit der Methode des Differential Evolution-Algorithmus in annehmbarer Berechnungszeit und mit begrenztem Speicherbedarf eine suboptimale Steuerung erstellt, in der die nötigen Beschränkungen berücksichtigt werden. Ein Vergleich mit der manuellen Steuerung per Controller zeigt, dass die rechnerisch ermittelte Steuerung eindeutig besser funktioniert und somit ihren Zweck erfüllt.

Inhaltsverzeichnis

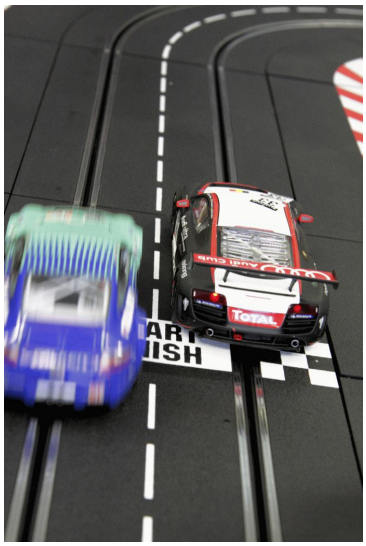
Abbildungsverzeichnis

1 Aufgabenstellung

Mit den Bestandteilen einer Carrera Digital 124 Bahn, bestehend aus Bahnteilen, Handregler, Blackbox und Fahrzeugen, soll eine zeitoptimale Steuerung für unterschiedliche Slot-Cars der Carrera-Bahn entworfen werden. Dabei ist auf einige Einschränkungen gewisser Systemgrößen zu achten, wie beispielsweise auf den Driftwinkel, da der Leitkiel des Slot-Cars nur bis zu einem bestimmten Winkel gedreht werden kann und außerdem bei zu schneller Fahrt das Fahrzeug aus der Bahn fällt, wenn dieser Winkel zu groß ist. Die Herleitung der Systemgleichungen mit den vorhandenen Zwangsbedingungen basiert auf dem Formalismus der Lagrange'sche Gleichungen 2.Art und wurde mit einem Computer-Algebra Programm durchgeführt. Die Umsetzung der Strecke, sowie die Steuerungsberechnung soll in Matlab/Simulink erfolgen. Als Endergebnis soll eine Funktion zur Verfügung gestellt werden, bei der mit Angabe der Streckenteile eine optimale Steuerung, sowie das Streckenprofil und die Werte für die Zustandsgrößen zurückgegeben werden.

Die Aufgabenstellung sieht folgende Themengebiete vor:

- Literaturrecherche zum Themengebiet ”optimale Steuerung”
- Erstellen eines Baukastensystems zur Definition der Strecke
- Modellieren der Fahrzeuge (in Folge auch Slot-Cars genannt)
- Entwurf und Vergleich einer einfachen Steuerung und einer optimalen Steuerung
- Visualisierung und Demonstration
- Implementierung und Analyse in Matlab/Simulink



2 Systembeschreibung

2.1 Streckengeometrie

Die Grundlage des Baukastensystems für die mathematische Beschreibung der Strecke bildet ein Werkzeug der Computergrafik, die sogenannten „B-Splines“. Mit ihnen soll es möglich sein, dass beliebige Strecken mit den vorhandenen Baukastenteilen erstellt werden können und die für das Modell benötigten Ableitungen der Strecke im gleichen Arbeitsschritt mitberechnet werden können.
Die Carrerra Digital 124 Bahn besteht, wie auch andere Carrera-Bahnen, grundsätzlich aus Geradenstücken und Kurvenstücken. Hierbei gibt es Geradenstücke verschiedener Längen und Kurvenstücke unterschiedlicher Radien. Die einzelnen Streckenabschnitte bestehen dabei jeweils aus einer äußeren Schiene und einer inneren Schiene. Somit werden bei einem Streckenaufbau immer zwei Kurse erstellt.
In dieser Masterarbeit soll der Aufbau der Strecke durch Streckensegmente im Baukastenprinzip entstehen, die anschließend flexibel zu einer Gesamtstrecke zusammengesetzt werden können. Es gibt prinzipiell drei zu unterscheidende Arten von Streckensegmenten:

1. Geraden mit verschiedenen Längen
2. Kurven mit verschiedenen Radien
3. Spurwechselstücke

Alle drei Arten sollen durch Splines beschrieben werden, da dies einige Vorteile bietet. Einerseits können bei einem Spline *n*-ter Ordnung *n-1* Ableitungen einfach berechnet werden, ohne dafür einen allzu großen rechentechnischen Aufwand betreiben zu müssen, andererseits ist die Funktion in jedem Punkt stetig, was natürlich eine Voraussetzung bei der Verwendung der Ableitungen sein muss. Für diese Problemstellung wurden Splines *4*. Ordnung gewählt, da in diesem Fall die zweite Ableitung ,die der Bahnkrümmung entspricht, aus stückweisen Geradenstücken besteht.
Um B-Splines, die für den Bau der Streckenabschnitte verwendet werden sollen, beschreiben zu können, muss zuerst auf einige Punkte der Theorie der Splines eingegangen werden. Einen wesentlichen Punkt stellt dabei die polynomielle Interpolation dar. Anhand einer stückweisen linearen Interpolation wird die Thematik grundsätzlich gezeigt. Als nächster Schritt wird mit dem Wissen der linearen Interpolation die kubische Interpolation verwendet, die komplexer ist, aber bessere Ergebnisse liefert. Die Grundgedanken und Gleichungen sind hierbei im Wesentlichen dieselben, der Unterschied ist vor allem darin auszumachen, dass bei linearer Interpolation Geradenstücke verwendet werden, bei kubischer Interpolation hingegen Polynome 4-ter Ordnung.

2.2 Bahnelektrik und Elektronik

Die Spannungsversorgung, wie auch die Befehle an die Fahrzeuge werden über eine Box, in der sich die Elektronik befindet, sicher gestellt, die in dem sogenannten Anschlussstück untergebracht ist. Die Nennspannung beträgt dabei 18V. Die Pulsweitenelektronik ist dabei auf den Autos untergebracht und es wird über den Handregler und die Box nur ein Binär-Wert vorgegeben, der vom Auto in eine entsprechende Spannung eines pulsweitenmodulierten Signals umgesetzt wird, indem bei niedrigen Werten die

Dauer des „High“-Abschnittes in einem Intervall klein wird, bei großen Werten hingegen groß wird. Diese Vorgabe geschieht für jedes Fahrzeug über einen mitgelieferten Controller, wie er in Abbildung 2.1 zu sehen ist.

Abbildung 2.1: Controller der Carrera Digital 124 Bahn

In dieser Arbeit soll dies über ein Simulink-Modell von Matlab aus vorgegeben werden können und somit dafür ausgelegt sein, dass über einen Rechner eine Steuerung erfolgen kann. Die Elektronik des Fahrzeugs wandelt das pulsweitenmodulierte Spannungssignal um und steuert entsprechend den Elektromotor der Fahrzeuge an. Auf der Hinterachse ist ein Getriebe mit nur einem Gang angebracht, wie in Abbildung 2.2 zu erkennen ist, das für den untersetzten Antrieb an der Hinterachse sorgt. Ein Achsdifferential ist bei diesen Fahrzeugen nicht vorhanden.

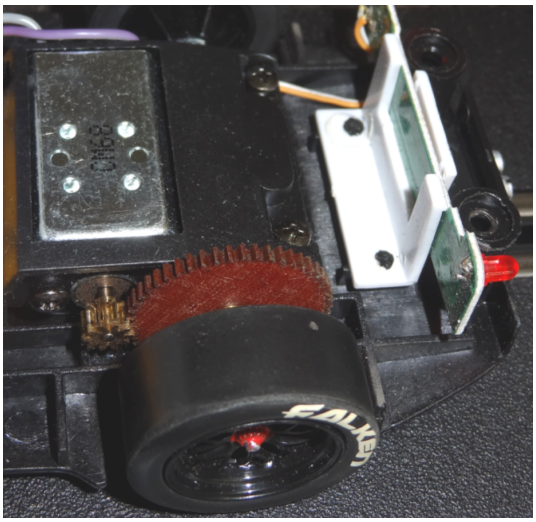


Abbildung 2.2: Antrieb und Untersetzung des Fahrzeugs

Zu erwähnen ist noch, dass das Carrera Digital 124 System das Nachfolgesystem der Carrera Evolution ist. Der Unterschied liegt darin, dass das Digital 124 Set digital aufgebaut ist, das Evolution Set hingegen analog arbeitet, wobei es für neuere Varianten Nachrüstsets für eine Umstellung auf ein digitales System gibt. Mit der Umstellung auf digitale Betriebsart sind einige Neuerungen hinzugekommen. Es ist nun zum Beispiel möglich das Bremsverhalten zu ändern, sowie die maximale Höchstgeschwindigkeit einzustellen, oder auch einen Tankinhalt zu simulieren.

2.3 Fahrzeug

Die Carrera-Fahrzeuge werden im Modell als ein Pendel mit zwei Massen modelliert, wobei die Massen in der Mitte der Vorderachse, bzw. der Hinterachse platziert werden. Dies soll die Zusammenfassung der Räder einer

Achse zu einem Rad in der Mitte der Achse darstellen. Durch diese Vereinfachungen kann das Modell als lineares Einspurmodell gesehen werden. Die Lenkung der Fahrzeuge ist nur aus kosmetischen Gründen installiert, kann aber nicht dazu verwendet werden, die Vorderachse zu lenken. Damit die Autos in der Spur bleiben, ist bei jedem Fahrzeug ein Leitkiel an der Unterseite ein wenig vor der Vorderachse in der Mitte angebracht, wie in Abbildung 2.3 zu sehen ist. Mit diesem Kiel halten die Autos die Spur, solange in einer Kurve der maximale Auslenkwinkel des Kiels nicht überschritten wird. Ist dies der Fall, fällt das Fahrzeug aus der Führungsschiene heraus. Der maximale Auslenkwinkel beträgt etwa 65° in beide Richtungen von der Nullposition aus, die in Fahrzeuglängsrichtung zeigt.

Abbildung 2.3: Leitkiel des Slot-Cars

Für die gute Bodenhaftung sind zwei Magnete verantwortlich, die an der Hinterachse in der Nähe des Motors, sowie am vorderen Teil direkt hinter der Vorderachse sitzen. Durch diese ist der Grenzbereich, bevor das Fahrzeug durch einen zu großen Driftwinkel aus der Bahn fällt, sehr gering und schwer zu kontrollieren. Für ein verändertes Fahrverhalten können

die Magnete auch entfernt werden, was die maximal mögliche Kurvengeschwindigkeit stark reduziert, aber im Gegenzug den Grenzbereich merklich größer werden lässt und die Fahrzeuge fahrbarer macht.

3 B-Splines und Splines

3.1 Theorie

Die Theorie der B-Splines ist sehr umfassend und ihre Beschreibung ist für den Umfang dieser Masterarbeit zu aufwendig. Aus diesem Grund werden nur die wichtigsten Aspekte erklärt, damit die prinzipielle Vorgehensweise deutlich wird. Die anderen Abschnitte der Theorie können in [4, Kapitel 1-11], oder auch in [1, Kapitel 1-9] nachgelesen werden. Zu erwähnen sei an dieser Stelle noch, dass auch die Umsetzung der Matlab-Toolbox für Splines auf der Darstellung von deBoor [4] beruht und damit implementiert wurde und einige Beispiele aus dem Buch auch in der Toolbox enthalten sind.

3.1.1 Polynome

Der große Vorteil von Polynomen ist es, dass diese in endlich vielen Schritten ausgewertet, differenziert und integriert werden können. Die dafür benötigten arithmetischen Operationen beschränken sich dabei auf Additionen, Subtraktionen und Multiplikationen.
Ein Polynom n-ter Ordnung

$$p(x) = a_1 + a_2\,x + \ \ldots\ + a_{n-1}\,x^{n-2} + a_n\,x^{n-1} = \sum_{j=1}^{n} a_j\,x^{j-1}, \qquad \mathrm{n} \geq 0 \quad (3.1)$$

ist ein Polynom mit Grad $n-1$ und besitzt genau n Koeffizienten, die Freiheitsgrade genannt werden. Die Differentiation

$$p'(x) = a_2 + 2\,a_3\,x + \ \ldots\ + (n-2)\,a_{n-1}x^{n-3} + (n-1)\,a_n\,x^{n-2} = \sum_{j=2}^{n}(j-1)\,a_j\,x^{j-2} \tag{3.2}$$

und die Integration

$$\int p(x)\,dx = a_1\,x + \frac{1}{2}a_2\,x^2 + \ \ldots\ + \frac{1}{n+1}\,a_n\,x^n = \sum_{j=1}^{n}\frac{a_j}{j+1}\,x^j \tag{3.3}$$

können wie bereits erwähnt einfach durchgeführt werden.

3.1.2 Newton-Form

Für eine willkürlich gewählte Funktion g kann eine eindeutige Lösung in Form eines Interpolanten p gefunden werden, welcher der ursprünglichen Funktion g an den Stellen $\tau_1 \ldots \tau_n$ gleicht. Dafür wird hier die Newton-Form verwendet:

$$p_n(x) = \sum_{i=1}^{n}(x-\tau_1)\ \ldots\ (x-\tau_{i-1})[\tau_1,\ \ldots\ ,\tau_i]g\,, \tag{3.4}$$

da diese einen guten Kompromiss aus einfacher Erstellung und einfacher Berechnung darstellt. Hierbei stellt der Ausdruck $[\tau_1,\ \ldots\ ,\tau_i]g$ eine symmetrische Funktion der Argumente $\tau_1,\ \ldots\ ,\tau_i$ dar. Die benötigten Koeffizienten $[\tau_1]\,g, [\tau_1,\tau_2]\,g,\ \ldots\ , [\tau_1,\ \ldots\ ,\tau_n]\,g$ der Newton-Form können in einfacher Weise in einer divided-differences-Tabelle berechnet werden, wie sie in Tabelle 3.1 dargestellt ist.

Tabelle 3.1: Tabelle der divided differences, stückweise Geraden

Interpol. -stelle	Wert	erste div. diff.	zweite div. diff.	...	(n-2)te div.diff	(n-1)te div.diff
τ_1	$g(\tau_1)$					
		$[\tau_1, \tau_2]g$				
τ_2	$g(\tau_2)$		$[\tau_1, \tau_2, \tau_3]g$			
		$[\tau_2, \tau_3]g$				
τ_3	$g(\tau_3)$		$[\tau_2, \tau_3, \tau_4]g$		$[\tau_1, ..., \tau_{n-1}]g$	
		$[\tau_3, \tau_4]g$				$[\tau_1, ..., \tau_n]g$
τ_4	$g(\tau_4)$		.		$[\tau_2, ..., \tau_n]g$	
.	.	.				
.	.					
.	.					
τ_{n-1}	$g(\tau_{n-1})$		$[\tau_{n-2}, \tau_{n-1}, \tau_n]g$			
		$[\tau_{n-1}, \tau_n]g$				
τ_n	$g(\tau_n)$					

Wird nun $[\tau_i, \ \dots, \tau_{i+r}]\, g$ berechnet, beträgt im Fall von $\tau_i = \tau_{i+r}$ der Wert $[\tau_i, \ \dots, \tau_{i+r}]\, g = g^{(r)}(\tau_i)/r!$, oder mit $\tau_i \neq \tau_{i+r}$

$$[\tau_i, \ \dots, \tau_{i+r}]\, g = \frac{[\tau_{i+1}, \ \dots, \tau_{i+r}]\, g - [\tau_i, \ \dots, \tau_{i+r-1}]\, g}{\tau_{i+r} - \tau_i}, \tag{3.5}$$

welcher dadurch mit den beiden Einträgen der linken Spalte berechnet werden kann. Somit können die Einträge der Tabelle der divided-differences Spalte für Spalte berechnet werden. Die oberste Diagonale enthält am Ende die Einträge der Newton-Form, die in Gleichung 3.4 angegeben ist.

Beispiel: Interpolation einer Logarithmus-Funktion

Zur Veranschaulichung soll nun der Wert $g(1.5)$ für die Logarithmus-Funktion $g(x) = ln(x)$ berechnet werden. Die Übereinstimmung zwischen der Funktion $ln(x)$ und dem Polynom $p_4(x)$ soll an den Stellen $1, 1, 2, 2$ stattfinden. Da nun alle notwendigen Informationen vorhanden sind, kann nach Gleichung 3.5 die Berechnung der Koeffizienten durchgeführt werden. Das Ergebnis ist in Tabelle 3.2 dargestellt.

Tabelle 3.2: Tabelle der divided differences für die Interpolation des Logarithmus

τ	$g(\tau)$	1. div. diff	2. div. diff	3. div. diff
1	0			
		1		
1	0		-0.306853	
		0.693147		0.113706
2	0.693147		-0.193147	
		0.5		
2	0.693147			

Mit den abgelesenen Werten der ersten Diagonale kann nun das Polynom $p_4(x)$ angegeben werden:

$$p_4(x) = 0 + (x-1)\,1 + (x-1)^2(-0.306853) + (x-1)^2(x-2)(0.113706)$$

Wird der Wert $x = 1.5$ eingesetzt und damit die Elemente berechnet, ergibt sich, dass $p_4(1.5) = 0.409074 \approx 0.405465 = ln(1.5)$.

Die Schreibweise eines Polynoms kann zur Berechnung der Ableitungen von p aber noch weiter verändert werden:

$$p(x) = \sum_{i=1}^{n} (x - \tau_1) \ldots (x - \tau_{i-1}) a_i \tag{3.6}$$

Dies kann auch in verschachtelter Form als

$$p(x) = a_1 + (x - \tau_1)a_2 + (x - \tau_2)[a_3 + \ ... \ + (x - \tau_{n-2})(a_{n-1} + (x - \tau_{n-1}a_n) \ ...] \tag{3.7}$$

geschrieben werden, woraus der Algorithmus für verschachtelte Multiplikationen abgeleitet werden kann. Mit gegebenen Werten für die Stellen $\tau_1, \ ... \ , \tau_{n-1}$ und den zugehörigen Koeffizienten $a_1, \ ... \ , a_n$ der Newton-Form, sowie einer weiteren Stelle τ_0 kann der Algorithmus ausgeführt werden:

$b_n = a_n$
für $k = n-1, n-2, \ ... \ , 1$ mache folgendes:
$\quad b_k = a_k + (\tau_0 - \tau_k)\, b_{k+1}$

Mit diesem Berechnungsschema ergibt sich am Ende $b_1 = p(\tau_0) = \sum_{i=1}^{n}(\tau_0 - \tau_1) \ ... \ (\tau_0 - \tau_{i-1})a_i$. Es tragen alle b_k einen Teil der Informationen über p. Mit diesem Ansatz ergibt sich:

$$\begin{aligned} p(x) = b_1 + (x - \tau_0)\, b_2 + (x - \tau_0)(x - \tau_1)\, b_3 + \ ... \\ +(x - \tau_0)(x - \tau_1) \ ... \ (x - \tau_{n-2})b_n \end{aligned} \tag{3.8}$$

Der Algorithmus berechnet somit aus der Newton-Form für p an den Stellen $\tau_1, \ ... \ , \tau_{n-1}$ die Newton-Form für p an $\tau_0, \ ... \ , \tau_{n-2}$. Der Aufbau ist in Tabelle 3.3 dargestellt.

Tabelle 3.3: Tabelle der divided differences mit verschachtelten Multiplikationen

Interpol.-stelle	Wert	erste div. diff.	zweite div. diff.	...	(n-2)te div.diff	(n-1)te div.diff
τ_{-1}	c_1					
		c_2				
τ_0	b_1		c_3			
		b_2				
τ_1	a_1		b_3		c_{n-1}	
		a_2				c_n
τ_2	.		a_3		b_{n-1}	
		.				b_n
τ_3	.		.		a_{n-1}	
		.				a_n
.	.		.		.	

Der Algorithmus kann für die Stellen $\tau_0, \ldots, \tau_{n-2}$, die Koeffizienten $b_1, \ldots, b_n$ und einer neuen Stelle τ_{-1} wiederholt werden. Daraus ergeben sich die Koeffizienten der neuen obersten Diagonale mit

$$p(x) = c_1 + (x - \tau_{-1})c_2 + \ \ldots \ + (x - \tau_{-1}) \ldots (x - \tau_{n-3})c_n.$$

Der Algorithmus ist sehr nützlich zur Berechnung der Ableitungen von p an einer Stelle, da $p^{(j)}(\tau)/j!$ als der $j+1$te Koeffizient in jeder Newton-Form von p ist, wo die $j+1$ ersten Stellen alle gleich τ sind.

Beispiel: Berechnung der Ableitungen eines Polynoms in Newton-Form

Für die Berechnung der Ableitung wird derselbe Ansatz wie im bereits beschriebenen Beispiel verwendet. Es werden drei weitere Diagonalen zu der

divided-differences-Tabelle 3.2 für p_4, jeweils für den Punkt *1.5*, hinzugefügt. Die letzte berechnete Diagonale der Tabelle 3.4, die die oberste der Tabelle ist, enthält die Ableitungen von p_4 an der Stelle *1.5*, die sogenannten Taylor-Koeffizienten.

Tabelle 3.4: Tabelle der divided differences für die Interpolation des Logarithmus, Ableitungen

τ	$g(\tau)$	1.div.diff	2.div.diff	3.div.diff
1.5	0.4090735			
		0.6647205		
1.5	0.4090735		-0.25	
		0.6647205		0.113706
1.5	0.4090735		-0.306853	
		0.818147		0.113706
1	0		-0.363706	
		1		0.113706
1	0		-0.306853	
		0.693147		0.113706
2	0.693147		-0.193147	
		0.5		
2	0.693147			

Aus der Diagonale abgelesen ergibt sich:

$$p_4(x) = 0.409735 + (x - 1.5)\,0.6647205 - (x - 1.5)^2\,0.25 + (x - 1.5)^3\,0.113706$$

3.1.3 Approximation einer Funktion

Bei einer Approximation können Teilstücke mit niedriger Ordnung genauso verwendet werden, wie Teilstücke mit höherer Ordnung, wobei mit höherer

Ordnung die erzielbaren Ergebnisse einen flüssigeren Verlauf aufweisen und auch effektiver sind.
Zu jedem Intervall $[\tau_i\,,\,\tau_{i+1}]$ gibt es ein f, das mit den Polynomen P_i 4-ter Ordnung übereinstimmt:

$$f(x) = P_i(x) \quad \text{für } \tau_i \leq x \leq \tau_{i+1} \quad i = 1,\ \ldots\ , n-1 \tag{3.9}$$

Das i-te Stück des Polynoms P_i muss folgenden Ansprüchen genügen:

$$\begin{aligned} &P_i(\tau_i) = g(\tau_i), \quad && P_i(\tau_{i+1}) = g(\tau_{i+1}) \\ &P_i'(\tau_i) = s_i, && P_i'(\tau_{i+1}) = s_{i+1} \end{aligned} \qquad i = 1,\ \ldots\ , n-1 \tag{3.10}$$

Hierbei sind $s_i,\ \ldots\ , s_n$ freie Parameter. Die resultierende stückweise kubische Funktion f stimmt mit g an den Stellen $\tau_1,\ \ldots\ , \tau_n$ überein und ist stetig und besitzt eine stetige erste Ableitung, unabhängig davon wie die freien Parameter $(s_i)_1^n$ gesetzt werden. Zur Berechnung der Koeffizienten der Teilstücke kann wiederum die Newton-Form verwendet werden. Die frei wählbaren Parameter $s_2,\ \ldots\ , s_{n-1}$ werden mit der Bedingung berechnet, dass f zweifach stetig differenzierbar ist. Die Funktion f besitzt aus diesem Grund neben einer ersten, auch eine zweite stetige Ableitung. Daraus entsteht die Bedingung, dass mit $i = 2,\ \ldots\ , n-1$

$$P_{i-1}''(\tau_i) = P_i''(\tau_i)$$

zu einem tridiagonalen, linearen System mit n-2 Gleichungen für n-2 Unbekannte führt. Daraus ist erkennbar, dass es nur genau eine Lösung für das System geben kann.

3.1.4 Representation der stückweisen Polynomfunktionen mit B-Splines

Obwohl die Basis-Splines, auch als B-Splines bezeichnet, komplexer sind als andere, mögliche Methoden, hat ihre Verwendung einige Vorteile gegenüber

den meisten anderen Vorgehensweisen, die von großer Bedeutung sind. So kann jede Spline-Funktion der Ordnung k als eine Linearkombination von B-Splines mit derselben Ordnung geschrieben werden.
Per Definition sei $t = (t_j)$ eine nicht kleiner werdende Sequenz, die als Knotenvektor oder Stützstellen bezeichnet werden. Mit der Definition von Curry-Schönberg ist für $[t_j\,,\, t_{j+k}]$ die Funktion $B_{j,k,t}(x)$ stückweise polynomiell der Ordnung k und nur in einem kleinen, lokalen Bereich ist der Wert ungleich Null:

$$B_{j,k,t}(x) = 0 \quad \text{für } x \notin [t_j\,,\, t_{j+k}] \tag{3.11}$$

Die B-Splines B_{jk} sind normalisiert, sodass

$$\sum_{j=r}^{s} B_{jk} = 1 \quad mit\, [t^{+}_{r+k-1}\,,\, t^{-}_{s+1}] \tag{3.12}$$

ist, wobei mit den Indizes $+$ und $-$ bei den Knotenwerten t_j die jeweils rechts- und linksseitigen Werte einer rechtsseitig stetigen Funktion bezeichnet werden. Zur einfacheren Lesbarkeit wird in der Regel auf den Index t verzichtet und anstatt $B_{j,k,t}$ nur B_{jk} geschrieben.
Eine spezielle Knotensequenz stellt hierbei die äquidistante Knotensequenz $t = \mathbb{Z} = (..., -1, 0, 1, ...)$ dar. In diesem Fall sind die zugehörigen B-Splines Integer-Transformationen voneinander.
Per Definition von B_{j1} ist die charakteristische Funktion des j-ten Knotenintervalls:

$$B_{j1}(x) = \begin{cases} 1, & \text{für } t_j \leq x < t_{j+1} \\ 0, & \text{sonst.} \end{cases} \tag{3.13}$$

Die Funktionen sind rechtsseitig stetig. Diese Annahme kann auch anders getroffen werden, die einzige Bedingung ist, dass $\sum_j B_{j1} = 1$, wobei mit $t_j = t_{j+1}$ daraus $B_{j1} = 0$ folgt. Beginnend mit diesen B-Splines *1*-ter Ordnung können nun B-Splines höherer Ordnung wie folgt definiert werden:
Für $k > 1$:

$$B_{jk} = \omega_{jk} B_{j,k-1} + (1 - \omega_{j+1,k} B_{j+1,k-1}) \tag{3.14}$$

mit

$$\omega_{jk}(x) = \frac{x - t_j}{t_{j+k-1} - t_j}$$

Damit wäre ein B-Spline 2-ter Ordnung durch

$$B_{j2} = \omega_{j2} B_{j,1} + (1 - \omega_{j+1,2}) B_{j+1,1}$$

gegeben. Dieser besteht aus zwei nichttrivialen linearen Teilen, die stetig zueinander sind und außerhalb von $[t_j\,,\, t_{j+1})$ verschwinden.
Wenn nun $t_j = t_{j+1}$ sei, dann besteht B_{j2} aus nur einem Teil, mit einem Sprung bei b_j, ist aber immer noch stetig bei t_{j+1}. Der B-Spline 3-ter Ordnung wäre durch

$$\begin{aligned} B_{j3} =& \omega_{j3} B_{j2} + (1 - \omega_{j+1,3}) B_{j+1,2} \\ =& \omega_{j3}\omega_{j2} B_{j1} + (\omega_{j3}(1 - \omega_{j+1,2}) + (1 - \omega_{j+1,3})\omega_{j+1,2}) B_{j+1,1} \\ &+ (1 - \omega_{j+1,3})(1 - \omega_{j+2,2} B_{j+2,1}) \end{aligned}$$

beschrieben. Dieser besteht aus drei parabolischen Teilstücken, die zusammengefügt eine $C^{(1)}$-Funktion bilden, welche außerhalb der Sequenz $[t_j\,,\, t_{j+2})$ verschwindet.
Nach $k - 1$ Schritten der Rekursion ergibt sich somit B_{jk} in der Form:

$$B_{jk} = \sum_{r=j}^{j+k-1} b_{rk} B_{r1} \tag{3.15}$$

Jeder Term b_{rk} ist hierbei selbst ein Polynom mit Grad $k - 1$, hingegen ist B_{r1} ein konstanter Wert.
Der B-Spline $B_{j,k,t}$ mit der Ordnung k ist somit stückweise polynomiell im Bereich $t_j, \ldots, t_{j+k}$ und außerhalb des Intervalls $[t_j\,,\, t_{j+k})$ gleich null, sowie positiv für

$$B_{j,k,t}(x) > 0, \quad \text{für } t_j < x < t_{j+k} \tag{3.16}$$

während bei

$$t_j = t_{j+k} \quad \Rightarrow \quad B_{jk} = 0 \tag{3.17}$$

ist. Der B-Spline $B_{j,k,t}$ hängt somit nur von den $k+1$ Knoten $t_j, \ldots, t_{j+k}$ ab. Die Glattheit von B_{jk} hängt hingegen von der Anzahl der Stützstellen in der Knotensequenz (t_j, t_{j+k}) ab.
Eine Spline-Funktion der Ordnung k mit der Knoten-Sequenz t ist eine Linearkombination von B-Splines der Ordnung k. Die Menge all dieser Funktionen wird mit $\$_{k,t}$ bezeichnet.

$$\$_{k,t} = \left\{ \sum_i \alpha_i B_{i,k,t} \ : \ \alpha_i \text{ real, für alle } i \right\}, \tag{3.18}$$

wobei $\$_{k,t} \subseteq \Pi_{<k,t}$ ist. $\Pi_{<k,t}$ gibt hierbei den gesamten Raum von stückweisen Polynomen mit der Knotensequenz t an. Das Basis-Intervall für $\$_{k,t}$ ist:

$$I_{k,t} = (t_-, t_+) \text{ mit}$$

$$t_- = \begin{cases} t_k & \text{mit } t = (t_1, \ldots) \\ inf_j\, t_j & \text{sonst} \end{cases} \qquad t_+ = \begin{cases} t_{n+1} & \text{mit } t = (\ldots, t_{n+k}) \\ sup_j\, t_j & \text{sonst} \end{cases} \tag{3.19}$$

Somit ist auch für infinite und biinfinite Sequenzen sichergestellt, dass die Knoten-Sequenz endlich ist und einen ersten und einen letzten Knoten besitzt.

Definition der B-Form

Die B-Spline-Form für $f \in \Pi_{k,\xi,\nu}$ beinhaltet somit:

- Die Integer k und n, die die Ordnung von f, bzw. die Anzahl der linearen Parameter vorgeben.
- Den Knotenvektor $t = (t_i)_1^{n+k}$, der die Knoten in aufsteigender Ordnung beinhaltet.
- Den Vektor $\alpha = (\alpha_i)_1^n$ mit den Koeffizienten von f mit der B-Spline-Basis $(B_i)_1^n$ für $\Pi_{k,\xi,\nu}$ der Knotensequenz t

Der Wert f an einer Stelle x in $[t_k \,..\, t_{n+1}]$ ist daher

$$f(x) = \sum_i 1^n \alpha_i B_i(x) \tag{3.20}$$

oder mit $t_j \leq x \leq t_{j+1}$ für $j \in \{k, \ ... \ , n\}$

$$f(x) = \sum_{i=j-k+1} \alpha_i B_i(x). \tag{3.21}$$

Eine stabile Evaluierung von B-splines und Splines

B-Splines können mit Hilfe der Rekursionsformel einfach berechnet werden, da hierbei nicht auf die Multiplizität der Knoten oder ähnliches geachtet werden muss. Aus dieser Formel kann ein Algorithmus für die Werte von x der k B-Splines, mit der Ordnung k, angeschrieben werden.
Es sei $t_i < t_{i+1}$ und $x \in [t_i, \ ... \ , t_{i+1}]$. Alle Werte der B-Splines, die nicht automatisch null ergeben, können in einer Triangulärmatrix geschrieben werden, wie in Tabelle 3.5 dargestellt.
Die Triangulärmatrix kann Spalte für Spalte ermittelt werden, da die erste Spalte bekannt ist. Aus diesem Grund können alle weiteren Spalten der Reihe nach berechnet werden. Bei der Berechnung des ersten und des letzten Eintrages einer Spalte, der nicht null ist, kann die Tatsache genutzt werden, dass einer der Nachbarn zur linken Seite null ist.
Ein Beispiel: Es sei die j-te Spalte mit $B_{i-j+1,j}(x), \ ... \ , B_{i,j}(x)$ bereits berechnet und in einem Vektor $b = (b_r)_1^j$ gespeichert. Der Vektor $b' = (b'_r)_1^{j+1}$, der die Werte $B_{i-j,j+1}(x), \ ... \ , B_{i,j+1}(x)$ enthalten soll, kann folgendermaßen berechnet werden:

$$\begin{aligned} b'_r = (x - t_{i-j+r-1} \frac{b_{r-1}}{t_{i+r-1} - t_{i-j+r-1}} + (t_{i+r} - x) \frac{b_r}{t_{i+r} - t_{i-j+r}}) \\ \text{für } r = 1, \ ... \ , j+1 \text{ und } b_0 = bj + 1 = 0 \end{aligned} \tag{3.22}$$

Tabelle 3.5: Triangulärmatrix der B-Splines mit Ordnung $\leq k$, die im Intervall $[t_i \,..\, t_{i+1}]$ nicht null ergeben

				o
			o	
		.		$B_{i-k+1,k}$
		o	$B_{i-k+2,k-1}$	
	o	.		$B_{i-k+2,k}$
o		$B_{i-2,3}$	$B_{i-k+3,k-1}$	
	$B_{i-1,2}$		.	$B_{i-k+3,k}$
B_{i1}		$B_{i-1,3}$	.	.
	B_{i2}		.	.
o		B_{i3}	$B_{i-1,k-1}$	
	o	.		$B_{i-1,k}$
		o	$B_{i,k-1}$	
		.		B_{ik}
			o	
				o

mit den Abkürzungen

$$\delta_{sj}^{L} = x - t_{i-j+s-1} \quad \delta_{s}^{R} = t_{i+j} - x \quad s = 1, \ldots, k-1$$

als

$$b'_r = \delta^L_{r-1,j} \frac{b_{r-1}}{\delta^L_{r-1,j} + \delta^R_{r-1}} + \delta^R_r \frac{b_r}{\delta^L_{rj} + \delta_r}^{R} \quad r = 1, \ldots, j+1. \tag{3.23}$$

Differentiation

Die *B-Spline-Eigenschaft der Differentiation* kann über folgende Gleichung beschrieben werden:

$$D(\sum_j \alpha_j B_{jk}) = (k-1) \sum_j \frac{\alpha_j - \alpha_{j-1}}{t_{j+k-1} - t_j} B_{j,k-1} \tag{3.24}$$

Somit kann die erste Ableitung einer Spline-Funktion $\sum_j \alpha_j B_{jk}$ einfach durch differenzieren der B-Spline-Koeffizienten gefunden werden, indem ein Spline eine Ordnung niedriger verwendet wird, als der ursprüngliche Spline. Mit den zugehörigen Grenzen ergibt sich

$$D(\sum_{j=r}^{s} \alpha_j B_{jk}) = \sum_{j=r}^{s+1} (k-1) \frac{\alpha_j - \alpha_{j-1}}{t_{j+k-1} - t_j} B_{j,k-1} \quad \text{wobei } \alpha_{r-1} = 0 = \alpha_{s+1} \tag{3.25}$$

und wiederholt für die m-te Ableitung ergibt sich:

$$D^m \left(\sum_j \alpha_j B_{jk} \right) = \sum_j \alpha_j^{(m+1)} B_{j,k-m} \tag{3.26}$$

mit

$$\alpha_r^{(m+1)} = \begin{cases} \alpha_r & \text{für } m = 0 \\ \frac{\alpha_r^{(m)} - \alpha_{r-1}^{(m)}}{(t_{r+k-m} - t_r)/(k-m)} & \text{für } m > 0 \end{cases} \tag{3.27}$$

Diese Gleichung führt an einer Stelle zu einer Division durch 0, deshalb wird auch hier für den Fall $t_{r+k-m} - t_r = 0$ der Wert $B_{r,k-m} = 0$ gesetzt [4].

3.2 Implementierung in Matlab

Es gibt insgesamt vier verschiedene Arten von Streckensegmenten, die in Matlab umzusetzen sind:

- Geraden, 3 verschiedene Längen
- 30°-Kurven, 4 verschiedene Radien
- 60°-Kurven, 4 verschiedene Radien
- Spurwechsel, auf einem Geradenstück

Um eine Art Baukastensystem für die Streckenabschnitte zu erhalten, wurde eine eigene Klasse *Bspline_curve* erstellt, mit der es möglich ist, vordefinierte Kurven mit verschiedenen Radien zu erstellen und sie in eine Gesamtstrecke zusammenzufügen. In dieser Klasse sind verschiedenste Methoden definiert, mit denen die benötigten Daten der Strecke für das verwendete Modell ausgelesen werden können. Alle vorhandenen Dateien sind auf der beiliegenden CD vorhanden.

3.2.1 Klasse Bspline_curve

Um das Baukastensystem umzusetzen, wurde in Matlab eine eigene Klasse mit dem Namen "Bspline_curve" erstellt, die alle Anforderungen erfüllt. In den folgenden Abschnitten werden die wichtigsten Funktionen erläutert. Eine detaillierte Erklärung der Klasse ist auf der beiliegenden CD zu finden.

Parameter der Klasse

Wie auch bei anderen Programmiersprachen können in Matlab bei der Definition einer Klasse Datenelemente mit unterschiedlichen Zugriffsrechten

festgelegt werden. In dieser Arbeit wurden Objekteigenschaften verwendet, die nur zur internen Berechnung, als auch für die Ausgabe und Weiterverarbeitung benötigt wurden. Die sogenannten internen Parameter besitzen nur einen privaten Zugriff für Lese- als auch Schreibrechte und können so nur von internen Methoden geändert werden. Folgende Parameter sind enthalten:

- Ordnung der B-Splines, Skalar
- Knoten der Spline-Funktion, Vektor eindimensional
- Kontrollpunkte für die Erstellung der Spline-Funktion und ihre Ableitungen, Vektor zwei- oder dreidimensional
- berechnete Koordinatenwerte der Spline-Funktion und ihre Ableitungen, Vektor zwei- oder dreidimensional
- Länge der Strecke, Vektor eindimensional
- Winkel, als auch Krümmung der zugehörigen Stellen, Vektoren eindimensional
- Werte zur Einstellung der Genauigkeit, Skalare

Konstruktor

Im Konstruktor werden, wenn für eine Instanz keine Streckenteile angegeben wurden, nur allgemeine Werte festgelegt. Dies sind die Ordnung der Spline-Funktion, die durch die Art der Verwendung immer auf vier gesetzt wird, damit die 2. Ableitung lineare Abschnitte als Werte der Krümmung liefert, sowie der Winkel, der anfangs auf null initialisiert wird. Die Streckenteile sind als Strings anzugeben, die durch Beistriche getrennt werden. Wenn bei der Erstellung einer Instanz bereits Streckenteile angegeben werden, werden diese im Konstruktor anhand der Funktion *add_track_part* gespeichert.

Funktion add_track_part

Mit den Strings *inner* oder *outer* kann am Beginn der Eingabe gewählt werden, ob die innere oder die äußere Bahn erstellt werden soll. Für die Eingabe einer Geraden sind folgende Strings verfügbar:

- *straight*
- *straight_3*
- *straight_4*

Wie aus den Bezeichnungen ersichtlich, sind Geradenstücke mit drei verschiedenen Längen verfügbar. Die Standardgerade hat eine Länge von 0.345m, die beiden anderen haben ein Drittel, beziehungsweise ein Viertel dieser Länge.
Für die Erstellung der Kurvenabschnitte können die Strings mit den Namen nach folgendem Schema eingegeben werden: *curve_XXX_rX*. Für die erste freigelassene Stelle ist der Winkel des gewünschten Segments einzugeben, wobei Winkel mit 30° oder 60° möglich sind. Die zweite freigelassene Stelle ist für die Eingabe des Radius' des Abschnittes vorgesehen. Es sind Radien mit

- *0.3465m*, bzw. *0.2475m*
- *0.5445m*, bzw. *0.4455m*
- *0.7425m*, bzw. *0.6435m*, sowie mit
- *0.9405m*, bzw. *0.8415m*

möglich. Die Wertepaare sind immer für das Innenstück und das zugehörige Außenstück zu verstehen. Als Beispiel sieht eine Eingabe der gewünschten Kurve folgendermaßen aus: *curve_30p_r1*. Dies wäre ein 30° Kurvenstück mit dem ersten möglichen Radius, in mathematischer Zählweise positiver Richtung.
Neben Geraden und Kurvenstücken gibt es noch die Kreuzung der Spuren. Das Stück ähnelt einer Gerade, jedoch kreuzen sich die zwei Spuren und

es wechselt somit die Innenspur mit der Außenspur. Dies ist aus Sicht der Spline-Funktion ein wichtiger Punkt. Bei dem Wechsel der Spuren treten durch die relativ engen Radien hinsichtlich der Ableitungen der Funktion vergleichsweise große Werte auf, weshalb diese Stelle bei der Optimierung ein entscheidender Bereich ist, da hier das Fahrzeug stark verlangsamt werden muss, damit es nicht von der Strecke fällt.

Methode compute_spline

Diese Methode ist sozusagen das Kernstück der Klasse. Mit den vorhandenen Daten der Ordnung, dem Knotenvektor und einer monoton steigenden Folge τ, wird mit der Funktion *spcol* der Spline-Toolbox eine Triangulärmatrix erstellt, wie es in Tabelle 3.5 dargestellt ist. Eine Multiplikation dieser Matrix mit den Kontrollpunkten ergibt die Koordinatenwerte der Spline-Funktion. Die Triangulärmatrix beinhaltet die Gewichtungen der Kontrollpunkte, das heißt, dass nur diejenigen Punkte ungleich null sind, die für den jeweiligen Koordinatenpunkt von Bedeutung sind. Dies zeigt die Eigenschaft der Lokalität der Spline-Funktion.

Methode compute_derivative

Nach der Berechnung der Koordinatenpunkte werden mit dieser Methode die Ableitungen berechnet. Die Ordnung wird um je einen Wert pro Ableitung verringert und mit der in Gleichung 3.24 erläuterten Formel werden die neuen Kontrollpunkte berechnet. Die Berechnung der Koordinatenpunkte erfolgt danach auf die gleiche Weise wie bei der Berechnung in 3.2.1.

Methode plot

Die plot-Funktion von Matlab wurde überladen, um damit die Kontrollpunkte, sowie die Koordinatenpunkte darstellen zu können.

Hilfsmethode get_points

In dieser Methode werden Standardstücke für die Geraden und die unterschiedlichen Kurven erstellt und danach so bearbeitet, dass diese an die geeignete Größe und Position angepasst werden. Die Kontrollpunkte für den jeweiligen Streckenabschnitt werden bereitgestellt, die durch eine Anpassung der Spline-Abschnitte an die Kreisstücke erstellt wurden, wie in Kapitel 3.2.2 beschrieben. Der prinzipielle Ablauf ist hier bei den Geraden und den Kurven derselbe. Zuerst werden die Kontrollpunkte mit einem Faktor skaliert, damit jeder Abschnitt an die gewünschte Länge angepasst wird. Bei den Geraden wird hierbei die Länge skaliert, bei den Kurvenabschnitten hingegen der Radius. Zu beachten ist außerdem, dass der Radius je nach Innen- oder Außenstück einen unterschiedlichen Wert hat. Anschließend werden die Punkte so gedreht, dass der erste Punkt an den vorigen Abschnitt angepasst wird. Durch die unzulängliche Genauigkeit der 2. Ableitung wurde in dieser Funktion zusätzlich die Tatsache ausgenützt, dass mit der Angabe der Abschnitte bekannt ist, um welchen Abschnitt es sich jeweils handelt. Mit dieser Tatsache werden in einer Tabelle die Länge, sowie der Radius von jedem Sektor gespeichert, wobei für eine Gerade der Wert für den Radius mit null angegeben wird. Für die Werte der Krümmung muss somit nur mit der jeweiligen Länge eine Funktion aufgerufen werden, die den Wert berechnet. Nach einem gesamten Durchlauf der angegebenen Abschnitte werden alle Kontrollpunkte bereitgestellt und die Berechnung der Spline-Punkte durchgeführt.

3.2.2 Anpassung der Spline-Abschnitte an Kreisstücke

Damit die Kontrollpunkte für die Anpassung des Splines an ein Kreisstück möglichst gut gesetzt werden können, wurde in Matlab eine Optimierung der Spline-Punkte durchgeführt. Als Referenz wurden jeweils Kreisbögen mit 30°, sowie mit 60° verwendet. Für die Optimierung werden der Start- und der Endpunkt mit einem fest vorgegebenen Wert festgelegt. Alle weiteren Kontrollpunkte werden anhand einer Optimierung mit der Matlab-Funktion "*fminsearch*" bearbeitet. Es wird für jeden Kontrollpunkt ein Startwert vorgegeben. Die Kontrollpunkte werden nun solange in alle Richtungen verschoben, bis die geometrische Abweichung der resultierenden Punkte der Spline-Kurve zu den Punkten des Kreisstücks die kleinste Summe aufweist. Die Anzahl der veränderbaren Kontrollpunkte wurde auf eine geringe Zahl eingeschränkt, da in vielen Versuchen zu erkennen war, dass bei den benötigten Abschnitten mit $30°$ und $60°$ die besten Ergebnisse in einem Bereich zwischen 2 und 4 Punkten gefunden wurden. Die Abweichung wurde mit der Methode der kleinsten Fehlerquadrate berechnet und optimiert. In Abbildung 3.1, und einem Vergrößerungsausschnitt der Anpassung in Abbildung 3.2, ist die geringe Abweichung des Spline-Abschnittes zum richtigen Kreisbogen erkennbar.

3.2.3 Beispiele

Der folgende Abschnitt veranschaulicht einige Ergebnisse der Berechnungen einer Spline-Funktion. Die benötigten Werte werden nach Eingabe der Streckenabschnitte, siehe Kapitel 3.2.1, und der Berechnung der Spline-Funktion, sowie ihren Ableitungen, ausgegeben. Einige Vorlagen für Strecken einer Carrera Bahn sind auf der Seite von Carrera-toys [3] verfügbar.

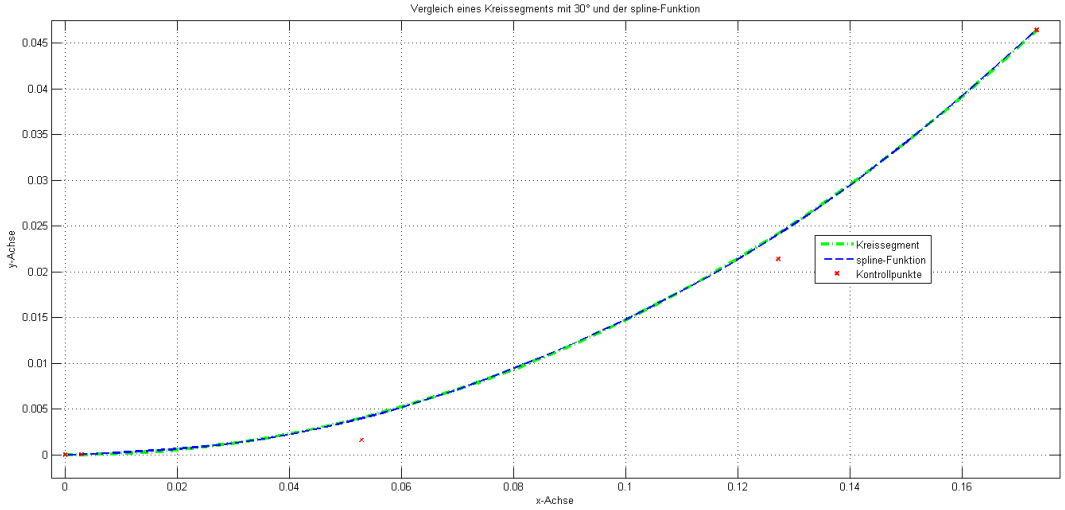


Abbildung 3.1: Kurvensegment mit 30° und Spline-Abschnitt im Vergleich

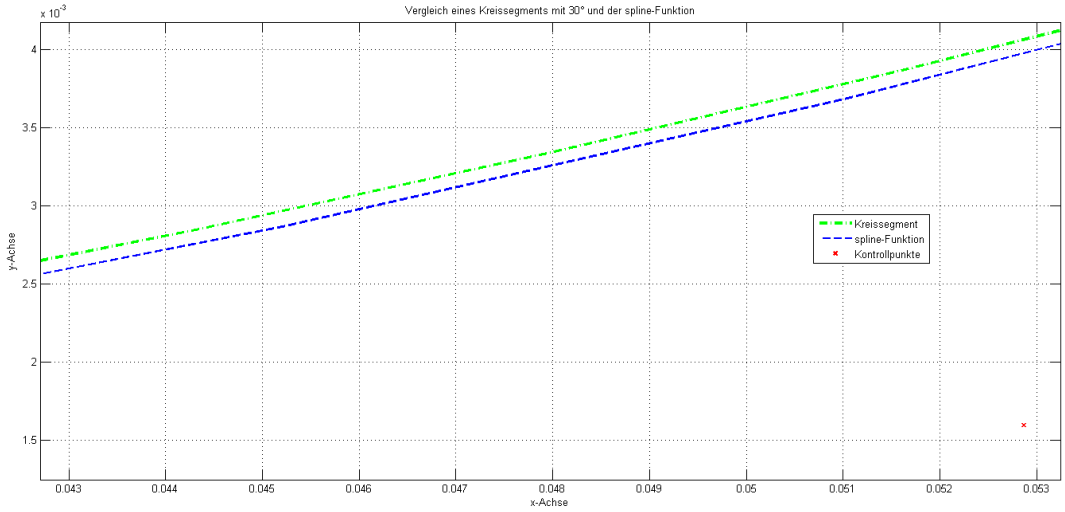


Abbildung 3.2: Vergrößerungsausschnitt der Anpassung

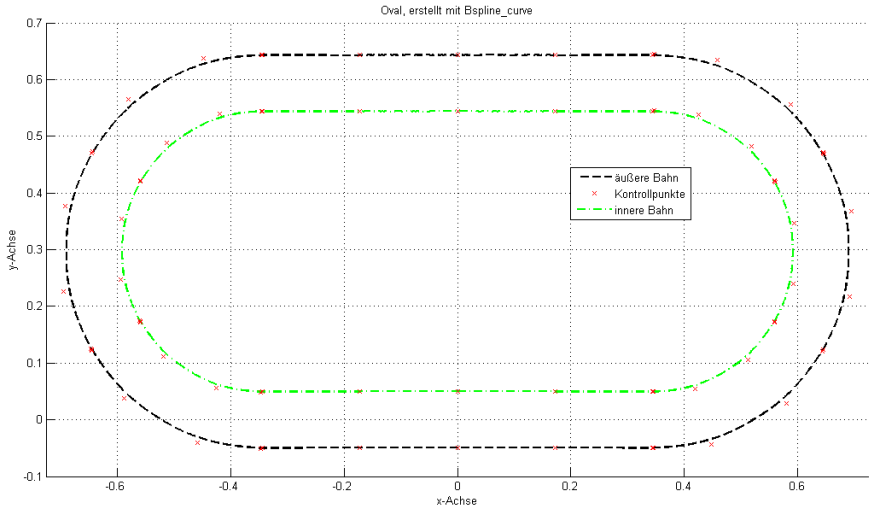


Abbildung 3.3: Ovale Strecke mit Innen- und Außenspur

Einfache Strecke: Oval

Als einfaches Beispiel zur Veranschaulichung ist in Abbildung 3.3 ein Oval dargestellt, wie es auch später für die Steuerungsoptimierung erzeugt wird. Es besteht aus vier Geradenstücken und sechs Kurventeilen mit einem positiven Radius von 60° pro Stück. Wie nun aus der Theorie der Splines bekannt ist, wird die Ableitung einer Spline-Funktion 4. Ordnung durch eine Spline-Funktion 3. Ordnung beschrieben. In Abbildung 3.4 ist gut zu erkennen, dass die 1. Ableitung eines Ovals einen Kreis beschreibt. Die Ableitungen der Geraden sind, da die Steigung konstant bleibt, Punkte auf denselben Koordinaten. Somit bleiben noch die Ableitungen der Kurvenstücke übrig, die wiederum einen Kreis ergeben. An den Übergängen weichen die Werte etwas von der idealen Kennlinie ab, dies kann mit einer Glättung der Daten behoben werden, sodass es nicht mehr erkennbar ist.
Bei den Werten für den Winkel ϕ, wie in Abbildung 3.5 ersichtlich, und die

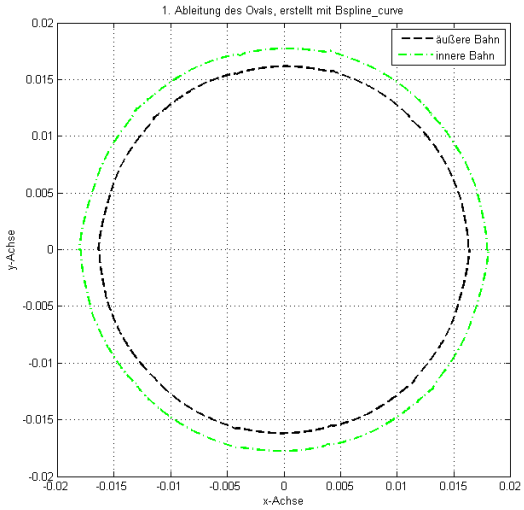


Abbildung 3.4: 1. Ableitung der ovalen Strecke

Krümmung ρ, in Abbildung 3.6 ersichtlich, ist ebenfalls erkennbar, dass die Abweichungen minimal sind. Nur bei jeweils einem Drittel und zwei Drittel der Kurve, genau in den Übergängen der vorgegebenen Streckenabschnitte, weist der Winkel kleine Spitzen auf. Auf den Geraden ist der Winkel, sowie die Krümmung jeweils 0, bei den Kurven steigt der Winkel gleichmäßig an, da diese jeweils einen Halbkreis beschreiben. Die Krümmung in den Kurven ist daher ein konstanter Wert, da die Ableitung einer gleichmäßig ansteigenden Geraden immer denselben Wert besitzt.

Komplexe Strecke: Suzuka

Die notwendigen Daten können für einfache Strecken mit ein wenig Übung relativ schnell per Hand erzeugt werden. Bei komplexeren Strecken ist dies jedoch nicht mehr möglich. Hier sind die Stärken des Baukastensystems mit

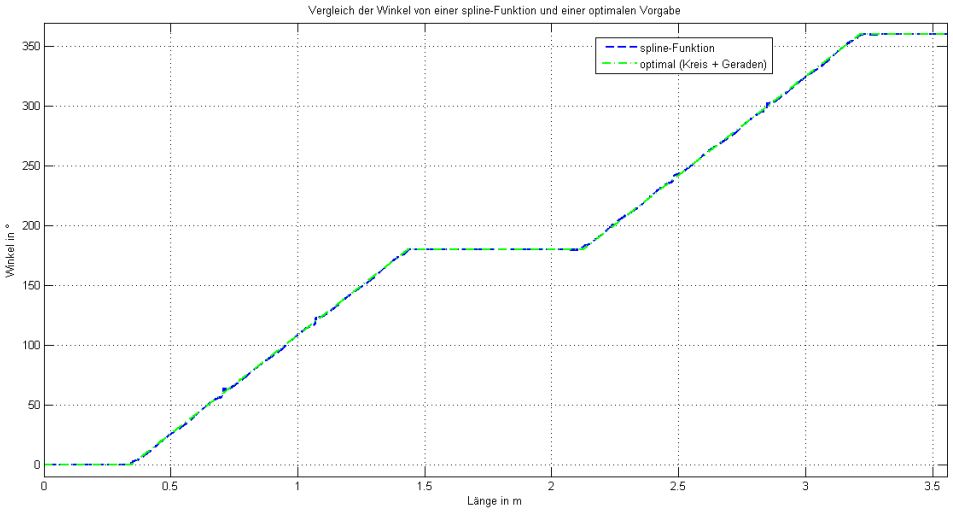


Abbildung 3.5: Winkel der Strecke für Spline-Funktion und optimaler Vorgabe

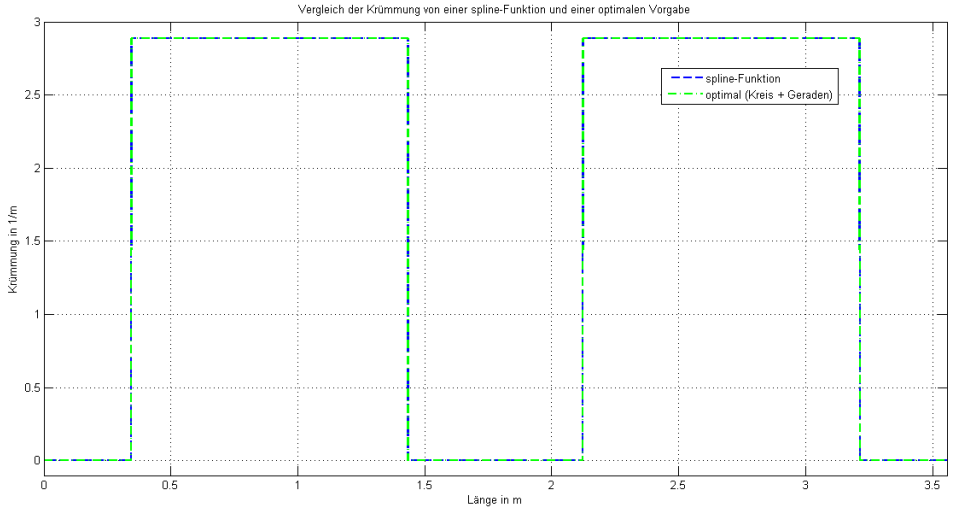


Abbildung 3.6: Krümmung der Strecke für Spline-Funktion und optimaler Vorgabe

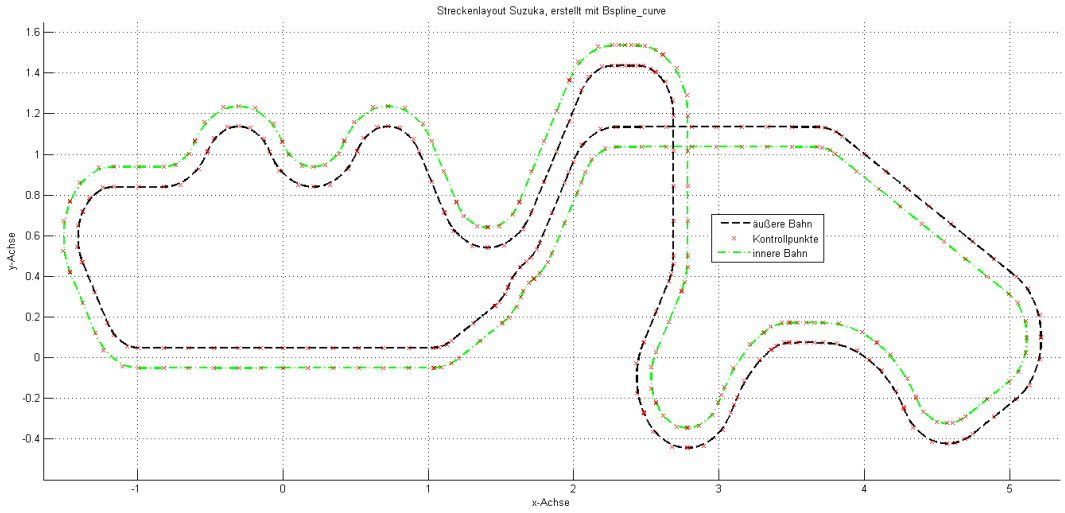


Abbildung 3.7: Strecke Suzuka, Ausführung mit Innen- und Außenspur

der Spline-Funktion deutlich erkennbar. Die Spline-Funktion und ihre Ableitungen können schnell ermittelt werden, da nur die Abfolge der Bauteile einzugeben ist. Bei einem hohen Genauigkeitsgrad kann die Berechnung viel länger dauern, daher gilt es abzuwägen, wie das Verhältnis Rechenaufwand zu Rechendauer ist. In den Abbildungen 3.7 und 3.8 sind die vier möglichen Strecken dargestellt. Für jede dieser Strecken kann eine eigene Steuerungsoptimierung ausgeführt werden. Die Strecke der Innenspur in Abbildung 3.7 hat eine Länge von 18.21m, die der Außenspur hat dieselbe Länge, da die zwei Spuren auf einem Teil angebracht sind und bei diesen der Anfang und das Ende bei einer Strecke zusammenführen. In Abbildung 3.8 sind die Längen unterschiedlich, weil die Kreuzung der Bahnen zu einem Zeitpunkt erfolgt, wo die hier als Innenspur, mit einer Länge von 18.27m, bezeichnete Strecke einen viel größeren Anteil der Runde auf der äußeren Seite der Teile verbringt als die Außenspur, die nur 18.17m lang ist.

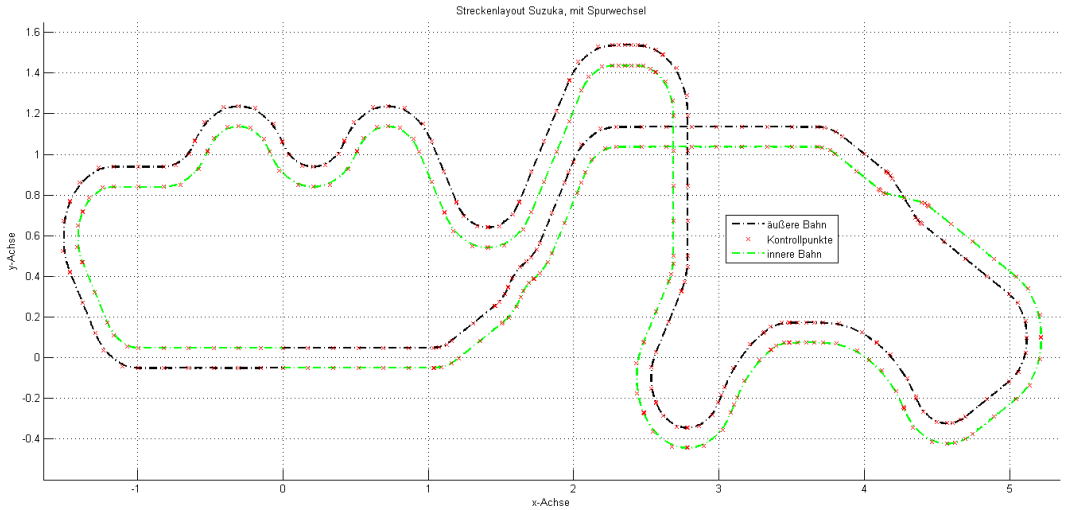


Abbildung 3.8: Strecke Suzuka, Ausführung mit Spurwechsel

4 Fahrzeugmodell

4.1 Lineares Einspurmodell

Die Entwicklung eines geeigneten Modells ist der maßgebliche Teil dieser Arbeit. Auf Basis dieses Modells werden anschließend die Optimierungen durchgeführt, die aber nur ein befriedigendes Ergebnis liefern können, wenn das Modell möglichst gut das Verhalten der Fahrzeuge abbilden kann.
Als Ausgangspunkt für das letztendlich verwendete Modell wurde das lineare Einspurmodell verwendet, wie es in [9, Kapitel 18] beschrieben ist. Dabei wird ein theoretisches Fahrzeugmodell mit zwei vereinfachten Annahmen vorausgesetzt:

- Der Schwerpunkt des Fahrzeugs liegt in Fahrbahnhöhe. Damit verändert beispielsweise die im Schwerpunkt angreifende Zentrifugalkraft nicht die Radlasten, wobei die zusätzliche Belastung der kurvenäußeren und die Entlastung der kurveninneren Räder aus Gründen der Einfachheit des Modells vernachlässigt wird. Weiters soll kein Wanken auftreten, das bedeutet, dass Bewegungen, die als Rotation um die y-Achse auftreten würden, nicht berücksichtigt werden.
- Es liegt ein lineares System vor, welches vor allem durch die Annahme einer konstant bleibenden Geschwindigkeit, wie auch durch Vereinfachungen wie einer proportionalen Reifenseitenkraft zum Schräglaufwinkel,

oder einer proportionalen seitlichen Luftkraft zum Anströmwinkel angenommen werden kann.

Mit diesen Vereinfachungen wird das Fahrverhalten eines Fahrzeugs in Normalsituationen beschrieben. Es wird auch in der Praxis bei modernen Fahrzeugregelungen angewendet. Als Beispiel kann die Fahrdynamikregelung, auch als *ESP* bezeichnet, genannt werden. Hierbei werden mit Sensoren der Lenkradeinschlagwinkel δ_L und die Gierwinkelgeschwindigkeit $\dot{\psi}_{Mess}$ gemessen und zeitgleich mit einem Rechner der theoretische Wert $\dot{\psi}_{Theorie}$ bestimmt. Bei einer Differenz $\dot{\psi}_{Theorie}$ - $\dot{\psi}_{Mess}$ wird das Fahrzeug durch ein Moment um die Hochachse, welches durch einseitiges Bremsen erzeugt wird, in die gewünschte Lage gedreht [9, Kapitel18].
In den Abschnitten 4.1.1 und 4.1.2 wird jeweils auf die kinematischen Größen, Kräfte und Bewegungsgleichungen des linearen Einspurmodells näher eingegangen. Die weiteren Größen und Eigenschaften des linearen Einspurmodells, wie in [9, Kapitel 18] beschrieben, sind für das in weiterer Folge verwendete Modell nicht von Bedeutung.

4.1.1 Kinematische Größen

Abbildung 4.1 zeigt das ebene Modell eines zweiachsigen, vierrädrigen Kraftfahrzeuges. Die Räder an der Vorderachse werden zu einem Vorderrad zusammengefasst und äquivalent dazu werden die Hinterräder zu einem Hinterrad in der Achsmitte zusammen gefasst. Das Fahrzeug schrumpft zum Einspurmodell. Die Geschwindigkeit v ist Tangential zur Bahnkurve gerichtet, ebenso die Tangentialbeschleunigung $\dot{v}$, die Zentripedalbeschleunigung v^2/ρ zeigt zum Krümmungsmittelpunkt M der Bahnkurve. Der Abstand $\overline{M\ SP}$ ist der Krümmungsradius ρ der Bahnkurve. Der Winkel zwischen v und der Fahrzeugmittellinie ist der Schwimmwinkel β. Der

Kurswinkel setzt sich aus dem Gierwinkel ψ und dem dem Schwimmwinkel β zusammen und wird als Kurswinkel bezeichnet.

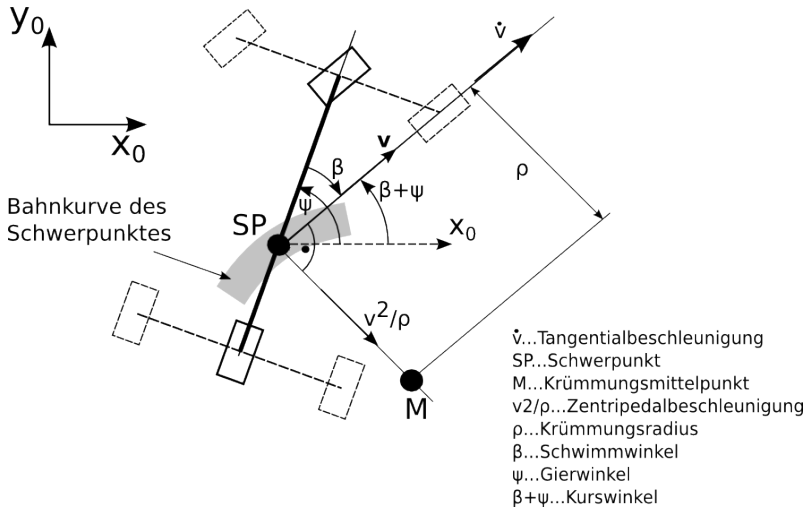


Abbildung 4.1: Kinematische Größen an einem Einspurmodell

4.1.2 Kräfte

In Abbildung 4.2 sind die Kräfte am Fahrzeug dargestellt. F_{xV} und F_{xH} sind die Umfangskräfte an Vorder- und Hinterrad, die in Richtung der Räder zeigen. F_{yV} und F_{yH} sind die Seitenkräfte, die senkrecht auf die Räder wirken. Die Abstände der Achsen zum Schwerpunkt SP sind mit l_V und l_H dargestellt, sowie die Länge zwischen den Achsen mit l. δ_V bezeichnet den Lenkradeinschlag an der Vorderachse. Im Druckmittelpunkt DP, dessen Abstand zu SP mit e_{SP} bezeichnet wird, greift die seitliche Luftkraft F_{Ly} an und der Luftwiderstand ist F_{Lx}.

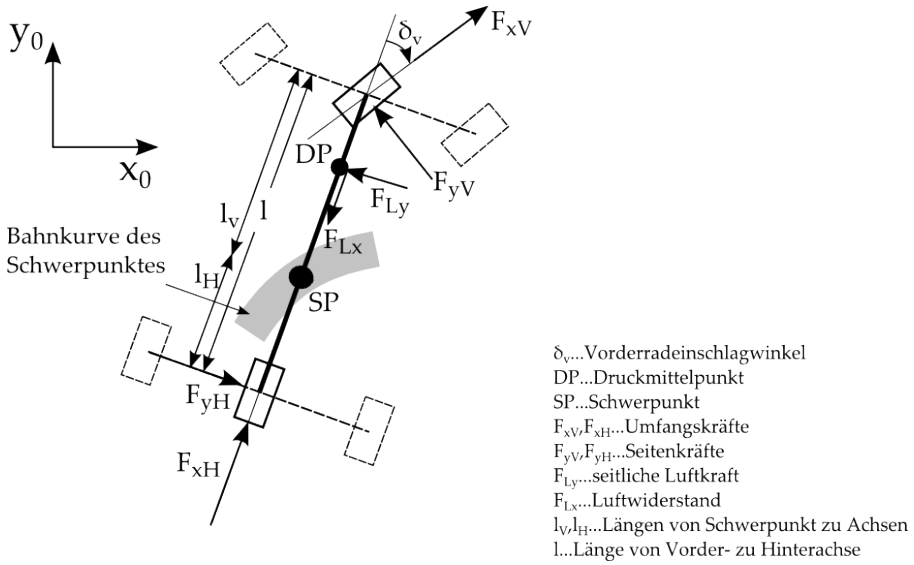


Abbildung 4.2: Kräfte an einem Einspurmodell

4.2 Verwendetes Modell

Das verwendete Modell ist mit einigen Überlegungen des linearen Einspurmodells erstellt worden. Manche der Überlegungen werden jedoch nicht verwendet, da bei einem Carrera-Auto viel kleinere Größenordnungen der Kinematischen Größen, sowie auch der Kräfte, auftreten. Die Lenkung ist bei diesen Autos nicht vorhanden, da das Fahrzeug von der Bahn über den Kiel geleitet wird. Somit fällt der Parameter δ_V, der nur mit der Lenkung zu tun hat, weg. Durch die geringe Größe der Autos und der recht niedrigen Geschwindigkeit im Vergleich zu echten Fahrzeugen wurde auch der Einfluss des Fahrtwindes vernachlässigt. Dieser wirkt sich erst bei höheren Geschwindigkeiten stärker aus.

Mit dem Wissen des linearen Einspurmodells wird als Ansatz ein Modell gewählt, das dem eines geführten Pendels mit zwei Massen gleicht. Zur Berechnung werden die Lagrange-Gleichungen 2.Art verwendet, worauf Kapitel 4.3 genau eingeht.

4.2.1 Kinematische Größen

Es ist zu erkennen, dass sich das Modell doch in einigen Punkten vom linearen Einspurmodell unterscheidet. Das Modell wird anhand von zwei Massenpunkten aufgebaut. Dabei steht der vordere Massenpunkt für die Vorderachse, wobei die zwei Vorderräder in der Mitte der Vorderachse zu einem Vorderrad zusammengefügt sind, wie es auch im linearen Einspurmodell verwendet wird. Der zweite Massenpunkt stellt die Hinterachse dar, welches aus den beiden Hinterrädern zusammengefasst wird.
Durch die Führungsschiene des Fahrzeuges und die nicht-funktionsfähige Lenkung fährt das Auto im Normalfall in Richtung der Führungsschiene. Bei großen Kurvengeschwindigkeiten beginnt das Auto zu driften, wobei durch das Driften die Fahrzeugmittellinie nicht mehr parallel zur Führungsschiene ist. Diese Abweichung wird mit ϕ bezeichnet und ergibt addiert mit dem Winkel ψ die zweite generalisierte Koordinate der Lagrange-Gleichungen. Die erste generalisierte Koordinate ist die Bogenlänge der Bahn.

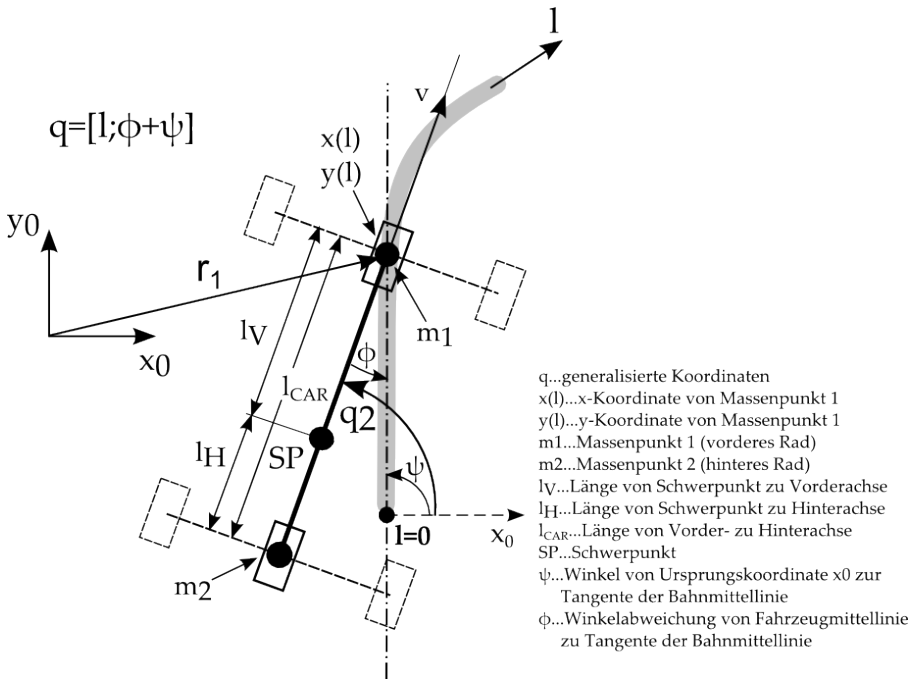


Abbildung 4.3: Kinematische Größen des verwendeten Modells

4.2.2 Kräfte

Die Kräfte des Luftwiderstandes, F_{Ly} und F_{Lx} werden wie bereits erwähnt vernachlässigt, da das Modell-Auto nicht in solche Geschwindigkeitsbereiche kommt, dass diese einen großen Einfluss auf das Fahrverhalten haben. Die theoretische maximale Geschwindigkeit ohne Reibwerte mit der maximalen Drehzahl von 18000 U/min beträgt in etwa $6m/s$, was etwa $21km/h$ entspricht. Für die Dimension der Carrera-Bahn ist dies ein doch recht ordentlicher Wert, wie bei einer eigenen Fahrt festgestellt werden kann.
Durch den Elektromotor an der Hinterachse entsteht die Antriebskraft am Hinterrad. Die nötige Steuerung dieses Antriebs kann anstatt über den

Handregler mit einer eigens dafür entworfenen Platine auch über MatLab-Simulink vorgegeben werden, was in einem späteren Abschnitt für die optimale Steuerung genutzt wird.
Die Größe der Gleitreibungskraft ist nahezu geschwindigkeitsunabhängig und ihre Richtung der Geschwindigkeit entgegengesetzt, mit der der betrachtete Körper über einen anderen gleitet. Die Reibungszahl μ_{Gl} ist in aller Regel kleiner als die Haftreibungszahl μ_0 und hängt von der Oberfläche ab. Durch Magnete, die an der Fahrzeugunterseite verbaut sind, wird das Fahrzeug derart an die Fahrbahn gedrückt, dass der Wert in diesem Fall die Haftreibungszahl jedoch deutlich übersteigt. Die Rollreibungskraft erschwert die Bewegung eines rollenden Rades. Sie ist wiederum nahezu geschwindigkeitsunabhängig und die Rollreibungszahl μ_R ist wesentlich kleiner als die Gleitreibungszahl μ_{Gl} [8, Kapitel 4].

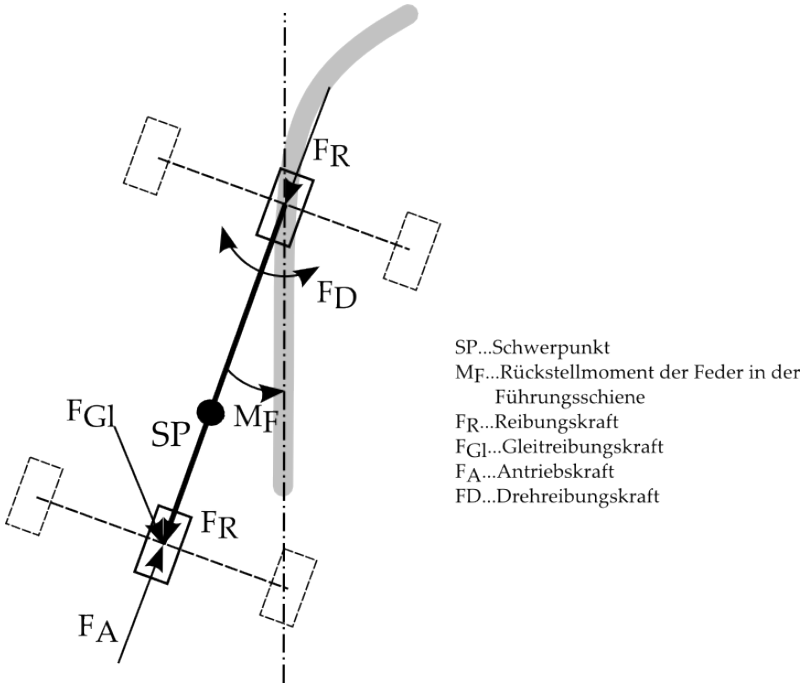


Abbildung 4.4: Kräfte des verwendeten Modells

4.3 Lagrange-Gleichungen 2.Art

Per Definition, wie in [8, Kapitel 3], ist die Lagrangefunktion $L := T - V$ die Differenz der kinetischen und der potentiellen Energie. Sie ist eine Funktion der $3N - k$ unabhängigen, generalisierten Koordinaten q_j, als auch ihrer zeitlichen Ableitungen $\dot{q_j}$, wobei $j = 1 \ldots 3N - k$ ist. In bestimmten Fällen kann es auch vorkommen, dass die Lagrangefunktion von der Zeit abhängt, in den meisten Anwendungen ist dies allerdings nicht der Fall. Für die Beschreibung eines Massenpunkten in der Ebene sind jeweils zwei Freiheitsgrade notwendig, daher sind für N Massenpunkte $2N$ Freiheitsgrade notwendig. Das k steht in diesem Fall für die Anzahl der Zwangsbedingungen des Systems. In Summe ergibt sich damit die notwendige Anzahl von $2N - k$ Freiheitsgrade in der Ebene für die $2N - k$ generalisierten Koordinaten, die bereitgestellt werden müssen, um das System zu beschreiben.
Die Lagrangegleichungen 2. Art gelten für alle Systeme mit holonomen Zwangsbedingungen. Sie sind von der d'Alembertgleichung abgeleitet und sind unter Punkttransformationen $q_i \rightarrow Q_i(q, t)$ forminvariant. Daher besitzen sie auch in krummlinigen Koordinaten und in beschleunigten Bezugssystemen stets die gleiche Gestalt, was in verschiedenen Anwendungen von großem Vorteil sein kann.

4.3.1 Herleitung Lagrangegleichungen 2. Art

Die Basis, von der ausgegangen wird, bildet die d'Alembertgleichung:

$$\sum_{i=1}^{N} (m_i \ddot{\mathbf{r}}_i - \mathbf{F}_i) \cdot \delta \mathbf{r}_i = 0 \qquad mit \qquad i = 1, ..., N \tag{4.1}$$

In dieser Gleichung werden die Massenpunkte mit m_i dargestellt, F_i sind die eingeprägten Kräfte an den Massenpunkten m_i und als r_i werden die

Ortsvektoren der Massenpunkte bezeichnet. Die Elemente $\delta\mathbf{r}_i$ sind die virtuellen Verschiebungen, in denen auch die Zwangskräfte versteckt sind. Die d'Alembertgleichung kann wiederum aus den Bewegungsgleichungen des zweiten newtonschen Gesetzes abgeleitet werden. Im Vergleich zur newtonschen Bewegungsgleichung treten hier keine Zwangskräfte mehr auf, diese sind in den virtuellen Verschiebungen untergebracht. Die Transformationsgleichungen

$$\mathbf{r}_i = \mathbf{r}_i(q_1, ..., q_n, t) \tag{4.2}$$

sollen nun für die Ortsvektoren r_i der N Teilchen gelten. Die n, noch unbestimmten, verallgemeinerten Koordinaten q_j müssen nicht unabhängig voneinander sein. Es gilt:

$$\delta\mathbf{r}_i = \sum_{j=1}^{n} \frac{\partial \mathbf{r}_i}{\partial q_j}\, \delta q_j$$

Die virtuelle Arbeit der eingeprägten Kräfte lässt sich nun schreiben als:

$$\sum_{i=1}^{N} \mathbf{F}_i \cdot \delta\mathbf{r}_i = \sum_{j=1}^{n} \left[\sum_{i=1}^{N} \mathbf{F}_i \cdot \frac{\partial \mathbf{r}_i}{\partial q_j} \right] \delta q_j = \sum_{j=1}^{n} Q_j\, \delta q_j \tag{4.3}$$

Die sogenannten generalisierten Kräfte sind somit:

$$Q_j := \sum_{i=1}^{N} \mathbf{F}_i \cdot \frac{\partial \mathbf{r}_i}{\partial q_j} \tag{4.4}$$

Für die weitere Bearbeitung wird als erstes der erste Term der Gleichung 4.1 untersucht:

$$\begin{aligned} \sum_{i=1}^{N} m_i \ddot{\mathbf{r}}_i \cdot \delta\mathbf{r}_i &= \sum_{j=1}^{n} \left[\sum_{i=1}^{N} m_i \ddot{\mathbf{r}}_i \cdot \frac{\partial \mathbf{r}_i}{\partial q_j} \right] \delta q_j = \\ &= \sum_{j=1}^{n} \left[\sum_{i=1}^{N} \frac{d}{dt} \left(m_i \dot{\mathbf{r}}_i \cdot \frac{\partial \mathbf{r}_i}{\partial q_j} \right) - m_i \dot{\mathbf{r}}_i \cdot \frac{d}{dt} \frac{\partial \mathbf{r}_i}{\partial q_j} \right] \delta q_j \end{aligned}$$

Mit

$$\begin{aligned} \frac{\partial \mathbf{r}}{\partial q_i} &= \frac{\partial \dot{\mathbf{r}}}{\partial \dot{q}_i} \\ \frac{\partial \dot{\mathbf{r}}}{\partial q_i} &= \frac{d}{dt}\frac{\partial \mathbf{r}}{\partial q_i} \end{aligned} \tag{4.5}$$

und der kinetischen Energie

$$T = \frac{1}{2}\sum_{i=1}^{N} m_i \mathbf{v}_i^2 \tag{4.6}$$

ergibt sich

$$\begin{aligned} \sum_{i=1}^{N} m_i \ddot{\mathbf{r}}_i \cdot \delta \mathbf{r}_i &= \sum_{j=1}^{n}\left[\sum_{i=1}^{N}\frac{d}{dt}\left(m_i \mathbf{v}_i \cdot \frac{\partial \mathbf{v}_i}{\partial \dot{q}_j}\right) - m_i \mathbf{v}_i \cdot \frac{\partial \mathbf{v}_i}{\partial q_j}\right]\delta q_j = \\ &= \sum_{j=1}^{n}\left[\frac{d}{dt}\left\{\frac{\partial}{\partial \dot{q}_j}\sum_{i=1}^{N}\frac{1}{2}m_i \mathbf{v}_i^2\right\} - \frac{\partial}{\partial q_j}\sum_{i=1}^{N}\frac{1}{2}m_i \mathbf{v}_i^2\right]\delta q_j = \\ &= \sum_{j=1}^{n}\left[\frac{d}{dt}\frac{\partial T}{\partial \dot{q}_j} - \frac{\partial T}{\partial q_j}\right]\delta q_j. \end{aligned} \tag{4.7}$$

Mit den Gleichungen 4.3, 4.4 und 4.7 ergibt sich nun:

$$\sum_{j=1}^{n}\left[\frac{d}{dt}\frac{\partial T}{\partial \dot{q}_j} - \frac{\partial T}{\partial q_j} - Q_j\right]\delta q_j = 0 \tag{4.8}$$

Im nächsten Schritt wird nun angenommen, dass das N-Punktmassensystem k holonomen Zwangsbedingungen unterliegt. Weiters sollen die Transformationsgleichungen von Gleichung 4.2 $n = 3N - k$ unabhängige generalisierte Koordinaten q_j enthalten. Somit verschwinden die $3N - k$ eckigen Klammern wegen der Unabhängigkeit der virtuellen Verrückungen δq_j:

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{q}_j} - \frac{\partial T}{\partial q_j} - Q_j = 0 \qquad j = 1, ..., 3N - k \tag{4.9}$$

In der Mechanik kommen meist konservative Kräfte $\mathbf{F}_i(\mathbf{r}_1, ..., \mathbf{r}_N, t)$ vor, die gleich dem negativen Gradienten eines Potentials sind:

$$\mathbf{F}_i(\mathbf{r}, t) = -\frac{\partial V(\mathbf{r}, t)}{\partial \mathbf{r}_i} \qquad i = 1, ..., N$$

Die abgekürzte Schreibweise von $V(\mathbf{r}_1, ..., \mathbf{r}_N, t)$ ist hierbei $V(\mathbf{r}, t)$, womit die generalisierten Kräfte

$$Q_j = \sum_{i=1}^{N} \mathbf{F}_i \cdot \frac{\partial \mathbf{r}_i}{\partial q_j} = -\sum_{i=1}^{N} \frac{\partial V(\mathbf{r}, t)}{\partial \mathbf{r}_i} \cdot \frac{\partial \mathbf{r}_i}{\partial q_j} \tag{4.10}$$

lauten.
Dieser Ausdruck ist nach der Kettenregel die partielle Ableitung von

$$\hat{V}(q_1, .., q_{3N-k}, t) := V[\mathbf{r}(q_1, ..., q_{3N-k}, t), t].$$

Das Dach bei $\hat{V}(q, t)$ weist darauf hin, dass das Potential $\hat{V}$ im allgemeinen eine andere Form als V hat. In weiterer Folge wird das Dach des Ausdruckes weggelassen. Die Gleichung lautet somit:

$$Q_j = -\frac{\partial V(q_1, ..., q_{3N-k}, t)}{\partial q_j} \tag{4.11}$$

Als Beispiel können hierfür vor allem Gravitationskräfte, sowie Federkräfte genannt werden.

Die Lagrangefunktion $L(q_1, ..., q_{3N-k}, \dot{q}_1, ..., \dot{q}_{3N-k}, t)$ kann nun definiert werden:

$$L := T - V \tag{4.12}$$

Aus den Gleichungen 4.9, 4.11 und 4.12 oder 4.9 und 4.12 die **Lagrangegleichungen 2. Art** ergibt sich:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_j} - \frac{\partial L}{\partial q_j} = Q_j \qquad j = 1, ..., 3N - k \tag{4.13}$$

Sie gelten nur für holonome Zwangsbedingungen und für Kräfte, die nach 4.11 aus einem Potential abgeleitet werden können, sowie für allgemeine Kräfte, die durch Q_j berücksichtigt werden.
In der Lagrangeschen Mechanik geben die Lagrangefunktion und die Energie die wichtigen Terme an, die Skalare sind. Hingegen ist es in der Newtonschen Mechanik anders, hier geben Kraft und Impuls den Ton an, die beide vektoriell sind.
Es kann nun für eine Anwendung der Lagrangegleichungen 2. Art eine allgemein gültige Gebrauchsanweisung zur Aufstellung der Bewegungsgleichungen angeben werden:

a) Auswahl der generalisierten Koordinaten $q = (q_1, \ldots, q_n)$, mit $n = 3N - k$ im Raum, oder $n = 2N - k$ in der Ebene
b) Schreibe die Transformationsgleichungen $\vec{r}_i = (q_1, \ldots q_n)$, mit $i = 1, \ldots, N$, an
c) Bestimme die Lagrangefunktion $L = T - V$ als Funktion der n unabhängigen Koordinaten, der Geschwindigkeiten und eventuell der Zeit, $L = L(q, \dot{q}, t)$
d) Stelle die Lagrangegleichungen, wie in Gleichung 4.13 auf.

Die Lagrange Gleichungen 2. Art ergeben im Gegensatz zu der Beschreibung mit Hilfe des 2. Newtonschen Axioms, oder der Lagrange Gleichungen 1. Art, die minimale Anzahl an Gleichungen für die Beschreibung der Bewegung eines Systems von Massenpunkten. Ein Nachteil hingegen ist, dass die Zwangskräfte nicht direkt berechnet werden können.

Werden die auftretenden Kräfte durch eine Dissipationsfunktion, wie etwa Reibungskräfte, mit

$$P = P(\mathbf{v}_1, \ldots, \mathbf{v}_N, t)$$

beschrieben, so sind die generalisierten Kräfte

$$Q_j = \sum_{i=1}^{N} \mathbf{F}_i^* \frac{\partial \mathbf{r}_i}{\partial q_j} = -\frac{\partial P}{\partial \dot{q}_j}.$$

Die Lagrange Gleichungen 2. Art lauten in diesem Fall: [5]

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_j} - \frac{\partial L}{\partial q_j} = -\frac{\partial P}{\partial \dot{q}_j} \qquad j = 1, ..., n \tag{4.14}$$

Somit kann folgendes bestätigt werden: Wenn die Lagrangegleichungen in einem Koordinatensystem erfüllt sind, so sind sie es auch in jedem anderen Koordinatensystem. Die Lagrangegleichungen sind unter Punkttransformationen forminvariant, sie haben in allen Koordinaten- und Bezugssystemen die gleiche Form. Wegen der im allgemeinen verschiedenen Gestalt der Lagrangefunktionen ist das explizite Aussehen der Bewegungsgleichungen natürlich nicht in allen Koordinaten- und Bezugssystemen gleich.
Die Forminvarianz der Lagrangegleichungen ist von großer Nützlichkeit, da sie eine leichte und schnelle Aufstellung der Bewegungsgleichungen in allen Koordinatensystemen und auch in beschleunigten Bezugssystemen ermöglicht, wie sie in dieser Arbeit verwendet werden [8, Kapitel 3].

4.3.2 Wichtigkeit der Lagrangegleichungen 2. Art

> *„Die Wichtigkeit der Lagrangegleichungen 2. Art kann kaum überschätzt werden. Sie bilden das Herzstück der Lagrangeschen Mechanik und den Ausgangspunkt der formalen Weiterentwicklung der Mechanik. Aufgrund ihrer Einfachheit und Forminvarianz unter Punkttransformationen sowie der geradezu idealen Anpassung an holonome Nebenbedingungen eignen sie sich von allen Prinzipien und Formalismen oft am besten zur Aufstellung der Bewegungsgleichungen. Auch zur Konstruktion von Erhaltungsgrößen ist der Lagrangeformalisums wie*

geschaffen: Kontinuierliche Transformationen, die die Lagrangefunktion - eventuell bis auf eine Umweichung - invariant lassen, deuten auf Erhaltungsgrößen hin und ermöglichen deren einfache Berechnung. Darüber hinaus ist der Lagrangeformalismus auch für die klassischen Feldtheorien von großer Bedeutung [8, Seite 30].“

4.4 Maxima-Implementierung

Das zuvor theoretisch angedachte Modell soll nun mit dem Computer-Algebra-Programm *Maxima* umgesetzt werden. Das vollständige Maxima-Dokument kann wiederum auf der CD angesehen werden. In diesem Abschnitt werden die wichtigsten Punkte der Reihe nach abgearbeitet.

4.4.1 Generalisierte Koordinaten

Als erster Schritt werden alle benötigten Variablen mit den zugehörigen Abhängigkeiten definiert.

- $q\,(t)$, die generalisierten Koordinaten, mit $q_1(t)$ als Bogenlänge und $q_2(t)$ als Winkel des Fahrzeugs vom Koordinatenurpsrung aus
- $\psi\,(q_1)$, der aktuelle Winkel der Tangente der Bahn gemessen zum ruhenden Bezugskoordinatensystem
- $\rho\,(q_1)$, die 2. Ableitung der Bahnkoordinaten, auch als Krümmung bezeichnet
- $x\,(q_1)\,, y\,(q_1)$, die Koordinatenwerte der Mitte der Vorderachse

Die Anzahl der verallgemeinerten Koordinaten wird mit $n = 2$ vorgegeben. Diese sind die Variablen l, die zurückgelegte Bogenlänge der Strecke zum Startpunkt, sowie $\psi + \phi$, der absolute Winkel der Position des Fahrzeugs

bezogen auf den Koordinatenursprung. Mit diesen 2 generalisierten Koordinaten ist es möglich, die Lage und Orientierung des Fahrzeuges vollständig zu beschreiben.

4.4.2 Ortskoordinaten der Massenschwerpunkte

Die Ortskoordinaten der Massenschwerpunkte werden vom Koordinatenursprung zum jeweiligen Ort des Massenschwerpunktes festgelegt. Bei diesen Gleichungen ist eine Zwangsbedingung implizit enthalten, da die zwei Achsen immer denselben Abstand zueinander haben, was durch die Geometrie fest vorgegeben ist

$$\mathbf{r}_1 = \begin{pmatrix} x \\ y \end{pmatrix}, \qquad \mathbf{r}_2 = \begin{pmatrix} x - \cos(q_2)\ l_{car} \\ y - \sin(q_2)\ l_{car} \end{pmatrix} . \tag{4.15}$$

4.4.3 Eingeprägte Kräfte an den Massenschwerpunkten

Bei dem verwendeten Modell kommt nur eine eingeprägte Kraft vor. Diese bildet die Antriebskraft F_A an der Hinterachse ab, die in Richtung der Längsachse wirkt, da das Heck fest verankert ist

$$\mathbf{F}_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \mathbf{F}_2 = \begin{pmatrix} \cos(q_2)\ F_A \\ \sin(q_2)\ F_A \end{pmatrix} . \tag{4.16}$$

4.4.4 Verallgemeinerte Kräfte

Die verallgemeinerten Kräfte werden wie in Gleichung 4.4 beschrieben berechnet: Bei dem verwendeten Modell kommen 2 generalisierte Kräfte

vor:

$$\begin{aligned} Q_1 =& \mathbf{F}_1 \frac{d\mathbf{r}_1}{dq_1} + \mathbf{F}_2 \frac{\partial \mathbf{r}_2}{\partial q_1} = \sin(q_2) \left(\frac{\partial}{\partial q_1} y \right) F_A + \cos(q_2) \left(\frac{\partial}{\partial q_1} x \right) F_A \\ Q_2 =& \mathbf{F}_1 \frac{\partial \mathbf{r}_1}{\partial q_2} + \mathbf{F}_2 \frac{\partial \mathbf{r}_2}{\partial q_2} = 0 \end{aligned} \tag{4.17}$$

Auffällig ist, dass die verallgemeinerte Kraft Q_2 sich durch Differentiation der Ortskoordinaten mit der generalisierten Koordinate q_2 zu 0 ergibt. Dies ist physikalisch erklärbar, da der Antrieb mit seiner festen Geometrie nur in Richtung der ersten generalisierten Koordinate wirkt, der Wert in Richtung der zweiten generalisierten Koordinate ergibt daher Null.

4.4.5 Kinetische Energie

Für die Berechnung der kinetischen Energie werden zuerst die Geschwindigkeiten der zwei Ortskoordinaten der Massenschwerpunkte berechnet:

$$\mathbf{v}_i = \dot{\mathbf{r}}_i$$

Für die beiden Punkte ergeben sich die Geschwindigkeiten durch anwenden der Kettenregel zu:

$$\begin{aligned} \mathbf{v}_1 &= \begin{pmatrix} \frac{\partial x}{\partial q_1} \frac{\partial q_1}{\partial t} \\ \frac{\partial y}{\partial q_1} \frac{\partial q_1}{\partial t} \end{pmatrix} \\ \mathbf{v}_2 &= \begin{pmatrix} \frac{\partial x}{\partial q_1} \frac{\partial q_1}{\partial t} + sin(q_2) \frac{\partial q_2}{\partial t} l_{car} \\ \frac{\partial y}{\partial q_1} \frac{\partial q_1}{\partial t} + cos(q_2) \frac{\partial q_2}{\partial t} l_{car} \end{pmatrix} \end{aligned}$$

Mit den Geschwindigkeiten kann die kinetische Energie für die 2 Massenpunkte wie in Gleichung 4.6 berechnet werden:

$$\begin{aligned} T =& \frac{1}{2}\left(m_1\,\mathbf{v}_1^2 + m_2\,\mathbf{v}_2^2\right) = \\ =& \frac{m_2\left(\frac{d}{dt}\,q_2\right)^2 l_{car}^2 + \left(\frac{d}{dt}\,q_1\right)\left(\left(m_2+m_1\right)\left(\frac{d}{dt}\,q_1\right) - 2\,m_2\left(\frac{d}{dt}\,q_2\right)\sin\left(\psi - q_2\right)\,l_{car}\right)}{2} \end{aligned} \tag{4.18}$$

4.4.6 Potentielle Energie

Die einzige potentielle Energie entsteht hierbei durch die Verdrehung des Führungskiels, der dafür sorgt, dass das Fahrzeug in der Spur bleibt. Wird der Führungskiel verdreht, sorgt eine Feder für die Rückstellung in seine ursprüngliche Position. Durch die Anordnung geschieht diese Rückstellung über eine Drehbewegung. Das entsprechende Rückstellmoment wird mit

$$\mathbf{M}_F = -\alpha_F\,k_F \tag{4.19}$$

abgebildet, wobei α_F den Verdrehungswinkel beschreibt und k_F die Federkonstante bezeichnet. Das zugehörige Potential lautet im vorliegenden Fall mit $\alpha_F = q_2 - \psi$:

$$V = \frac{\left(q_2 - \psi\right)^2 k_F}{2} \tag{4.20}$$

Das Rückstellmoment hat bei den Fahrzeugen einen geringen Einfluss auf die Bewegung. Durch Gleitreibungsterme, die in Folge noch behandelt werden, entsteht beim Vorwärtsfahren ebenfalls ein Moment um die Fahrzeughochachse, das eine ähnliche rückstellende Wirkung hat.

4.4.7 Dissipationsfunktionen

Die auftretenden Dissipationsfunktionen entstehen durch die Rollreibung und die Gleitreibung.

Rollreibung

Die Rollreibung ist jene Kraft, die beim Abrollen eines Rades entsteht und der Bewegung entgegengerichtet ist. Durch den Einfluss der Magnete des Carrera-Autos ist der Wert deutlich größer als ohne Magnete.
Der grundsätzliche Term der Rollreibung lautet:

$$\mathbf{F}_R = \mu_R \cdot \mathbf{F}_N \tag{4.21}$$

Es ist aus der Formel zu erkennen, dass die Reibungskraft dabei unabhängig von der Reibungsfläche der Körper ist und nur von der Normalkraft F_N des aufliegenden Körpers und vom Reibungskoeffizienten μ_R abhängt. Umso glatter die Flächen sind, umso kleiner wird dabei der Wert des Koeffizienten. Als erste Idee für die Kraft der Rollreibung mit der Tatsache, dass die Reibung der Bewegung entgegen gesetzt ist, wurde eine *signum*-Funktion verwendet:

$$\mathbf{F}_R = -\text{signum}\left(\frac{d}{d\,t}\,q_1\right)\,\left|\cos\left(\psi - q_2\right)\right|\,g\,m\,\mu_R \tag{4.22}$$

Die Geschwindigkeit erfolgt dabei immer in Richtung der Fahrzeuglängsachse. Die Dissipationsfunktion würde in diesem Fall wie folgt aussehen:

$$P_R = \left|\frac{d}{d\,t}\,q_1\right|\,\left|\cos\left(\psi - q_2\right)\right|\,g\,m\,\mu_R \tag{4.23}$$

Die *signum*-Funktion müsste umständlich per Hand implementiert werden, da das Ableiten des Potentials in diesem Fall nicht wie üblich ausgeführt

werden kann. Aus diesem Grund musste ein für das Modell geeigneterer Term gefunden werden. Es soll eine Funktion sein, die im Gegensatz zur *signum*-Funktion überall differenzierbar ist. Aus diesem Grund wurde eine *tanh*-Funktion für das Modell verwendet, da es mit dieser möglich ist sie direkt bei der gesamten Berechnung des Modells einzubinden. Bei einer passenden Skalierung der x-Achse der *tanh*-Funktion sieht diese der *signum*-Funktion praktisch beliebig ähnlich und kann daher mit geringer Abweichung verwendet werden. Die verwendete Rollreibung wird nun mit

$$\mathbf{F}_R = -m\, g\, \mu_R \tanh\left(k_R\, v_R\right) \left|\cos\left(\psi - q_2\right)\right| \tag{4.24}$$

beschrieben. Für die Geschwindigkeit in Fahrzeuglängsachse wird wieder der Term

$$\left(\frac{d}{d\,t}\, q_1\right) \cos\left(\psi - q_2\right)$$

eingesetzt. Da die Reibung aber nicht negativ werden kann, auch wenn das Fahrzeug verkehrt steht, wird obige Formel erweitert zu:

$$\left(\frac{d}{d\,t}\, q_1\right) \left|\cos\left(\psi - q_2\right)\right| \tag{4.25}$$

Steht das Auto verkehrt auf der Strecke und rollt in Fahrtrichtung bedeutet dies, dass sich die Rollrichtung der Räder geändert hat.
Für die verwendete Gleichung der Rollreibung ergibt sich die benötigte Dissipationsfunktion zu:

$$P_R = \frac{m\, g\, \mu_R \ln\left(\cosh\left(\left(\frac{d}{d\,t} q_1\right) \left|\cos\left(\psi - q_2\right)\right| k_R\right)\right)}{k_R} \tag{4.26}$$

Zur Überprüfung kann nun die Dissipationsfunktion abgeleitet werden, daraus entsteht die gesuchte Reibungskraft, wie in Gleichung 4.24 beschrieben:

$$\frac{d}{d\left(\frac{d}{d\,t}\, q_1\right)} P_R = -m \left|\cos\left(\psi - q_2\right)\right| g\, \mu_R \tanh\left(\left(\frac{d}{d\,t}\, q_1\right) \left|\cos\left(\psi - q_2\right)\right| k_R\right) \tag{4.27}$$

Gleitreibung Hinterachse

Die Gleitreibung tritt bei Objekten auf, die nicht abrollen können. Wird die maximale Haftreibung überschritten, beginnt ein Körper zu gleiten, dabei wirkt der Zugkraft die Gleitreibungskraft entgegen. Ist die Zugkraft weiterhin größer als die Gleitreibungskraft, so wird der Gegenstand weiter beschleunigt. Auch hier ist der Gleitreibungskoeffizient abhängig von den Materialien, die aufeinander treffen und der Oberflächenbeschaffenheit der Flächen.
Auch hier tritt bei Verwendung der *signum*-Funktion für die Kraft

$$\mathbf{F}_{Gl} = -m_2\, g\, \mu_{Gl}\, \text{signum}\,(k_{Gl}\, v_{Gl}) \tag{4.28}$$

dasselbe Problem auf, wie im oberen Absatz für die Rollreibung bereits erklärt wurde. Aus diesem Grund wird ebenfalls die *tanh*-Funktion als adäquater Ersatz herangezogen:

$$\mathbf{F}_{Gl} = -m_2\, g\, \mu_{Gl}\, \tanh\,(k_{Gl}\, v_{Gl}) \tag{4.29}$$

Die benötigte Dissipationsfunktion ergibt sich somit zu:

$$P_{Gl} = \frac{m_2\, g\, \mu_{Gl}\, \ln\,(\cosh\,(k_{Gl}\, v_{Gl}))}{k_{Gl}}$$

Für die Ermittlung der Geschwindigkeiten an der Hinterachse in Fahrzeugkoordinaten wird als erstes anhand der Rotationsmatrix

$$Rot = \begin{pmatrix} \cos\,(q_2) & \sin\,(q_2) \\ -\sin\,(q_2) & \cos\,(q_2) \end{pmatrix}$$

die Geschwindigkeit in Fahrzeuglängsachse und senkrecht zur Fahrzeuglängsachse berechnet:

$$\mathbf{v}_{HECK} = Rot * \mathbf{v}_2$$

$$\mathbf{v}_{HECK} = \begin{pmatrix} \left(\frac{d}{dt}\, q_1\right) \cos\,(\psi - q_2) \\ \left(\frac{d}{dt}\, q_1\right) \sin\,(\psi - q_2) - \left(\frac{d}{dt}\, q_2\right)\, l_{car} \end{pmatrix} \tag{4.30}$$

Die Geschwindigkeit normal zur Fahrzeuglängsachse ist diejenige, die für die Berechnung der Kraft benötigt wird. Durch Einsetzen der Geschwindigkeit ergibt sich nun die vollständige Dissipationsfunktion für die Gleitreibung an der Hinterachse:

$$P_{Gl} = \frac{m_2\, g\, \mu_{Gl} \ln\left(\cosh\left(\left(\left(\frac{d}{dt}\, q_1\right)\, \sin\left(\psi - q_2\right) - \left(\frac{d}{dt}\, q_2\right)\, l_{car}\right)\, k_{Gl}\right)\right)}{k_{Gl}} \tag{4.31}$$

Durch die Ableitung der Dissipationsfunktion in den Langrange-Gleichungen 2. Art mit den generalisierten Koordinaten ergibt sich daraus sowohl ein Term in Richtung Fahrzeugslängsachse, als auch normal dazu. Das bedeutet, dass das Gleiten des Fahrzeugs bei einem Drift, sowohl einen Einfluss auf die Geschwindigkeit hat, als auch einen Einfluss auf die Winkelgeschwindigkeit.

Drehreibung Vorderachse

Für ein genaueres Modell wurde die Drehreibung an der Vorderachse des Fahrzeuges berücksichtigt. Fährt das Slot-Car in der vorgegebenen Spur entsteht durch die Fahrzeugdrehung um die Hochachse in den Kurven ein Reibungsterm. Kommt eine Kurve dreht sich am Kurveneingang der Leitkiel etwas zur Kurveninnenseite, da durch die Fahrtgeschwindigkeit und die Trägheit das Fahrzeug am Beginn weiter geradeaus schiebt. Die Vorderachse dreht sich und dadurch drehen sich die Räder der Vorderachse mit. Bei dieser Drehung tritt wieder eine Art Gleitreibung, aber an der Vorderachse, auf, die Drehreibung genannt wird und mit folgendem Term im Modell berücksichtigt ist:

$$P_D = \frac{m_1\, b_{car}\, g\, \mu_R \ln\left(\cosh\left(\left(\frac{d}{dt}\, q_2\right)\, k_R\right)\right)}{2\, k_R} \tag{4.32}$$

Durch die Differentiation der Dissipationsfunktion mit den generalisierten Koordinaten wird der gewünschte Term erhalten:

$$\mathbf{F}_D = \frac{m_1\, b_{car}\, g\, \mu_R \tanh\left(\left(\frac{d}{dt} q_2\right)\, k_R\right)}{2} \tag{4.33}$$

Auch hier wurde wieder anstatt einer *signum*-Funktion die günstigere *tanh*-Funktion verwendet.
Die gesamte Dissipationsfunktion setzt sich aus den drei Termen für Rollreibung, Gleitreibung der Hinterachse und Drehreibung der Vorderachse zusammen:

$$P = P_R + P_{Gl} + P_D$$

$$\begin{aligned} P = &\frac{(m_2 + m_1)\, g\, \mu_R \ln\left(\cosh\left(\left(\frac{d}{dt} q_1\right)\, |\cos(\psi - q_2)|\, k_R\right)\right)}{k_R} + \\ &\frac{m_1\, b_{car}\, g\, \mu_R \ln\left(\cosh\left(\left(\frac{d}{dt} q_2\right)\, k_R\right)\right)}{2\, k_R} + \\ &\frac{m_2\, g\, \mu_{Gl} \ln\left(\cosh\left(\left(\left(\frac{d}{dt} q_1\right)\, \sin(\psi - q_2) - \left(\frac{d}{dt} q_2\right)\, l_{car}\right)\, k_{Gl}\right)\right)}{k_{Gl}} \end{aligned} \tag{4.34}$$

4.4.8 Lagrange-Funktion

Die Lagrange-Funktion kann nun mit der kinetischen Energie und der potentiellen Energie wie in Gleichung 4.12 berechnet werden:

$$\begin{aligned} L = -\frac{1}{2}\Bigg(&\left(\psi^2 + q_2\, (q_2 - 2\,\psi)\right)\, k_F - m_2 \left(\frac{d}{dt} q_2\right)^2 l_{car}^2 + \\ &\left(\frac{d}{dt} q_1\right)\left(2\, m_2\, \left(\frac{d}{dt} q_2\right)\, \sin(\psi - q_2)\, l_{car} - (m_2 + m_1)\, \left(\frac{d}{dt} q_1\right)\right)\Bigg) \end{aligned} \tag{4.35}$$

4.4.9 Lagrangegleichungen 2. Art

Mit den Gleichungen aus 4.14 ergibt sich Gleichung 1 zu

$$\begin{aligned}
&\psi\left(\frac{d}{d\,q_1}\,\psi\right)\,k_F-q_2\left(\frac{d}{d\,q_1}\,\psi\right)\,k_F-m_2\left(\frac{d^2}{d\,t^2}\,q_2\right)\,\sin\left(\psi-q_2\right)\,l_{car}+\\
&+m_2\left(\frac{d}{d\,t}\,q_2\right)^2\cos\left(\psi-q_2\right)\,l_{car}+m_2\left(\frac{d^2}{d\,t^2}\,q_1\right)+m_1\left(\frac{d^2}{d\,t^2}\,q_1\right)=-\\
&-\frac{m_2\,\left|\cos\left(\psi-q_2\right)\right|\,g\,\mu_R\sinh\left(\left(\frac{d}{d\,t}\,q_1\right)\,\left|\cos\left(\psi-q_2\right)\right|\,k_R\right)}{\cosh\left(\left(\frac{d}{d\,t}\,q_1\right)\,\left|\cos\left(\psi-q_2\right)\right|\,k_R\right)}-\\
&-\frac{m_1\,\left|\cos\left(\psi-q_2\right)\right|\,g\,\mu_R\sinh\left(\left(\frac{d}{d\,t}\,q_1\right)\,\left|\cos\left(\psi-q_2\right)\right|\,k_R\right)}{\cosh\left(\left(\frac{d}{d\,t}\,q_1\right)\,\left|\cos\left(\psi-q_2\right)\right|\,k_R\right)}+\\
&+\sin\left(q_2\right)\,\left(\frac{d}{d\,q_1}\,y\right)\,F_A+\cos\left(q_2\right)\,\left(\frac{d}{d\,q_1}\,x\right)\,F_A+\\
&+\frac{m_2\sin\left(\psi-q_2\right)\,g\,\mu_{Gl}\sinh\left(\left(\frac{d}{d\,t}\,q_2\right)\,l_{car}\,k_{Gl}-\left(\frac{d}{d\,t}\,q_1\right)\,\sin\left(\psi-q_2\right)\,k_{Gl}\right)}{\cosh\left(\left(\frac{d}{d\,t}\,q_2\right)\,l_{car}\,k_{Gl}-\left(\frac{d}{d\,t}\,q_1\right)\,\sin\left(\psi-q_2\right)\,k_{Gl}\right)}
\end{aligned}\tag{4.36}$$

und Gleichung 2 zu

$$\begin{aligned}
&-\psi\,k_F+q_2\,k_F+m_2\left(\frac{d^2}{d\,t^2}\,q_2\right)\,l_{car}^2-\\
&-m_2\left(\frac{d^2}{d\,t^2}\,q_1\right)\,\sin\left(\psi-q_2\right)\,l_{car}-m_2\left(\frac{d}{d\,t}\,q_1\right)^2\left(\frac{d}{d\,q_1}\,\psi\right)\,\cos\left(\psi-q_2\right)\,l_{car}=-\\
&-\frac{m_1\,b_{car}\,g\,\mu_R\sinh\left(\left(\frac{d}{d\,t}\,q_2\right)\,k_R\right)}{2\cosh\left(\left(\frac{d}{d\,t}\,q_2\right)\,k_R\right)}-\\
&-\frac{m_2\,l_{car}\,g\,\mu_{Gl}\sinh\left(\left(\frac{d}{d\,t}\,q_2\right)\,l_{car}\,k_{Gl}-\left(\frac{d}{d\,t}\,q_1\right)\,\sin\left(\psi-q_2\right)\,k_{Gl}\right)}{\cosh\left(\left(\frac{d}{d\,t}\,q_2\right)\,l_{car}\,k_{Gl}-\left(\frac{d}{d\,t}\,q_1\right)\,\sin\left(\psi-q_2\right)\,k_{Gl}\right)}\quad.
\end{aligned}\tag{4.37}$$

An diesen zwei Gleichungen ist die Komplexität des Modells sehr gut zu erkennen. Eine Berechnung per Hand ist sehr aufwendig. Um diese Gleichungen im Simulink-Modell verwenden zu können, müssen sie so umgeformt werden, dass auf der linken Seite nur mehr die zweite Ableitung der Zustandsgrößen x stehen. Diese Umformung kann sehr gut in Vektorform angeschrieben werden:

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{r} = \boldsymbol{\tau}$$

mit

$$\ddot{\mathbf{q}} = \begin{pmatrix} \ddot{q_1} \\ \vdots \\ \ddot{q_n} \end{pmatrix} \qquad \dot{\mathbf{q}} = \begin{pmatrix} \dot{q_1} \\ \vdots \\ \dot{q_n} \end{pmatrix}$$

Somit sind zuerst alle Werte der linken Seite, die kein $\ddot{q}$ enthalten auf die rechte Seite zu bringen. Anschließend muss noch die Massenmatrix M invertiert und mit der Gleichung multipliziert werden. Daraus ergibt sich die gewünschte Form für Matlab/Simulink.

4.4.10 Massenmatrix

Die Massenmatrix kann wie folgt erstellt werden:

$$\mathbf{M} = \begin{pmatrix} \frac{\partial^2 T}{\partial \dot{q_1}^2} & \frac{\partial^2 T}{\partial \dot{q_1} \partial \dot{q_2}} & \cdots & \frac{\partial^2 T}{\partial \dot{q_1} \partial \dot{q_n}} \\ \frac{\partial^2 T}{\partial \dot{q_2} \partial \dot{q_1}} & \frac{\partial^2 T}{\partial \dot{q_2}^2} & \cdots & \frac{\partial^2 T}{\partial \dot{q_2} \partial \dot{q_n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 T}{\partial \dot{q_n} \partial \dot{q_1}} & \frac{\partial^2 T}{\partial \dot{q_n} \partial \dot{q_2}} & \cdots & \frac{\partial^2 T}{\partial \dot{q_n}^2} \end{pmatrix} \tag{4.38}$$

Obige Berechnung angewandt auf die kinetische Energie ergibt:

$$\mathbf{M} = \begin{pmatrix} m_2 + m_1 & -m_2 \sin(\psi - q_2)\ l_{car} \\ -m_2 \sin(\psi - q_2)\ l_{car} & m_2\, l_{car}^2 \end{pmatrix} \tag{4.39}$$

4.4.11 Rechte Seite der Differentialgleichungen

Nach den oben beschriebenen Umformungen ist das Ergebnis der rechten Seite für die 1. Gleichung

$$
\begin{aligned}
-\Bigg(& (m_2 + m_1)\ |\cos(\psi - q_2)|\ g\,\mu_R \tanh\left(\left(\frac{d}{d\,t}\,q_1\right)\ |\cos(\psi - q_2)|\ k_R\right) + \\
& + \psi\left(\frac{d}{d\,q_1}\,\psi\right)\ k_F - \left(q_2\left(\frac{d}{d\,q_1}\,\psi\right)\ k_F + \right. \\
& \left. + \left(\sin(q_2)\left(\frac{d}{d\,q_1}\,y\right) + \cos(q_2)\left(\frac{d}{d\,q_1}\,x\right)\right)\ F_A\right) + \\
& + m_2\left(\left(\frac{d}{d\,t}\,q_2\right)^2 \cos(\psi - q_2)\ l_{car} - \sin(\psi - q_2)\ g\,\mu_{Gl}\cdot\right. \\
& \left.\cdot\tanh\left(\left(\left(\frac{d}{d\,t}\,q_2\right)\ l_{car} - \left(\frac{d}{d\,t}\,q_1\right)\ \sin(\psi - q_2)\right)\ k_{Gl}\right)\right)\Bigg)
\end{aligned}
\tag{4.40}
$$

und für die 2. Gleichung:

$$
\begin{aligned}
& -\frac{m_1\,b_{car}\,g\,\mu_R \tanh\left(\left(\frac{d}{d\,t}\,q_2\right)\ k_R\right)}{2} + (\psi - q_2)\ k_F + \\
& + m_2\,l_{car}\left(\left(\frac{d}{d\,t}\,q_1\right)^2\left(\frac{d}{d\,q_1}\,\psi\right)\cos(\psi - q_2) - g\,\mu_{Gl}\cdot\right. \\
& \left.\cdot\tanh\left(\left(\left(\frac{d}{d\,t}\,q_2\right)\ l_{car} - \left(\frac{d}{d\,t}\,q_1\right)\ \sin(\psi - q_2)\right)\ k_{Gl}\right)\right)
\end{aligned}
\tag{4.41}
$$

4.4.12 Zustandsgleichungen: System 1. Ordnung

Als letzter Schritt bleibt nun noch die Vereinfachung des Systems auf ein System 1. Ordnung. Nach Einführen der Zustandsgrößen

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} q_1 \\ \dot{q_1} \\ q_2 \\ \dot{q_2} \end{pmatrix}$$

lässt sich das System in Zustandsraumdarstellung umschreiben.Werden diese Vereinfachungen in die zwei Gleichungen von Abschnitt 4.4.11 eingesetzt, ist das Ergebnis durch folgende vier Gleichungen gegeben:

$$\frac{d}{d\,t}\,x_1 = x_2 \tag{4.42}$$

$$\begin{aligned}\frac{d}{d\,t}\,x_2 = [&(2\,m_2 + 2\,m_1)\ |\cos(\psi - z_3)| \cdot l_{car}\,g\,\mu_R \tanh(z_2\ |\cos(\psi - z_3)|\ k_R) + \\ &+ m_1 \sin(\psi - z_3)\ b_{car}\,g\,\mu_R \tanh(z_4\,k_R) + \\ &+ ((2\,\psi - 2\,z_3)\ \rho\,l_{car} + (2\,z_3 - 2\,\psi)\ \sin(\psi - z_3))\ k_F + \\ &+ (-2\sin(z_3)\ \sin(\psi) - 2\cos(z_3)\ \cos(\psi))\ l_{car}\,F_A + \\ &+ 2\,m_2\,z_4^2 \cos(\psi - z_3)\ l_{car}^2 - 2\,m_2\,z_2^2 \cos(\psi - z_3)\sin(\psi - z_3)\ \rho\,l_{car}\Big]\ / \\ /&\left(\left(2\,m_2 \sin(\psi - z_3)^2 - 2\,m_2 - 2\,m_1\right) l_{car}\right)\end{aligned} \tag{4.43}$$

$$\frac{d}{d\,t}\,x_3 = x_4 \tag{4.44}$$

$$\begin{aligned}\frac{d}{dt}x_4 = &-[(m_2\sin(\psi - z_3)\ l_{car}\ g\ \mu_R\tanh(z_2\ |\cos(\psi - z_3)|\ k_R) + \\ &+ \psi\,\rho\,k_F - (z_3\,\rho\,k_F + (\sin(z_3)\ \sin(\psi) + \cos(z_3)\ \cos(\psi))\ F_A) + \\ &+ m_2\ \left(z_4^2\cos(\psi - z_3)\ l_{car} - \right. \\ &\left. -\sin(\psi - z_3)\ g\ \mu_{Gl}\tanh((z_4\,l_{car} - z_2\sin(\psi - z_3))\ k_{Gl})\right))\ / \\ &/\ (m_1 + m_2) - \left(-\frac{m_1\,b_{car}\,g\,\mu_R\tanh(z_4\,k_R)}{2} + m_2\,l_{car}\left(z_2^2\cos(\psi - z_3)\ \rho - \right.\right. \\ &\left.\left. -g\,\mu_{Gl}\tanh((z_4\,l_{car} - z_2\sin(\psi - z_3))\ k_{Gl}))\right)] \\ &/\left(m_2\left(1 - \frac{m_2\sin(\psi - z_3)^2}{m_2 + m_1}\right)l_{car}^2\right)\end{aligned} \tag{4.45}$$

Diese Gleichungen können in dieser Form auch in Matlab übernommen werden und kommen so auch in der Simulationsumgebung Simulink im Modell vor.

4.4.13 Maxima-Code

Modell der Carrera-Bahn Autos

1 Initialisierung

Alle Variablen loeschen.
„scifac“Package laden fuer bessere Formelvereinfachung mit „gcfac“-Befehl.

```
(%i184) remvalue(all)$
        kill(all)$
        load(scifac)$
        load(linearalgebra)$
        load(absimp)$
```

Warning : $WITH - COMPILATION - UNITisbeingredefined.$

2 Verallgemeinerte Koordinaten

Vereinbaren der Variablen.
Die Abhaengigkeiten sind wichtig fuer die Kettenregel bei der Ableitung.

```
(%i4)  declare(t,scalar)$      /* Zeit */
       declare(q,nonscalar)$

       declare(%psi,scalar)$
       declare(%rho,scalar)$

       depends(q,t)$
       depends(%psi,[q[1]])$
       depends(%rho,[q[1]])$
       depends(x,q[1])$
       depends(y,q[1])$
```

Anzeigen aller vereinbarter Abhaengigkeiten.

```
(%i13) dependencies;
```

$$(\%o13)\quad [q(t), \psi(q_1), \rho(q_1), x(q_1), y(q_1)]$$

Anzahl der verallgemeinerten Koordinaten

```
(%i14) n:2$
       display(n)$
```

$n = 2$

3 Ortskoordinaten der Massenschwerpunkte

Definition der Ortvektoren der Massenschwerpunkte

```
(%i16) r[1]:matrix([x],[y])$
       r[2]:r[1]-l[car]*matrix([cos(q[2])],[sin(q[2])])$
       display(r[1],r[2])$
```

$$r_1 = \begin{pmatrix} x \\ y \end{pmatrix} \qquad r_2 = \begin{pmatrix} x - \cos(q_2)\ l_{car} \\ y - \sin(q_2)\ l_{car} \end{pmatrix}$$

Anzahl der Massenschwerpunkte

```
(%i19) N:length(listarray(r))$
```

4 Eingepraegte Kraefte an den Massenschwerpunkten

Definition der Kraefte auf die Massen.

```
(%i20) F[1]:matrix([0],[0])$
       F[2]:F[A]*matrix([cos(q[2])],[sin(q[2])])$
       display(F[1],F[2])$
```

$$F_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad F_2 = \begin{pmatrix} \cos(q_2)\ F_A \\ \sin(q_2)\ F_A \end{pmatrix}$$

5 Verallgemeinerte Kraefte

Berechnung der verallgemeinerten Kraefte.

```
(%i23) Q(j,N):= sum(F[i].diff(r[i],q[j]),i,1,N);
```

$$(\%o23)\quad \mathrm{Q}(j,N) := \sum_{i=1}^{N} F_i.\mathrm{diff}\left(r_i, q_j\right)$$

```
(%i24)  for i_n:1 thru n do
            block(  Q[i_n]:Q(i_n,N),
                    display(Q[i_n])   )$
```

$$Q_1 = \sin(q_2)\left(\frac{d}{d\,q_1}\,y\right)F_A + \cos(q_2)\left(\frac{d}{d\,q_1}\,x\right)F_A \qquad Q_2 = 0$$

6 Kinetische Energie

6.1 Geschwindigkeiten

```
(%i25)  for n:1 thru length(listarray(r)) do
            block(
              v[n]:diff(r[n],t),
              display(v[n])
            )$
```

$$v_1 = \begin{pmatrix} \left(\frac{d}{d\,t}\,q_1\right)\left(\frac{d}{d\,q_1}\,x\right) \\ \left(\frac{d}{d\,t}\,q_1\right)\left(\frac{d}{d\,q_1}\,y\right) \end{pmatrix} \qquad v_2 = \begin{pmatrix} \left(\frac{d}{d\,t}\,q_1\right)\left(\frac{d}{d\,q_1}\,x\right) + \sin(q_2)\left(\frac{d}{d\,t}\,q_2\right)l_{car} \\ \left(\frac{d}{d\,t}\,q_1\right)\left(\frac{d}{d\,q_1}\,y\right) - \cos(q_2)\left(\frac{d}{d\,t}\,q_2\right)l_{car} \end{pmatrix}$$

6.2 Kinetische Energie

```
(%i26)  rCurve(glg):=block(
            dxdl:diff(x,q[1])=cos(%psi),
            dydl:diff(y,q[1])=sin(%psi),
            ddxddl:diff(x,q[1],2)=(-sin(%psi))*%rho,
            ddyddl:diff(y,q[1],2)=(cos(%psi))*%rho,
            dpsidl:diff(%psi,q[1],1)=%rho,
            glg:subst(rhs(dxdl),lhs(dxdl),glg),
            glg:subst(rhs(dydl),lhs(dydl),glg),
            glg:subst(rhs(ddxddl),lhs(ddxddl),glg),
            glg:subst(rhs(ddyddl),lhs(ddyddl),glg),
            glg:subst(rhs(dpsidl),lhs(dpsidl),glg),
            gcfac((glg))
        )$
```

```
(%i27) T:1/2*(m[1]*v[1].v[1]+m[2]*v[2].v[2])$
       T:gcfac(trigreduce(rCurve(T)))$
       display(T)$
```

$$T = \frac{m_2 \left(\frac{d}{dt} q_2\right)^2 l_{car}^2 + \left(\frac{d}{dt} q_1\right) \left((m_2 + m_1) \left(\frac{d}{dt} q_1\right) - 2\, m_2 \left(\frac{d}{dt} q_2\right) \sin\left(\psi - q_2\right) l_{car}\right)}{2}$$

7 Potentielle Energie

Federkraft und entsprechendes Potential der Federkraft.

```
(%i30) F[F]:-k[F]*(%alpha[F])$
       display(F[F])$
       V:-'integrate(F[F],(%alpha[F]))$
       display(V)$

       %alpha[F]:q[2]-%psi$
       V:-integrate(F[F],('(%alpha[F])))$
       V:ev(V,nouns)$
       display(V)$
```

$$F_F = -\alpha_F\, k_F$$
$$V = k_F \int \alpha_F d\alpha_F$$
$$V = \frac{(q_2 - \psi)^2\, k_F}{2}$$

Verallgemeinerte Kräfte Aufgrund des Federkraftpotentials:

```
(%i38) -diff(V,q[1]);
```

$$(\%o38)\quad (q_2 - \psi) \left(\frac{d}{d\, q_1} \psi\right) k_F$$

```
(%i39) -diff(V,q[2]);
```

$$(\%o39)\quad -\left(q_2 - \psi\right) k_F$$

8 Dissipationsfunktion

8.1 Geschwindigkeiten am Heck

Geschwindigkeit am Heck

```
(%i40)  v[2]:rCurve(v[2])$
```

Rotationsmatrix um q2 –> zur Berechnung der Geschwindigkeitsanteile in Fahrzeuglaengs und -querrichtungen.

```
(%i41)  Rot:matrix([cos(q[2]),sin(q[2])],[-sin(q[2]),cos(q[2])])$
        display(Rot)$
```

$$Rot = \begin{pmatrix} \cos(q_2) & \sin(q_2) \\ -\sin(q_2) & \cos(q_2) \end{pmatrix}$$

Geschwindigkeit am Heck in Fahrzeugkoordinaten.

```
(%i43)  v[HECK]:trigreduce(Rot.v[2])$
        display(v[HECK])$
```

$$v_{HECK} = \begin{pmatrix} \left(\frac{d}{dt}\, q_1\right) \cos(\psi - q_2) \\ \left(\frac{d}{dt}\, q_1\right) \sin(\psi - q_2) - \left(\frac{d}{dt}\, q_2\right)\, l_{car} \end{pmatrix}$$

8.2 Fahrzeuggesamtmasse

Definition Gesamtmasse Fahrzeug

```
(%i45)  m[CAR]:m[1]+m[2]$
        display(m[CAR])$
```

$$m_{CAR} = m_2 + m_1$$

8.3 Gleitreibungsterm am Hinterrad in Fahrzeugquerrichtung.

Ansatz fuer Reibkraft normal zur Fahrzeuglaengsachse.

```
(%i47) kill(v[Gl])$
       F[Gl]:-m[2]*g*mu[Gl]*tanh(k[Gl]*v[Gl])$
       display(F[Gl])$
```

$$F_{Gl} = -m_2\, g\, \mu_{Gl} \tanh\left(k_{Gl}\, v_{Gl}\right)$$

Geschwindigkeit normal zur Fahrzeuglaengsachse.

```
(%i50) vGL:v[HECK][2,1]$
       disp(vGL)$
```

$$\left(\frac{d}{d\,t}\,q_1\right) \sin\left(\psi - q_2\right) - \left(\frac{d}{d\,t}\,q_2\right) l_{car}$$

Dissipationsfunktion

```
(%i52) P[Gl]:-integrate(F[Gl],v[Gl])$
       display(P[Gl])$

       P[Gl]:subst(vGL,v[Gl],P[Gl])$
       v[Gl]:vGL$
       display(v[Gl])$
       display(P[Gl])$
```

$$P_{Gl} = \frac{m_2\, g\, \mu_{Gl} \log\left(\cosh\left(k_{Gl}\, v_{Gl}\right)\right)}{k_{Gl}}$$

$$v_{Gl} = \left(\frac{d}{d\,t}\,q_1\right) \sin\left(\psi - q_2\right) - \left(\frac{d}{d\,t}\,q_2\right) l_{car}$$

$$P_{Gl} = \frac{m_2\, g\, \mu_{Gl} \log\left(\cosh\left(\left(\left(\frac{d}{d\,t}\,q_1\right) \sin\left(\psi - q_2\right) - \left(\frac{d}{d\,t}\,q_2\right) l_{car}\right) k_{Gl}\right)\right)}{k_{Gl}}$$

```
(%i58)  print("Kontrolle ")$
        for i_n:1 thru n do
           block(
             print(" "),
             QGL[i_n]:gcfac(trigreduce(-diff(P[Gl],diff(q[i_n],t)))),
             display(QGL[i_n])
        )$
```

Kontrolle

$$QGL_1 = -m_2 \sin(\psi - q_2)\, g\, \mu_{Gl} \tanh\left(\left(\left(\frac{d}{dt} q_1\right) \sin(\psi - q_2) - \left(\frac{d}{dt} q_2\right) l_{car}\right) k_{Gl}\right)$$

$$QGL_2 = m_2\, l_{car}\, g\, \mu_{Gl} \tanh\left(\left(\left(\frac{d}{dt} q_1\right) \sin(\psi - q_2) - \left(\frac{d}{dt} q_2\right) l_{car}\right) k_{Gl}\right)$$

8.4 Rollreibung Culombscher Ansatz

Culombscher Ansatz –> schlecht fuer numerische Integration

Dissipationsfunktionen fuer Reibkraft. (Nicht verwendet, stattdessen Reibkraefte R)

```
(%i60)  P[R]:-'integrate(-m*g*mu[R],v[R]);
        P[R]:subst(abs('diff(q[1],t)*cos(q[2]-%psi)),v[R],P[R])$
        display(P[R])$
        'diff('P[R],'diff(q[1],t))=diff(P[R],'diff(q[1],t));
        R[1]:-m*g*mu[R]*signum(diff(q[1],t))*abs(cos(q[2]-%psi))$
        display(R[1])$
        'diff('P[R],'diff(q[2],t))=diff(P[R],'diff(q[2],t));
        R[2]:-0$
        display(R[2])$
```

(%o60) $g\, m\, \mu_R\, v_R$

$$P_R = \left|\frac{d}{dt} q_1\right| \left|\cos(\psi - q_2)\right| g\, m\, \mu_R$$

(%o63) $$\frac{d}{d\left(\frac{d}{dt}q_1\right)}P_R = \frac{\left(\frac{d}{dt}q_1\right)\,|\cos(\psi - q_2)|\,g\,m\,\mu_R}{\left|\frac{d}{dt}q_1\right|}$$

$$R_1 = \text{signum}\left(\frac{d}{dt}q_1\right)\,|\cos(\psi - q_2)|\,g\,m\,\mu_R$$

(%o66) $$\frac{d}{d\left(\frac{d}{dt}q_2\right)}P_R = 0$$

$$R_2 = 0$$

8.5 Rollreibung tanh

Ansatz fuer Reibkraft in Richtung Fahrzeuglaengsachse.

```
(%i69) kill(v[R])$
       F[R]:gcfac(-m[CAR]*g*mu[R]*tanh(k[R]*v[R]))$
       display(F[R])$
```

$$F_R = -(m_2 + m_1)\,g\,\mu_R\tanh(k_R\,v_R)$$

Geschwindigkeit in Richtung Fahrzeuglaengsachse.

```
(%i72) vR:v[HECK][1,1]$
       disp(vR)$
```

$$\left(\frac{d}{dt}q_1\right)\cos(\psi - q_2)$$

Reibung darf nicht negativ werden, wenn das Fahrzeug verkehrt steht. In diesem Fall dreht sich die Rollrichtung ebenfalls um. Deshalb muss hier der Betrag verwendet werden.

```
(%i74) vR:diff(q[1],t)*abs(cos(q[2]-%psi))$
       disp(vR)$
```

$$\left(\frac{d}{dt}q_1\right)\,|\cos(\psi - q_2)|$$

Dissipationsfunktion

```
(%i76)  P[R]:-integrate(F[R],v[R])$
        display(P[R])$

        P[R]:subst(vR,v[R],P[R])$
        v[R]:vR$
        display(v[R])$
        display(P[R])$
```

$$P_R = \frac{(m_2 + m_1)\ g\ \mu_R \log\left(\cosh\left(k_R\ v_R\right)\right)}{k_R}$$

$$v_R = \left(\frac{d}{d\,t}\,q_1\right)\ \left|\cos\left(\psi - q_2\right)\right|$$

$$P_R = \frac{(m_2 + m_1)\ g\ \mu_R \log\left(\cosh\left(\left(\frac{d}{d\,t}\,q_1\right)\ \left|\cos\left(\psi - q_2\right)\right|\ k_R\right)\right)}{k_R}$$

```
(%i82)  print("Kontrolle ")$
        for i_n:1 thru n do
           block(
         print(" "),
             QR[i_n]:gcfac(trigreduce(-diff(P[R],diff(q[i_n],t)))),
             display(QR[i_n])
        )$
```

Kontrolle

$$QR_1 = -\left(m_2 + m_1\right)\ \left|\cos\left(\psi - q_2\right)\right|\ g\ \mu_R \tanh\left(\left(\frac{d}{d\,t}\,q_1\right)\ \left|\cos\left(\psi - q_2\right)\right|\ k_R\right)$$

$$QR_2 = 0$$

8.6 Drehreibung Vorderachse

Ansatz fuer Reibkraft gegen Fahrzeugdrehung.

```
(%i84)  %tau[D]:gcfac(-m[1]/2*g*mu[R]*b[car]*tanh(k[R]*diff(q[2],t)))$
        display(%tau[D])$
```

$$\tau_D = -\frac{m_1\ b_{car}\ g\ \mu_R \tanh\left(\left(\frac{d}{d\,t}\,q_2\right)\ k_R\right)}{2}$$

Dissipationsfunktion

```
(%i86)  P[D]:-integrate(%tau[D],diff(q[2],t))$
        display(P[D])$
```

$$P_D = \frac{m_1\, b_{car}\, g\, \mu_R \log\left(\cosh\left(\left(\frac{d}{dt}\, q_2\right)\, k_R\right)\right)}{2\, k_R}$$

```
(%i88)  print("Kontrolle ")$
        for i_n:1 thru n do
           block(
             print(" "),
             QD[i_n]:gcfac(trigreduce(-diff(P[D],diff(q[i_n],t)))),
             display(QD[i_n])
        )$
```

Kontrolle

$$QD_1 = 0 \qquad QD_2 = -\frac{m_1\, b_{car}\, g\, \mu_R \tanh\left(\left(\frac{d}{dt}\, q_2\right)\, k_R\right)}{2}$$

8.7 Dissipationsfunktion gesamt

Addition der Dissipationsfunktionen

```
(%i90)  P:P[Gl]+P[R]+P[D]$
        display(P)$
```

$$P = \frac{(m_2 + m_1)\; g\, \mu_R \log\left(\cosh\left(\left(\frac{d}{dt}\, q_1\right)\, |\cos(\psi - q_2)|\, k_R\right)\right)}{k_R} + \frac{m_1\, b_{car}\, g\, \mu_R \log\left(\cosh\left(\left(\frac{d}{dt}\, q_2\right)\, k_R\right)\right)}{2\, k_R} + \frac{m_2\, g\, \mu_{Gl} \log\left(\cosh\left(\left(\left(\frac{d}{dt}\, q_1\right)\, \sin(\psi - q_2) - \left(\frac{d}{dt}\, q_2\right)\, l_{car}\right)\, k_{Gl}\right)\right)}{k_{Gl}}$$

9 Lagrange-Funktion

```
(%i92) L:ratsimp(trigreduce(trigsimp(T-V)))$
       L:gcfac(L)$
       display(L)$
```

$$L = -\frac{1}{2}\left(\left(\psi^2 + q_2\ (q_2 - 2\,\psi)\right)\ k_F - m_2\left(\frac{d}{d\,t}\,q_2\right)^2 l_{car}^2 + \left(\frac{d}{d\,t}\,q_1\right)\left(2\,m_2\left(\frac{d}{d\,t}\,q_2\right)\sin\left(\psi - q_2\right)\ l_{car} - \left(m_2 + m_1\right)\left(\frac{d}{d\,t}\,q_1\right)\right)\right)$$

10 Lagrange-Gleichungen 2.Art

```
(%i95) for i_n:1 thru n do
          block(
            print(""),
            print("Gleichung ",i_n,":"),
            Gl[i_n]:expand(diff(diff(L,diff(q[i_n],t)),t)-diff(L,q[i_n])=
            Q[i_n]-diff(P,diff(q[i_n],t))),
            display(Gl[i_n])
       )$
```

*Gleichung*1 :

$$\psi\left(\frac{d}{d\,q_1}\,\psi\right)\ k_F - q_2\left(\frac{d}{d\,q_1}\,\psi\right)\ k_F - m_2\left(\frac{d^2}{d\,t^2}\,q_2\right)\sin\left(\psi - q_2\right)\ l_{car} + m_2\left(\frac{d}{d\,t}\,q_2\right)^2 \cos\left(\psi - q_2\right)\ l_{car} + m_2\left(\frac{d^2}{d\,t^2}\,q_1\right) + m_1\left(\frac{d^2}{d\,t^2}\,q_1\right) = -\frac{m_2\ \left|\cos\left(\psi - q_2\right)\right|\ g\,\mu_R\sinh\left(\left(\frac{d}{d\,t}\,q_1\right)\ \left|\cos\left(\psi - q_2\right)\right|\ k_R\right)}{\cosh\left(\left(\frac{d}{d\,t}\,q_1\right)\ \left|\cos\left(\psi - q_2\right)\right|\ k_R\right)} - \frac{m_1\ \left|\cos\left(\psi - q_2\right)\right|\ g\,\mu_R\sinh\left(\left(\frac{d}{d\,t}\,q_1\right)\ \left|\cos\left(\psi - q_2\right)\right|\ k_R\right)}{\cosh\left(\left(\frac{d}{d\,t}\,q_1\right)\ \left|\cos\left(\psi - q_2\right)\right|\ k_R\right)} + \sin\left(q_2\right)\left(\frac{d}{d\,q_1}\,y\right)\ F_A + \cos\left(q_2\right)\left(\frac{d}{d\,q_1}\,x\right)\ F_A +$$

$$\frac{m_2 \sin(\psi - q_2)\ g\ \mu_{Gl} \sinh\left(\left(\frac{d}{d\,t}\,q_2\right)\ l_{car}\,k_{Gl} - \left(\frac{d}{d\,t}\,q_1\right)\ \sin(\psi - q_2)\ k_{Gl}\right)}{\cosh\left(\left(\frac{d}{d\,t}\,q_2\right)\ l_{car}\,k_{Gl} - \left(\frac{d}{d\,t}\,q_1\right)\ \sin(\psi - q_2)\ k_{Gl}\right)}$$

*Gleichung*2 :

$$-\psi\,k_F + q_2\,k_F + m_2\left(\frac{d^2}{d\,t^2}\,q_2\right)\ l_{car}^2 - m_2\left(\frac{d^2}{d\,t^2}\,q_1\right)\ \sin(\psi - q_2)\ l_{car} - m_2\left(\frac{d}{d\,t}\,q_1\right)^2\left(\frac{d}{d\,q_1}\,\psi\right)\cos(\psi - q_2)\ l_{car} = -\frac{m_1\,b_{car}\,g\,\mu_R \sinh\left(\left(\frac{d}{d\,t}\,q_2\right)\ k_R\right)}{2\cosh\left(\left(\frac{d}{d\,t}\,q_2\right)\ k_R\right)} - \frac{m_2\,l_{car}\,g\,\mu_{Gl} \sinh\left(\left(\frac{d}{d\,t}\,q_2\right)\ l_{car}\,k_{Gl} - \left(\frac{d}{d\,t}\,q_1\right)\ \sin(\psi - q_2)\ k_{Gl}\right)}{\cosh\left(\left(\frac{d}{d\,t}\,q_2\right)\ l_{car}\,k_{Gl} - \left(\frac{d}{d\,t}\,q_1\right)\ \sin(\psi - q_2)\ k_{Gl}\right)}$$

11 Massenmatrix M

```
(%i96) M:trigsimp(trigreduce(jacobian([lhs(Gl[1]),lhs(Gl[2])],
       [diff(q[1],t,2),diff(q[2],t,2)])),ratsimp, collect$
       display(M)$
```

$$M = \begin{pmatrix} m_2 + m_1 & -m_2 \sin(\psi - q_2)\ l_{car} \\ -m_2 \sin(\psi - q_2)\ l_{car} & m_2\,l_{car}^2 \end{pmatrix}$$

Determinante der Massenmatrix:

```
(%i98) DetM:determinant(M), expand$
       DetM:gcfac(DetM)$
       disp(DetM)$
```

$$-m_2\left(m_2 \sin(\psi - q_2)^2 - (m_2 + m_1)\right)\ l_{car}^2$$

12 Rechte Seite

```
(%i101) rs:matrix([lhs(Gl[1])],[lhs(Gl[2])])-
        M.matrix([diff(q[1],t,2)],[diff(q[2],t,2)]),trigsimp$
```

```
(%i102) RS:trigreduce(matrix([rhs(Gl[1])],[rhs(Gl[2])])-rs),trigsimp$
        RS:gcfac(RS)$
```

13 Zustandsgleichungen: System 1.Ordnung

Einfuehren der Zustandsvariablen, die die verallgemeinerten Koordinaten und deren Ableitungen ersetzen.

```
(%i104) depends(z,t);
```

(%o104) $[z(t)]$

Substitution der Ableitungen der gen. Koordinaten mit den Zustandsvariablen:

```
(%i105) rQZ(glg):=block(
          glg:subst(z[2],diff(q[1],t),glg),
          glg:subst(z[4],diff(q[2],t),glg),
          glg:subst(z[1],q[1],glg),
          glg:subst(z[3],q[2],glg),
          gcfac((glg))
        )$

        RSz:rQZ(RS)$

        Mz:rQZ(M)$
```

Zustandsgleichungen enthalten noch die Ableitung der globalen Bahnkoordinaten (x,y) nach der Bogenlaenge l=q[1]=z[1].

```
(%i108) DGLRS:invert(Mz).RSz$
        DGLRS:gcfac(DGLRS)$
```

```
(%i110) depends(%psi,[z[1]])$
        depends(%rho,[z[1]])$
        depends(x,[z[1]])$
        depends(y,[z[1]])$
        dependencies;
```

(%o114) $[q(t), z(t), \psi(z_1), \rho(z_1), x(z_1), y(z_1)]$

Ersetzungen aufgrund des Zusammenhangs zwischen Ortskoordinaten und der Bogenlaenge (erste v. Koord.).

```
(%i115) diff(x,z[1])=cos(%psi);
        diff(y,z[1])=sin(%psi);
        diff(x,z[1],2)=(-sin(%psi))*%rho;
        diff(y,z[1],2)=(cos(%psi))*%rho;
        diff(%psi,z[1],1)=%rho;
        diff(%psi,z[1],2)=diff(%rho,z[1]);

        rCurve(glg):=block(
         dxdl:diff(x,z[1])=cos(%psi),
           dydl:diff(y,z[1])=sin(%psi),
           ddxddl:diff(x,z[1],2)=(-sin(%psi))*%rho,
           ddyddl:diff(y,z[1],2)=(cos(%psi))*%rho,
           dpsidl:diff(%psi,z[1],1)=%rho,
           ddpsiddl:diff(%psi,z[1],2)=diff(%rho,z[1]),
           glg:subst(rhs(dxdl),lhs(dxdl),glg),
           glg:subst(rhs(dydl),lhs(dydl),glg),
           glg:subst(rhs(ddxddl),lhs(ddxddl),glg),
           glg:subst(rhs(ddyddl),lhs(ddyddl),glg),
           glg:subst(rhs(dpsidl),lhs(dpsidl),glg),
           glg:subst(rhs(ddpsiddl),lhs(ddpsiddl),glg),
        gcfac((glg))
        )$
```

(%o115) $\frac{d}{d\,z_1}\,x = \cos(\psi)$

(%o116) $\frac{d}{d\,z_1}\,y = \sin(\psi)$

(%o117) $\frac{d^2}{d\,z_1^2}\,x = -\sin(\psi)\ \rho$

(%o118) $\frac{d^2}{d\,z_1^2}\,y = \cos(\psi)\ \rho$

(%o119) $\frac{d}{d\,z_1}\,\psi = \rho$

(%o120) $\frac{d^2}{d\,z_1^2}\,\psi = \frac{d}{d\,z_1}\,\rho$

DGL

```
(%i122) DGL[1]:diff(z[1],t)=z[2]$
        DGL[2]:diff(z[2],t)=rCurve(DGLRS[1,1])$
        DGL[3]:diff(z[3],t)=z[4]$
        DGL[4]:diff(z[4],t)=rCurve(DGLRS[2,1])$

        print("******************************")$
        display(DGL[1])$ print("******************************")$
        display(DGL[2])$ print("******************************")$
        display(DGL[3])$ print("******************************")$
        display(DGL[4])$ print("******************************")$
```

$$**********$$
$$\frac{d}{d\,t}\,z_1 = z_2$$
$$**********$$
$$\frac{d}{d\,t}\,z_2 = (sin(\psi - z(3)) * (-g * m_1 * b_{CAR} * tanh(z(4) * k_R) * \mu_R/2 + m_2 * l_{CAR} * (z(2)^2 * cos(\psi - z(3)) * \rho - g * tanh(k_{GL} * (z(4) * l_{CAR} - z(2) * sin(\psi - z(3)))) * \mu_{GL}) + (\psi - z(3)) * k_F)/((1 - m_2 * sin(\psi - z(3))^2/m) * l_{CAR}) - (m_2 * sin(\psi - z(3))^2/(m * (1 - m_2 * sin(\psi - z(3))^2/m)) + 1) * (g * m * abs(cos(\psi - z(3))) * tanh(z(2) * abs(cos(\psi - z(3))) * k_R) * \mu_R + m_2 * (z(4)^2 * cos(\psi - z(3)) * l_{CAR} - g * sin(\psi - z(3)) * tanh(k_{GL} * (z(4) * l_{CAR} - z(2) * sin(\psi - z(3)))) * \mu_{GL}) + \psi * \rho * k_F - z(3) * \rho * k_F - (sin(z(3)) *$$

$sin(\psi) + cos(z(3)) * cos(\psi)) * F_A))/m$
$**********$
$\frac{d}{dt} z_3 = z_4$
$**********$
$\frac{d}{dt} z_4 = -(m_2 * sin(\psi - z(3)) * l_{CAR} * (g * m * abs(cos(\psi - z(3))) * tanh(z(2) * abs(cos(\psi - z(3))) * k_R) * \mu_R + m_2 * (z(4)^2 * cos(\psi - z(3)) * l_{CAR} - g * sin(\psi - z(3)) * tanh(k_{GL} * (z(4) * l_{CAR} - z(2) * sin(\psi - z(3)))) * \mu_{GL}) + \psi * \rho * k_F - z(3) * \rho * k_F - (sin(z(3)) * sin(\psi) + cos(z(3)) * cos(\psi)) * F_A)/m + g * m_1 * b_{CAR} * tanh(z(4) * k_R) * \mu_R/2 - m_2 * l_{CAR} * (z(2)^2 * cos(\psi - z(3)) * \rho - g * tanh(k_{GL} * (z(4) * l_{CAR} - z(2) * sin(\psi - z(3)))) * \mu_{GL}) - (\psi - z(3)) * k_F)/(m_2 * (1 - m_2 * sin(\psi - z(3))^2/m) * l_{CAR}^2)$
$**********$

4.4.14 Motordaten

Um direkt eine Antriebskraft vorgeben zu können, müssen zuerst noch die Daten der Motorgleichung bestimmt werden. Als Steuervorgabe wird eine Bit-Zahl von $0 - 255$ an den Controller gesendet, der anhand der Größe dieser Zahl ein entsprechendes Pulsweitenmoduliertes Signal an den Motor abgibt. Tatsächlich werden nur Werte bis 135 *Bit* umgesetzt, da ab diesem Wert bereits die maximale Pulsweite verwendet wird. Das Ersatzschaltbild des Elektromotors ist in Abbildung 4.5 dargestellt. Für die Bestimmung des Widerstandes soll nun der Motor festgehalten werden, da somit die Drehzahl gleich Null ist und die induzierte Spannung ebenso Null ist. Durch den geringen Anteil der Induktivität bleiben somit nur die Ankerspannung U_A und der Spannungsverlust am Widerstand U_R über. Mit einer Messung dieser Werte kann der Widerstand R wie in Tabelle 4.1 berechnet werden.

Tabelle 4.1: Messung zur Bestimmung des Ankerwiderstandes

Versuch	U_A	I_A	R (berechnet)
	V	A	Ω
1	1	0.125	8
2	2	0.250	8
3	3	0.400	7.5

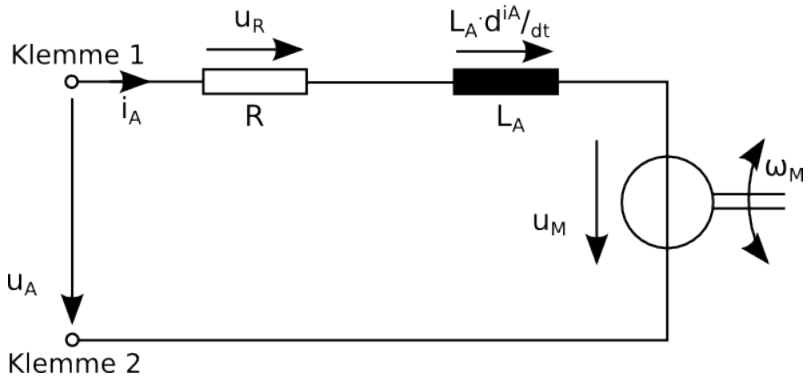


Abbildung 4.5: Ersatzschaltbild des Elektromotors

Anhand dieser Messung wird der Widerstand mit 8 Ohm festgelegt. Als nächstes ist die Motorkonstante k_M zu bestimmen. Die vereinfachte Motorgleichung lautet:

$$\begin{aligned} u_A &= R\, i_A + u_M \\ u_A &= R\, i_A + k_M\, \omega_M \end{aligned} \tag{4.46}$$

Für die Bestimmung der Motorkonstante wurden stationäre Testversuche mit einer Bitvorgabe durchgeführt, dafür musste das Slot-Car vom Boden gehoben werden, damit die Räder sich ohne Reibungslast drehen konnten. Dabei wurde mit einem Oszilloskop die Zeitspanne gemessen, bei der das Spannungssignal der Pulsweitenmodulation auf *HIGH* steht, was 18.4V

Tabelle 4.2: Messung der Spannung u_M über die Zeitspanne T_M, des Stroms I_M und der Periodendauer T_R

Bit	T_M	u_M	i_M	T_R
Bit	μs	V	mA	μs
20	16	4.600	115.6	1800
40	24	6.9	145.3	970
60	35	10.063	139.9	674
80	45	12.937	157.0	540
100	52	14.950	169.8	437
120	60	17.250	181.7	377
135	64	17,400	181.2	355

entspricht. Mit dem Wert von $64\mu s$ der gesamten Periodendauer T_{Mges} kann daraus der Wert der angelegten Spannung ermittelt werden:

$$u_M = \frac{T_M \; 18.4V}{T_{Mges}} \tag{4.47}$$

Mit einer zusätzlichen Strommessung und einer Lichtschrankenmessung des Zahnrades der Hinterachse sind alle Messwerte verfügbar, die für die Berechnung der weiteren Parameter benötigt werden. Die Ergebnisse der Messung sind in Tabelle 4.2 dargestellt.
Die Motorkonstante k_M kann über folgende Gleichung berechnet werden, da es die einzig verbleibende Unbekannte ist:

$$U_A = R * i_M + k_M * \omega_M \qquad \rightarrow \qquad k_M = \frac{U_A - R * i_M}{\omega_M} \tag{4.48}$$

Die noch benötigten Parameter können mit den Werten aus Tabelle 4.2

berechnet werden:

$$\begin{aligned} \omega_R &= \frac{2\,\pi\,f_R}{50} = \frac{2\,\pi}{T_R\,50} \\ \omega_M &= 5\,\omega_R \\ n_M &= \frac{1}{T_R}\frac{60\,5}{50} \end{aligned} \tag{4.49}$$

Das Zahnrad der Hinterachse besitzt 50 Zähne, da aber eine Umdrehung des Hinterrads verwendet werden soll, die Lichtschranke aber alle Zähne zählt, muss der errechnete Wert durch 50 dividiert werden, um auf die richtige Winkelgeschwindigkeit des Rades zu kommen. Mit der Untersetzung von 5 der Hinterachse auf den Motor kann die Winkelgeschwindikeit des Motors berechnet werden. Bei der Berechnung der Drehzahl kommt noch der Wert 60 für die Umrechnung von der Winkelgeschwindikeit in $\frac{rad}{s}$ auf die Drehzahl in U/min hinzu.
Mit den Werten aus Tabelle 4.2 und Gleichung 4.49 kann nun der Wert für k_M mit 0.0095 angegeben werden. Für die Umrechnung der vorgegebenen Bit-Zahl zur Motorspannung wird der Einfachheit wegen eine Konstante mit dem Wert 0.15 angenommen.
Laut Angabe des Carrera-Datenblattes kann der verwendete Elektromotor eine Drehzahl von 18000 U/min erreichen. Eine solch hohe Drehzahl ist bei den Fahrzeugen nicht verwendbar, daher ist die (Getriebe-)Übersetzung von Motor auf die Hinterachse als Untersetzung mit 10 Zähnen auf Motorseite und 50 Zähnen auf Achsseite ausgelegt, was eine Untersetzung von

$$i_G = \frac{z_{Abtrieb}}{z_{Antrieb}} = \frac{n_{Antrieb}}{n_{Abtrieb}} = \frac{50}{10} = 10 \tag{4.50}$$

ergibt. Mit der Umrechnung der maximalen Motordrehzahl von 18000 U/min auf die Geschwindigkeit in m/s mit

$$v = \frac{2 * r * \pi * n_M * 60}{i * 1000} = \frac{2 * 0,015 * \pi * 18000 * 60}{5 * 1000} = 5,655\ m/s = 20,36\ km/h$$

kann die theoretische Maximalgeschwindigkeit errechnet werden. Durch die auftretenden Reibungen und Ungenauigkeiten wird diese in der Praxis aber nie erreicht, dennoch kann daraus abgelesen werden, dass die Slot-Cars für die Streckengrößen eine stattliche Geschwindigkeit erreichen können.
Die benötigte Antriebskraft kann nun über das Leistungsgleichgewicht ermittelt werden:

$$\begin{aligned} U_A * i_M &= M_R * \omega_R \\ k_M * \omega_M * i_M &= M_R * \omega_R \\ k_M * \omega_R * i_G * i_M &= M_R * \omega_R \\ k_M * i_G * i_M = M_R = F_A * r_R \qquad &\rightarrow \qquad F_A = \frac{k_M * i_M}{r_R} \end{aligned} \tag{4.51}$$

Die angegebenen Indizes M und R stehen für die Zuordnung als *Motor* und *Rad*. Mit den Werten für den Motorstrom

$$\begin{aligned} i_M &= \frac{u_M - k_M * \omega_M}{R_M} \qquad \text{für } u_M > 0\ V \\ i_M &= \frac{0 - k_M * \omega_M}{R_M + R_B} \qquad \text{für } u_M = 0\ V \end{aligned}$$

ist die Umrechnung von der vorgegebenen Bit-Zahl zu der Antriebskraft komplett. Die Unterscheidung des Motorstroms i_M ist notwendig, da die Carrera-Fahrzeuge einen Bremswiderstand R_B integriert haben, der variabel in fünfzehn Stufen einstellbar ist und das Bremsverhalten deutlich beeinflusst. Wird keine Steuerung vorgegeben, wird eine Spannung im Motor induziert und der Stromfluss umgedreht, woraus sich ein negatives Moment und somit eine negative Kraft ergibt.
Die Winkelgeschwindigkeit ω_M wird über die Beziehung

$$\omega_M = \omega_R * i_G$$

ermittelt, wobei ω_R zum jeweiligen Zeitpunkt aus der Berechnung der Differentialgleichung für die Geschwindigkeit des Fahrzeugs bezogen wird:

$$\omega_R = x(2) * r_R$$

Das Modell wird der Einfachheit halber als schlupffrei betrachtet.

4.5 Implementierung in Matlab

In Matlab wurden die resultierenden Zustandsgleichungen 1. Ordnung wie in Abbildung 4.6 dargestellt in der Funktion *f_absq2* implementiert. Im Programm Maxima mit dem grafischen Aufsatz wxMaxima können die Gleichungen direkt als Matlab-Code exportiert werden. Dazu wurde ein eigenes Makro mit dem Namen *rMATLAB* geschrieben.

Simulink

Das Simulink-Modell ist einfach aufgebaut und besteht aus wenigen Bauteilen, wie in Abbildung 4.6 zu sehen ist. Der Signalgeber-Block ist für die Simulation der vorgegebenen Steuerung vorgesehen, damit sich das Auto bewegt. Bei Verwendung der optimalen Steuerung ist dieser Teil nicht mehr nötig, da die Antriebskraft über die optimale Steuerung vorgegeben wird. Der *Integrator* wird im Zusammenhang mit der *Matlab-Function* verwendet. In dieser Matlab-Funktion *f_ absq2* erfolgt die Berechnung der nötigen Daten aus der vorgegebenen Bit-Folge und der Zustandsgrößen der vorherigen Integration, wie im nachfolgenden Source-Code aus der Matlab-Funktion gezeigt wird:

```
function dx = f_absq2( v )

x = v(1:4);          % Zustandsgrößen
Bit = v(5);          % Steuergröße

% Parameter:
```

```
g = 9.81;            % Erdbeschleunigung
m_1 = 0.1;           % Masse Vorderachse
m_2 = 0.12;          % Masse Hinterachse
m = 0.22;            % Masse Gesamt
l_CAR = 0.098;       % Länge Vorder- Hinterachse
b_CAR = 0.067;       % Breite von Mitte Reifen
r_CAR = 0.015;       % Radradius
i_G = 5;             % Untersetzung Motor - Achse
R_M = 8;             % Widerstand Ankerkreis
k_M = 0.0095;        % Motorkonstante
R_B = 150;           % Bremswiderstand
k_F = 0.01;          % Federkonstante
k_GL = 0.1;          % Konstante Gleitreibung (tanh)
k_R = 0.1;           % Konstante Rollreibung (tanh)
mu_GL = 30.0;        % Koeffizient Gleitreibung
mu_R = 3.1;          % Koeffizient Rollreibung

% Beschreibung Motorgleichung - Antriebskraft:
u_M = 0.15*Bit;
omega_R = x(2)/r_CAR;
omega_M = omega_R * i_G;
if Bit > 0
   i_M = u_M - k_M * omega_M/(R_M);
   M_M = k_M * i_M;
   F_A = i_G * M_M/r_CAR;
else
   i_M = -k_M * omega_M/(R_M + R_B);
   M_M = k_M * i_M;
   F_A = i_G * M_M/r_CAR;
end
```

% Werte aus der Spline-Klasse:
ψ = get_Psi($x(1)$);
ρ = get_Rho($x(1)$);

$dx(1) = x(2)$;
$dx(2) = (sin(\psi - x(3)) * (-g * m_1 * b_{CAR} * tanh(x(4) * k_R) * \mu_R/2 + m_2 * l_{CAR} * (x(2)^2 * cos(\psi - x(3)) * \rho - g * tanh(k_{GL} * (x(4) * l_{CAR} - x(2) * sin(\psi - x(3)))) * \mu_{GL}) + (\psi - x(3)) * k_F)/((1 - m_2 * sin(\psi - x(3))^2/m) * l_{CAR}) - (m_2 * sin(\psi - x(3))^2/(m * (1 - m_2 * sin(\psi - x(3))^2/m)) + 1) * (g * m * abs(cos(\psi - x(3))) * tanh(x(2) * abs(cos(\psi - x(3))) * k_R) * \mu_R + m_2 * (x(4)^2 * cos(\psi - x(3)) * l_{CAR} - g * sin(\psi - x(3)) * tanh(k_{GL} * (x(4) * l_{CAR} - x(2) * sin(\psi - x(3)))) * \mu_{GL}) + \psi * \rho * k_F - x(3) * \rho * k_F - (sin(x(3)) * sin(\psi) + cos(x(3)) * cos(\psi)) * F_A))/m$;
$dx(3) = x(4)$;
$dx(4) = -(m_2 * sin(\psi - x(3)) * l_{CAR} * (g * m * abs(cos(\psi - x(3))) * tanh(x(2) * abs(cos(\psi - x(3))) * k_R) * \mu_R + m_2 * (x(4)^2 * cos(\psi - x(3)) * l_{CAR} - g * sin(\psi - x(3)) * tanh(k_{GL} * (x(4) * l_{CAR} - x(2) * sin(\psi - x(3)))) * \mu_{GL}) + \psi * \rho * k_F - x(3) * \rho * k_F - (sin(x(3)) * sin(\psi) + cos(x(3)) * cos(\psi)) * F_A)/m + g * m_1 * b_{CAR} * tanh(x(4) * k_R) * \mu_R/2 - m_2 * l_{CAR} * (x(2)^2 * cos(\psi - x(3)) * \rho - g * tanh(k_{GL} * (x(4) * l_{CAR} - x(2) * sin(\psi - x(3)))) * \mu_{GL}) - (\psi - x(3)) * k_F)/(m_2 * (1 - m_2 * sin(\psi - x(3))^2/m) * l_{CAR}^2)$;

end

Für die nötigen Startwerte der Simulation wird ein *Constant*-Block mit vier Einträgen verwendet, da jede der vier Differtialgleichungen einen eigenen Startwert benötigt. Die Ergebnisse werden einerseits für die direkte Überprüfung mit einem Scope-Block dargestellt, andererseits werden über die Funktion *myplot_absq2* die Daten ausgewertet und können in einer Grafik dargestellt werden.

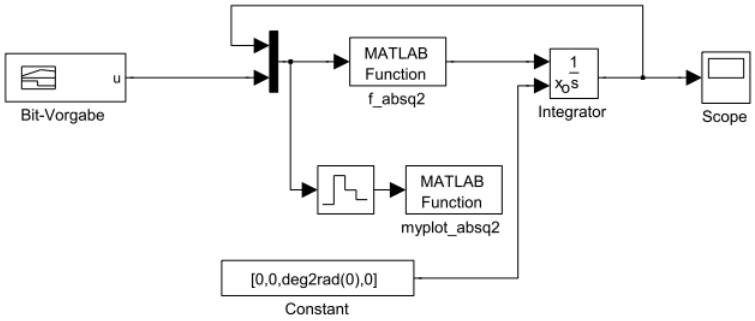


Abbildung 4.6: Simulink-Modell des Bewegungsmodells

5 Modellparameter

Als letzter Schritt vor der Berechnung einer zeitoptimalen Steuerung werden die konkreten Zahlenwerte aller Parameter des Slot-Cars anhand von teils einfachen Messungen, aber auch durch Identifikation mit Hilfe von Testergebnissen ermittelt. Zu bestimmen sind die Größen für:

- die Masse m des Autos,
- der Schwerpunkt SP, über dessen Lage anschließend die Massen m_1 und m_2 ermittelt werden,
- die Länge l_{car} des Fahrzeugs zwischen den Achsen,
- die Breite b_{car} des Autos,
- die Federkonstante k_F,
- die Koeffizienten k_R und k_{Gl} für die Skalierung der *tanh*-Funktion,
- der Rollreibungskoeffizient μ_R,
- der Gleitreibungskoeffizient μ_{Gl}.

5.1 Ermittlung der Parameterwerte

Jeder dieser Parameter wird mit einem Test oder einer geeigneten Messung ermittelt. Bei der Ermittlung der Koeffizienten für Roll- bzw. Gleitreibung ist die Reihenfolge entscheidend, da bei einem Test auf einer Kreisbahn zur Ermittlung des Gleitreibungskoeffizienten auch die Rollreibung eine entscheidende Rolle übernimmt, jedoch bei einem Versuch auf einer Geraden zur Ermittlung der Rollreibung keine Terme der Gleitreibung einen Einfluss

haben. Der Winkel zwischen den Achsen bleibt bei gerade aufgesetztem Slot-Car immer Null.

5.1.1 Fahrzeugparameter

Einige Parameter können einfach über direkte Messungen ermittelt werden. Dazu gehört die Masse m des gesamten Fahrzeugs, die mit verschiedenen Waagen gewogen wurde, um Ungenauigkeiten zu vermeiden. Die verschiedenen Slot-Cars haben auch unterschiedliche Gewichte und Längen, so ist die Version des Audi $238g$ schwer, während die des Porsche nur $220g$ wiegt. Die Länge l_{car}, die die Entfernung der Mittelpunkte der Vorderachse zur Hinterachse des Slot-Cars beschreibt, beträgt beim Audi $111mm$ und beim Porsche nur $98mm$. Bei dem verwendeten Modell wird die Gesamtlänge nicht benötigt, da die Massen auf die zwei Punkte an der Vorderachse, sowie an der Hinterachse konzentriert sind. Der Schwerpunkt ist beim Audi bei $47mm$, von der Hinterachse aus gemessen, und beim Porsche bei $44mm$. Der Radradius ist hingegen bei beiden Slot-Cars derselbe. Durch diese unterschiedlichen Daten verhalten sich die Slot-Cars bei der Steuerung auch merklich anders.

Der Schwerpunkt wurde ermittelt, indem das Slot-Car auf einer Schnur, die quer zur Fahrzeuglängsachse unter das Fahrzeug angebracht war, in Gleichgewichtslage gebracht wurde. Der Punkt dieser Lage wurde markiert und der Abstand der beiden Achsen zu diesem Punkt gemessen. Nach Ermittlung der Lage des Schwerpunktes können über folgende Längenverhältnisse die Massen m_1 und m_2 ermittelt werden:

$$\begin{aligned} m1 &= \frac{l_{carH}}{l_{car}} m, \\ m2 &= \frac{l_{carV}}{l_{car}} m, \end{aligned} \tag{5.1}$$

wobei l_{carV} und l_{carH} jeweils die Entfernungen vom Schwerpunkt in Richtung der jeweiligen Achse sind. Beim Audi ist die Verteilung der Massen mit $m_1 = 101g$ und $m_2 = 137g$ im Vergleich zum Porsche mit $m_1 = 99g$ und $m_2 = 121g$ der Schwerpunkt weiter an der Hinterachse.

5.1.2 Testversuche zur Ermittlung der Reibungskoeffizienten

Die Ermittlung des Rollreibungskoeffizienten erfolgt mit einer vereinfachten Version der Zustandsgleichungen. Da bei einer Geradeausfahrt weder eine Winkeländerung, noch eine daraus resultierende Winkelbeschleunigung auftritt, kann das Modell auf zwei Zustandsgleichungen für die Entfernung und Geschwindigkeit vereinfacht werden. Die Gleichung der Geschwindigkeit kann auf ein Minimum reduziert werden, da die Winkelfunktionen für einen Winkel mit 0° entweder den Wert 0 oder 1 als Betrag ergeben. Nach Eliminieren aller nicht benötigter Terme bleibt nur noch folgende Gleichung übrig:

$$\dot{v} = F_A - m * g * \mu_R * tanh(v * k_{Gl}) \tag{5.2}$$

Für diesen Versuch wurde ein Testsystem als gerade Strecke mit 11 Sektionen aufgebaut, was mit einer Sektionslänge von $0.345m$ einer Länge von $3.795m$ entspricht. Das Auto wurde für jeden Test vor dem Beginn der Zeitmessung auf eine konstante Geschwindigkeit beschleunigt. Mit den gemessenen Daten kann auch die Geschwindigkeit ermittelt werden. In Tabelle 5.1 sind die Werte des Tests der Geradeausfahrt dargestellt.

Tabelle 5.1: Tabelle mit Werten der Geradeausfahrt vom Carrera Auto

gemessene Werte		berechnete Werte	
Bit	Zeitintervall	Zeit	Geschwindigkeit
	s	s	$\frac{m}{s}$
40	3.45 - 6.58	3.13	1.21
50	3.60 - 6.02	2.42	1.57
55	4.95 - 7.15	2.20	1.72
60	3.30 - 5.30	2.00	1.90
65	3.65 - 5.57	1.92	1.98
70	3.00 - 4.80	1.80	2.11

Für die Bestimmung des Rollreibungskoeffizienten wurde noch ein weiterer Testversuch gemacht, bei dem das Ausroll- bzw. das Bremsverhalten untersucht wurde. Der Versuch ist wurde ebenfalls auf einer Geraden durchgeführt, jedoch wird hierbei das Fahrzeug zuerst wiederum auf eine konstante Geschwindigkeit beschleunigt und ab einem bestimmten Punkt wird die Spannungsversorgung auf $0V$ abgestellt. Für diesen Punkt wird das Startsegment verwendet, da auf diesem erkannt werden kann, dass das Slot-Car eine neue Runde beginnt und somit die Spannungsversorgung auf $0V$ geändert werden kann. Nach dem Ausrollen wird die Distanz der Position des Fahrzeuges zur Position ermittelt, wo das Ausrollen begann. Für eine exakte Ermittlung der Daten wurde die Messung, wie in Tabelle 5.2 dargestellt, viermal wiederholt. Anhand der Werte dieser Tabelle ist auch schon das Problem festzustellen, das es bei der Ermittlung der Parameter der Reibungskoeffizienten gibt. Die Werte bei denselben Bit-Vorgaben weichen sehr stark voneinander ab, deshalb können die Koeffizienten nicht exakt bestimmt werden. Aus diesem Grund wurden die Werte der Rollreibung, wie auch die der Gleitreibung, empirisch angepasst.
Mit den gemessenen Werten wurde der Wert für den Skalierungsfaktor

Tabelle 5.2: Tabelle mit Werten des Ausrollversuchs

Bit		20	40	60	80	100	120
Distanz	cm	11.5	37	64	87	147	194
Distanz	cm	15.5	37	78	91	103	194
Distanz	cm	17.0	37	65	113	124	124
Distanz	cm	12.5	37	78	95	104	220

k_R mit 0.1 bestimmt und der Wert des Rollreibungskoeffizienten μ_R auf einer Geraden mit 3.5 und einer Kurve mit 6. Diese Unterscheidung ist für eine genaue Anpassung nötig, da noch andere Kräfte, die nicht spezifiziert wurden, einen Einfluss auf den Reibungskoeffizienten haben.

Für die Ermittlung des Gleitreibungsterms wurde eine weitere Versuchsreihe durchgeführt. Mit dem aus der Geradeausfahrt ermittelten Rollreibungskoeffizienten und dem noch unbekannten Koeffizienten für die Gleitreibung kann nun eine Kreisfahrt verwendet werden, um diesen letzten unbekannten Term zu ermitteln. Bei konstant vorgegebener Spannung bleibt die Drehzahl des Motors auf konstantem Niveau, wodurch ein annähernd konstant auftretender Driftwinkel die Folge ist. Die Spannungsvorgabe wurde soweit erhöht, bis das Slot-Car die Haftungsgrenze erreichte und somit auch ein Vergleich zwischen den ermittelten Werten und den real gemessenen Werten möglich wurde.
Für die Ermittlung der Gleitreibung wurde ein Kreis verwendet, damit die Kurvenfahrt mit einem annähernd konstantem Winkel gefahren werden kann und weiters auch die Grenze der Vorgabe der Stellgröße ermittelt werden kann, bei der das Auto aus der Bahn fällt. Dazu ist eine Spannungsquelle zu verwenden, die die nötigen $18V$ zur Verfügung stellt. Für die Messung wurde das Auto jeweils zuerst auf die Endgeschwindigkeit beschleunigt und anschließend über eine Entfernung von 10 Runden die Zeit ermittelt. Durch die höhere Anzahl der Runden sollte der Ablesefehler,

der durch die recht hohe Geschwindigkeit definitiv gegeben ist, minimiert werden. Somit ist die Entfernung, sowie die dafür benötigte Zeit, bekannt. In Tabelle 5.3 sind die Ergebnisse dieser Messung dargestellt.

Tabelle 5.3: Tabelle mit Werten der Kurvenfahrtmessung vom Carrera Auto

gemessene Werte		berechnete Werte		
Bit	Gesamtzeit	Rundenzeit	Winkel	Geschwindigkeit
	s	s	°	m/s
40	21.1	2.11	-16.5	1.032
50	17.4	1.74	-14.0	1.251
60	14.4	1.44	-11.5	1.512
70	12.5	1.25	-13.0	1.742
80	10.9	1.09	-10.5	1.997
85	10.0	1.00	-11.0	2.177
88	9.6	0.96	-8.5	2.268

Zusätzlich wurde über der Bahn eine Kamera montiert und damit per Serienbild das Auto mit seinem Winkel zur Bahn aufgezeichnet. Dadurch ist es möglich, die Abweichung des Autos zur Tangente per Bildbearbeitung festzustellen. Hierzu wurde in den verwendeten Bildern per Inkscape an die zwei Übergänge der Schienenelemente jeweils eine Gerade gelegt, um den Mittelpunkt des Kurvenradius zu ermitteln. Mit einer weiteren Gerade zum Mittelpunkt der vorderen Achse und einer zu dieser um 90 Grad gedrehten Gerade wird die Tangente der Bahn an der Stelle der vorderen Achse ermittelt. Wird nun noch eine Gerade zwischen Mittelpunkt der vorderen Achse und Mittelpunkt der hinteren Achse gelegt, kann daraus ein rechtwinkeliges Dreieck ermittelt werden und über die Längenverhältnisse der Driftwinkel berechnet werden. Diese Berechnung ist in Abbildung 5.1, sowie in Abbildung 5.2 zu sehen.

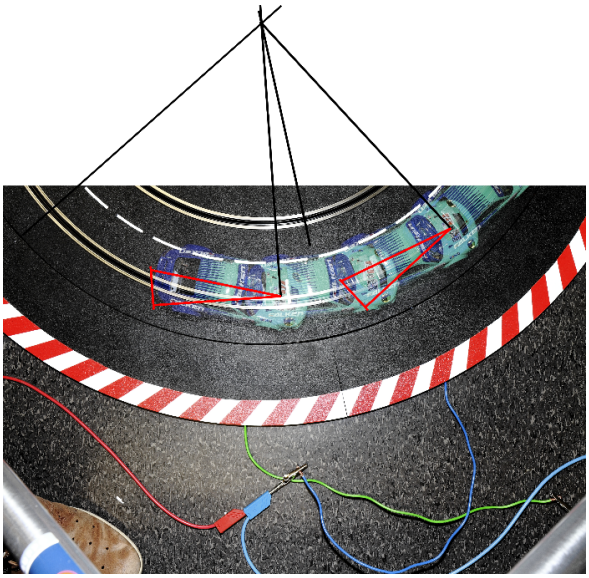


Abbildung 5.1: grafische Ermittlung des Driftwinkels

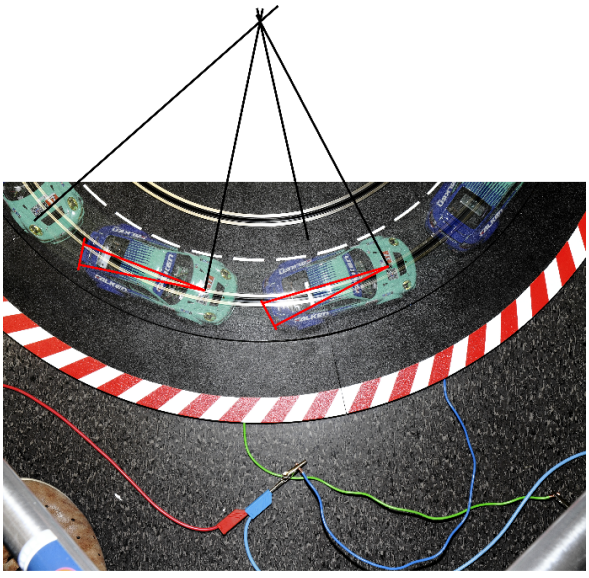


Abbildung 5.2: grafische Ermittlung des Driftwinkels

In Tabelle 5.3 sind die Werte der Messungen ersichtlich. Der Radius der Innenbahn beträgt 0.2475m und der Radius der Außenbahn 0.3465m, wodurch der Umfang, also eine Runde, auf der Innenbahn 1.555m und auf der Außenbahn 2.177m beträgt. Für die Messung wurde die Außenbahn verwendet, damit etwas höhere Geschwindigkeiten möglich werden. In den Abbildungen sind 2 Punkte gut zu erkennen:

- Auf einem Bild sind die abgebildeten Fahrzeuge weiter auseinander. Das deutet auf eine höhere Geschwindigkeit hin, da immer die selbe Auslösezeit bei der Kamera verwendet wurde.
- Die eingezeichneten Dreiecke sind in Abbildung 5.2 viel kleiner als in Abbildung 5.1. Daraus kann der Driftwinkel erkannt werden und somit auch auf die Geschwindigkeit geschlossen werden.

Mit diesen 3 ermittelten Werten kann nun die empirische Anpassung des Gleitreibungskoeffizienten durchgeführt werden. Der ermittelte Wert beträgt für den Skalierungsfaktor k_{GL} 0.1 und für den Gleitreibungskoeffizienten μ_{GL} 10.0 für den Bereich, wo der Magnet des Slot-Cars keine Wirkung mehr auf die Bahn hat, und 30.0, wo der Magnet noch einen Einfluss auf die Haftreibung der Bahn hat.

5.1.3 Vergleichsversuch Simulation/ Messung der Dynamik

Um einen Eindruck davon zu bekommen, wie gut das Modell die Realität abbildet, wird in einem Versuch mit einer vorgegebenen Stellgröße ermittelt, wie groß die Distanzabweichung des Modells mit den ermittelten Parametern und der Realität ist. Dazu wurde ein ovaler Kurs, bestehend aus 6 Geradenstücken und 6 Kurvenstücken, aufgebaut, wie er in Abbildung 5.3 veranschaulicht ist. Mit diesem Aufbau besitzt der Kurs eine Länge von $4.25m$. Die verwendete Steuerungsvorgabe ist in Abbildung 5.4 dargestellt. Das Fahrzeug wurde für 2 Sekunden mit einem Wert von 50 Bit beschleunigt, eine größere Vorgabe war bei diesem Test nicht möglich, da das Fahrzeug

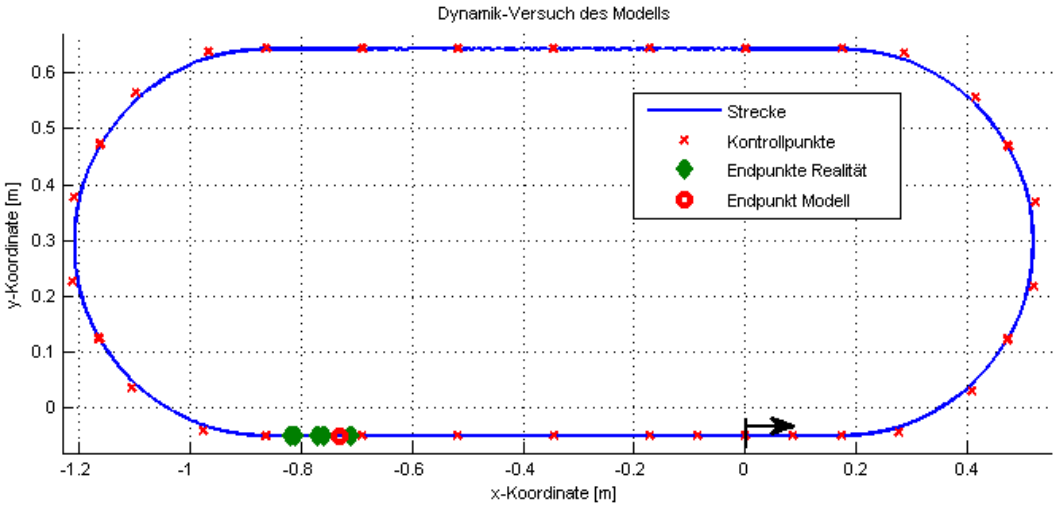


Abbildung 5.3: Teststrecke für Vergleichsversuch

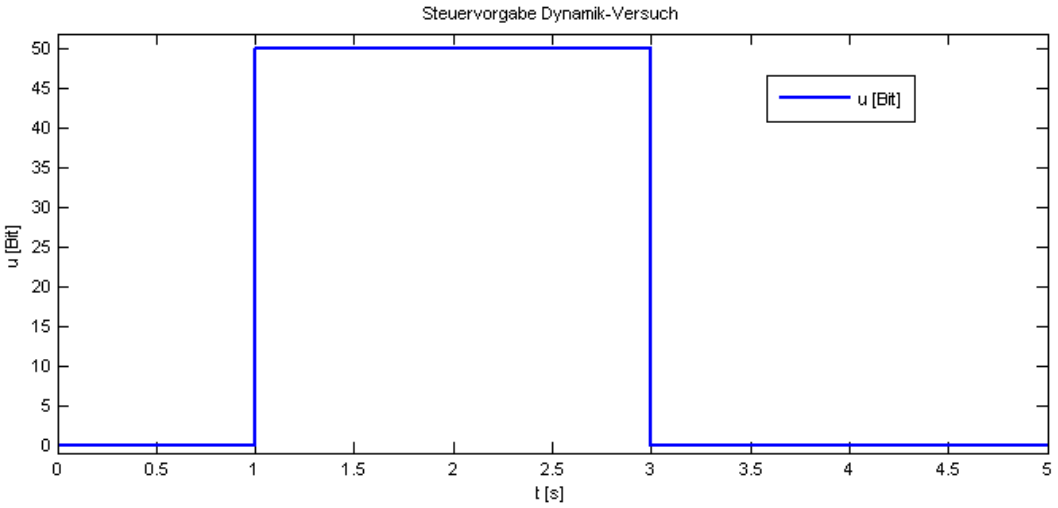


Abbildung 5.4: Steuerungsvorgabe für Vergleichsversuch

ansonsten zu schnell für die vorliegenden Kurven wäre.
Für den Versuch mit dem realen Slot-Car wurde das Steuersignal über den PC vorgegeben und das Fahrzeug von der Start-Linie weg beschleunigt. Der Versuch wurde mehrmals wiederholt, um die Abweichung bei gleicher Vorgabe zu untersuchen. In Abbildung 5.3 sind die Punkte des realen Versuchs, an welchen das Fahrzeug stehen blieb, mit grünen Diamanten gekennzeichnet. Die Endwerte weisen etwa eine Distanz von $3.43m$ bis $3.55m$ vom Startpunkt auf. Es ist ersichtlich, dass bei exakt selber Vorgabe der Steuerung eine Abweichung der Endpunkte von etwa $12cm$ vorhanden ist. Bei der Simulation des Versuchs kommt man zum Endpunkt der mit einem rot-farbigen Kreis dargestellt ist und eine Distanz von etwa $3.53m$ aufweist. Dieser liegt innerhalb der abweichenden Werte der Realität, was veranschaulicht, dass das Modell gut für eine Optimierung der Steuervorgabe verwendet werden kann. Als Algorithmus der Simulation wurde *ode*45 (*Dormand-Prince*) verwendet, die maximale Schrittweite wurde mit $0.05s$ angegeben.

6 Optimierung

Für die Optimierung wurden drei verschiedene Möglichkeiten in Betracht gezogen und die am besten geeignete wird für die vorhandene Problemstellung in 6.2 zur Berechnung der optimalen Steuertrajektorie verwendet.
Die drei Varianten sind:

- Variationsrechnung, Minimum-Prinzip
- Dynamische Programmierung
- Differential-Evolution aus der Klasse der Evolutionären Algorithmen

Diese Methoden unterscheiden sich grundlegend voneinander, das Ziel ist hingegen immer eine optimale Lösung zu finden. In den Abschnitten 6.1.1, 6.1.2 und 6.1.3 wird auf die Grundlagen der drei Methoden eingegangen und ihre Verwendbarkeit für die gegebene Problemstellung der Berechnung einer zeitoptimalen Steuerung der Slot-Cars für beliebige Strecken eruiert.

6.1 Varianten der Optimierung

6.1.1 Variationsrechnung

Es soll eine Funktion $x(t)$ im Intervall $[t_0, t_1]$ mit vorgegebenen Randwerten $x(t_0) = x_0$ und $x(t_1) = x_1$ für das Funktional

$$J = \int_{t_0}^{t_1} L(x, \dot{x}, t)dt$$

mit gegebener Funktion $L(x, \dot{x}, t)$ gesucht werden, bei der das Funktional ein Extremum wird. Für eine optimale Lösung des Problems, werden die Abweichungen der optimalen Funktion *x(t)*

$$\bar{x}(t) = x(t) + \epsilon\eta(t)$$

mit ϵ als Zahl und $\eta(t)$ als zweifach differenzierbare Funktion, betrachtet. Um zur optimalen Lösung zu gelangen, ist der Ausdruck als Funktion der Variablen ϵ aufzufassen und folgende notwendige Bedingung muss gelten:

$$F(\epsilon) := \int_{t_0}^{t_1} L(x + \epsilon\eta, \dot{x} + \epsilon\dot{\eta}, t)dt \ , \qquad \left.\frac{dF}{d\epsilon}\right|_{\epsilon=0} \stackrel{!}{=} 0$$

Die daraus gefundene, notwendige Bedingung für ein Extremum des Funktionals wird Euler Differentialgleichung genannt:

$$\frac{\partial L}{\partial x} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) = 0 \tag{6.1}$$

Die Problemstellung kann, wie in [6, Kapitel 7,8] und [10, Kapitel 9-11] beschrieben, um einen freien Rand mit Endbedingungen erweitert werden. Um dieses Prinzip auf dynamische Systeme anwenden zu können, müssen noch Nebenbedingungen berücksichtigt werden. Die mathematische Beschreibung eines dynamischen Systems als Nebenbedingung lautet in der Regelungstechnik

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \ ,$$

wobei $\mathbf{x} \in \mathbb{R}^n$ den Zustandsvektor mit dem Startwert

$$\mathbf{x}(t_0) = \mathbf{x}_0$$

darstellt und $\mathbf{u} \in \mathbb{R}^m$ den Stellgrößenvektor.
Mit diesen Voraussetzungen wird nun die optimale Steuerung **u**(t) im Intervall $[t_0, t_1]$ gesucht, damit das Gütefunktional

$$J[\mathbf{u}(t)] = S(\mathbf{x}(t_1), t_1) + \int_{t_0}^{t_1} L(\mathbf{x}, \mathbf{u}, t)dt$$

minimal wird.
Mit Einführung der **Hamilton-Funktion**

$$H(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, t) := L(\mathbf{x}, \mathbf{u}, t) + \boldsymbol{\lambda}^T(t)\mathbf{f}(\mathbf{x}, \mathbf{u}, t) \tag{6.2}$$

als vereinfachte Schreibweise können die notwendigen Bedingungen als

$$\dot{\boldsymbol{\lambda}} = -\left(\frac{\partial H}{\partial \mathbf{x}}\right)^T, \qquad \mathbf{0} = \left(\frac{\partial H}{\partial \mathbf{u}}\right)^T, \qquad \dot{\mathbf{x}} = \left(\frac{\partial H}{\partial \boldsymbol{\lambda}}\right)^T \tag{6.3}$$

mit den Radbedingungen

$$\mathbf{x}(t_0) = \mathbf{x}_0\,, \qquad \left(H + \frac{\partial S}{\partial t}\right)\Bigg|_{t_1} \delta t_1 + \left(\frac{\partial S}{\partial \mathbf{x}} - \boldsymbol{\lambda}^{\boldsymbol{T}}\right)\Bigg|_{t_1} \boldsymbol{\delta x_1} = 0$$

geschrieben werden.
Die Ausdrücke der rechten Seiten der Hamilton-Gleichungen 6.3 sind gegeben, womit die benötigten $2n$ Differentialgleichungen 1. Ordnung und die m algebraische Gleichungen gefunden wurden.
Für eine Integration der Differentialgleichungen müsste der Vektor der Stellgröße u(t) bekannt sein, bei der vorliegenden Problemstellung ist die Stellgröße außerdem beschränkt. Aus diesem Grund kann nur versucht werden, die Gleichungen

$$\mathbf{0} = \mathbf{h}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, t) \;\rightarrow\; \mathbf{u} = \tilde{\mathbf{h}}(\mathbf{x}, \boldsymbol{\lambda}, t)$$

nach der Stellgröße aufzulösen. Eingesetzt in die anderen Gleichungen kann versucht werden, die Anfangswerte der Lagrange-Multiplikatoren $\boldsymbol{\lambda}(t_0)$ anhand einer Optimierung zu variieren, damit eventuell eine Lösung gefunden wird, die mit ausreichender Genauigkeit die Randbedingungen erfüllt.
Der für diese Problemstellung erforderliche Rechenaufwand ist für aktuelle Rechner nicht akzeptabel, daher ist diese Methode nicht für den Einsatz in dieser Arbeit geeignet [6, Kapitel 7,8], [10, Kapitel 9-11]. Aufgrund der

Stellgrößenbeschränkung wäre außerdem die algebraische Gleichung

$$\mathbf{0} = \left(\frac{\partial H}{\partial \mathbf{u}}\right)^T$$

durch das im Allgemeinen viel schwieriger anzuwendende *Minimum-Prinzip von Pontryagin* zu ersetzen.

6.1.2 Dynamische Programmierung

Die dynamische Programmierung wurde von R. E. Bellman, wie in [2] als Vorgehensweise entwickelt, die sich auf dem von ihm formulierten Optimalitätsprinzip bezieht.
Das Optimalitätsprinzip von Bellman lautet wie folgt:

> *„Es sei* $\mathbf{u}^*(t)$ *die optimale Steuertrajektorie und* $\mathbf{x}^*(t)$ *die optimale Zustandstrajektorie. Jede Resttrajektorie* $\mathbf{u}^*(t)$, $t \in [t_1, t_e]$, $0 \leq t_1 \leq t_e$ *der optimalen Steuertrajektorie ist optimal im Sinne der Überführung des Zwischenzustandes* $\mathbf{x}^*(t)$ *in die Endbedingung* $\mathbf{g}(\mathbf{x}(t_e), t_e) = \mathbf{0}$ *[10, Seite 335].“*

Bei einer Teilung der optimalen Zustandstrajektorie $\mathbf{x}^*(t)$ in zwei Teile, ist nach der Aussage des Optimalitätsprinzip Teil 2 die optimale Zustandstrajektorie einer neuen Problemstellung. Als Lösungsweg kann in den meisten Fällen am hinteren Ende begonnen werden, um sich bis an den Beginn durch die einzelnen Stufen zu arbeiten. Am Ende erhält man nicht nur die optimalen Trajektorien zur Überführung des Anfangszustandes in die gewünschte Endbedingung, sondern auch ein optimales Steuergesetz. Mit diesem optimalem Steuergesetz ist es möglich, nicht nur den Anfangszustand in die gewünschte Endbedingung zu überführen, sondern jeden Punkt, sofern er überführbar ist.
Die Minimierung in den einzelnen Stufen kann sowohl analytisch, wie auch

numerisch durchgeführt werden. Bei komplexeren Problemen ist eine analytische Durchführung in der Regel nicht mehr möglich. Das Ergebnis ist ein globales Minimum der Problemstellung. Mit der diskreten dynamischen Programmierung werden noch einige Vorteile geboten, etwa dass Funktionen nicht stetig differenzierbar sein müssen. Ein Nachteil der dynamischen Programmierung ist es, dass der Rechenaufwand exponentiell mit der Ordnung des Systems wächst, somit ist diese Methode wiederum nicht für sehr komplexe Problemstellungen geeignet [2].

6.1.3 Evolutionäre Algorithmen

Der Algorithmus der Differential Evolution gehört zur Gruppe der evolutionären Algorithmen. Die Funktionsweise ist an die in der Natur vorkommende Evolution von Lebewesen angelehnt, die die Evaluierung verschiedenster Lösungsmöglichkeiten ebenso nützen, um zu einem möglichst guten Ergebnis zu gelangen. Es kann im Allgemeinen keine optimale Lösung in annehmbarer Zeit garantiert werden, jedoch wird in den meisten praktischen Fällen in akzeptabler Zeit ein Ergebnis gefunden, das gut genug für die betrachtete Anwendung ist.
Evolutionäre Algorithmen sind strukturell einfach aufgebaut. Sie arbeiten rundenbasiert, diese werden auch als Generationen bezeichnet, und agieren in einem Suchraum S. Anhand einer Funktion f, auch als Fitnessfunktion bezeichnet, wird den Punkten eine gewisse Güte zugewiesen. Ein evolutionärer Algorithmus arbeitet mit einer Anzahl an Punkten, auch als Population P bezeichnet, des Suchraumes, die zugehörigen Punkte der Population, die eine Teilmenge der Punkte im Suchraum darstellen, werden auch als Individuen bezeichnet. Die Population ändert sich mit jeder neuen Generation, daher wird die t-te Population mit P_t bezeichnet. Die erste Population P_0 wird Initialisierung genannt, die in der Regel zufallsbasiert, normalverteilt erstellt wird. Ein Zyklus, wie er bei evolutionären Algorithmen verwendet

werden kann, ist in Abbildung 6.1 dargestellt.
Für jedes Individuum der ersten Population wird der Fitness-Wert berechnet und gespeichert. Der erste Schritt jeder neuen Generation ist es, durch Selektion einige Individuen der Population auszuwählen, die verwendet werden, um neue Punkte im Suchraum zu erstellen. Diese ausgewählten Individuen werden auch als *Eltern* bezeichnet. Die Auswahl erfolgt dabei im Allgemeinen nach dem Wert der Fitness-Funktion der Individuen, je höher dieser Wert, desto größer die Wahrscheinlichkeit, dass das zugehörige Individuum ausgewählt wird. Als nächster Schritt wird auf diese Eltern das Prinzip der Variation angewandt, das mit Mutation und Rekombination neue Individuen erzeugt, die als *Nachwuchs* bezeichnet werden. Auf die Individuen des Nachwuchs wird noch einmal eine Selektion angewandt, um einige Individuen der aktuellen Generation durch neue, bessere Punkte des Nachwuchs zu ersetzen und daraus eine neue Generation zu formen, da in den meisten Fällen die Größe der Population gleich bleiben soll. Im Unterschied zur Selektion vor der Variation werden hier mit großer Wahrscheinlichkeit nicht die Individuen mit großen Werten der Fitness-Funktion, sondern jene mit kleinen Werten ausgewählt, damit die Population gesamt betrachtet nie einen schlechteren Fitness-Wert aufweist. Mit dieser neuen Generation wird anhand einer Abbruchbedingung untersucht, ob sie den geforderten Ansprüchen genügt, oder ob ein weiterer Zyklus stattfinden soll [7, Kapitel 1,2].

Differential Evolution

Das Prinzip auf dem der Algorithmus von Differential Evolution aufbaut, ist eine einfache, schnelle und populationsbasierte, stochastische Suchtechnik für globale Optimierungen. Es gibt mittlerweile sehr viele weiterentwickelte Versionen dieser Technik und die Einsatzgebiete sind weit gestreut. Die Basis bilden dabei die Vorgehensweisen der **Variation** und der **Selektion**, durch die sichergestellt wird, dass der gesamte vorhandene Suchbereich durch die

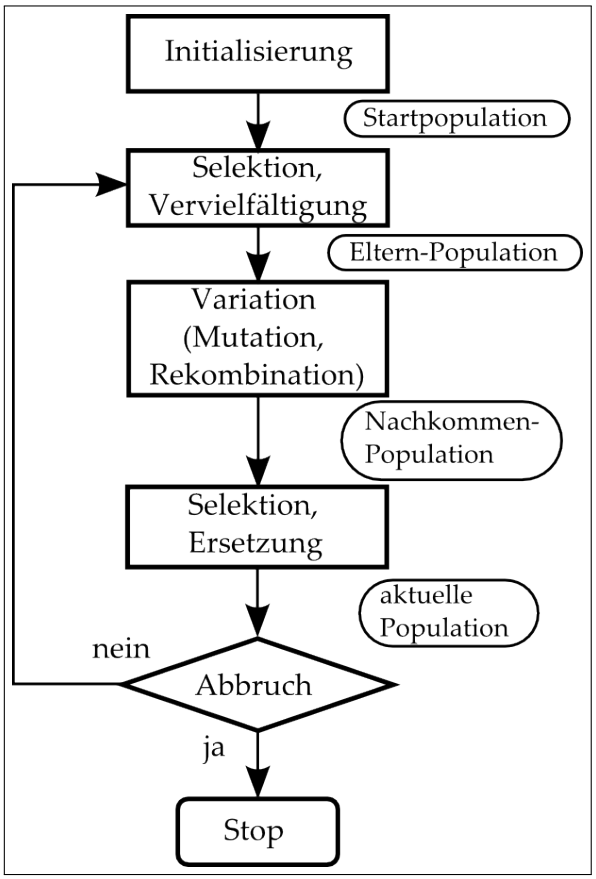


Abbildung 6.1: Evolutionszyklus von Evolutionären Algorithmen

Individuen erforscht wird und die bestmögliche Lösung gefunden werden kann. In einem D-dimensionalen Suchraum wird ein Individuum als D-dimensionaler Vektor $(x_{i_1}, x_{i_2}, \ldots, x_{i_D})$ mit der Laufvariable $i = 1, 2, \ldots, NP$ geschrieben, wobei NP die Anzahl der Individuen ist, also die Größe der Population.

Für die erste Population von Individuen wird normalerweise eine zufällige, normal verteilte Auswahl erstellt, die **Initialisierung** genannt wird. Sollten Beschränkungen vorliegen, kann die Initialisierung darauf angepasst werden, um schneller ein entsprechendes Ergebnis zu bekommen, da auf diese Weise Punkte, die nicht erlaubt sind, erst gar nicht erstellt werden.

Im Abschnitt der **Mutation** wird für jedes Individuum der aktuellen Population ein Versuchsvektor erzeugt, der durch eine Mutation eines Zielvektors mit einer gewichteten Differenz zweier Individuen entsteht. Als Zielvektor wird jeweils das Individuum bezeichnet, welches für den aktuellen Durchlauf der Mutation ausgewählt wurde, um als Basis für den neuen Versuchsvektor zu dienen. Der Versuchsvektor hingegen besteht aus Punkten, bei denen zum Zielvektor eine Differenz zweier, anderer Individuen mit einem Skalierungsfaktor multipliziert, addiert wird. Es wird sozusagen ein Punkt in der Nähe des Individuums, das als Zielvektor ausgewählt wurde, erforscht. Als Pseudo-Code kann dies wie folgt geschrieben werden:

1. Wähle den Zielvektor $x_{i_1}(G)$ zufällig aus der aktuellen Generation G der Population, wobei die Laufvariable $i \neq i_1$ sein muss, damit sichergestellt ist, dass sich die Vektoren voneinander unterscheiden.
2. Wiederhole den Vorgang für zwei weitere Zielvektoren $x_{i_2}(G)$ und $x_{i_3}(G)$, wobei $i \neq i_1 \neq i_2 \neq i_3$ gelte.
3. Der Versuchsvektor kann nun wie folgt berechnet werden:

$$u_i(G) = x_{i_1}(G) + F(x_{i_2}(G) - x_{i_3}(G))$$

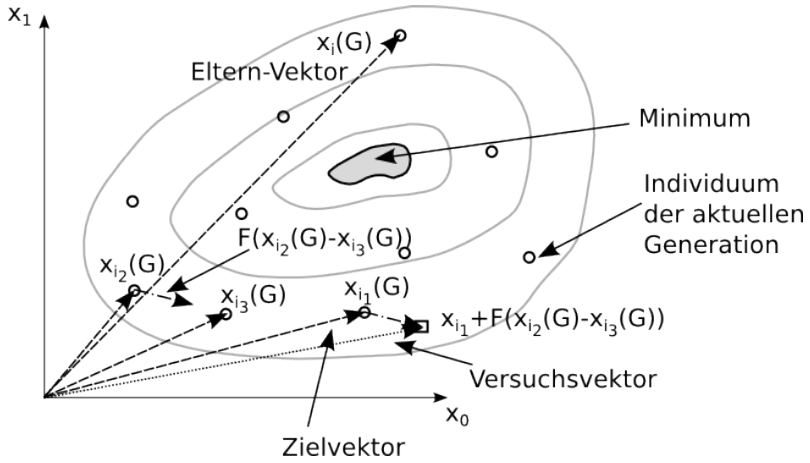


Abbildung 6.2: Mutation von Individuen

In dieser Gleichung ist $F \in [0, 2]$ ein Skalierungsfaktor, der verwendet wird, um die Größenordnung der Variation zu kontrollieren. Der schematische Vorgang für eine Mutation ist in Abbildung 6.2 dargestellt.

Die Nachkommen $x'_i(G)$ der aktuellen Generation werden durch die Methode der **Rekombination** der Eltern-Vektoren $x_i(G)$ und die Versuchsvektoren $u_i(G)$ erzeugt. Der Grundgedanke dabei ist, neue Punkte, die ihren Vorgängern in einer Weise ähnlich sind, zu erzeugen:

$$x'_{ij}(G) = \begin{cases} u_{ij}(G), & \text{wenn } j \in J \\ x_{ij}(G), & \text{sonst} \end{cases}$$

J ist in diesem Fall die Menge der Punkte, die ausgetauscht werden, $x_{ij}(G)$ das j-te Element des Vektors $x_i(G)$. Um die Menge der auszutauschenden Punkte zu ermitteln, gibt es eine Vielzahl an Möglichkeiten. Eine der häufigsten ist dabei die binomial-Rekombination, in der die Punkte zufällig aus der Menge der möglichen Punkte ausgewählt werden:

$J = \emptyset$; % leere Menge
$j^* \sim U(1, D)$;
$J \leftarrow J \cup j^*$;
for each $j \in 1 ... D$ **do**
  if $U(0,1) < CR$ and $j \neq j^*$ **then**
    $J \leftarrow J \cup j$;
  end if
end if

In diesem Pseudo-Code ist J die Menge der Rekombinationspunkte, CR die Wahrscheinlichkeit der Rekombination, *U(1,D)* ein normalverteilter Integerwert zwischen 1 und D, *U(0,1)* ein gleichverteilter Wert zwischen 0 und 1, sowie D die Dimension der Problemstellung. Das Prinzip der Rekombination ist in Abbildung 6.3 noch einmal dargestellt.

Die **Selektion** wird in zwei unterschiedlichen Fällen eingesetzt, einerseits um die Individuen für die Mutation auszusuchen, andererseits um die jeweils besseren Individuen der Nachwuchs-Population oder der Eltern-Population anhand ihrer Fitness-Werte zu eruieren und zu selektieren, damit der Fitness-Wert nicht verkleinert wird:

$$x_i(G+1) = \begin{cases} x'_i(G) & \text{wenn } f(x'_i(G)) > f(x_i(G)) \\ x_i(G) & \text{sonst} \end{cases}$$

Für die Selektion der Punkte der Mutation ist genau der umgekehrte Fall zu verwenden, es sollen diejenigen Punkte ausgesucht werden, die die kleinsten Fitness-Werte aufweisen. Das Basiskonzept des Differential Evolution-

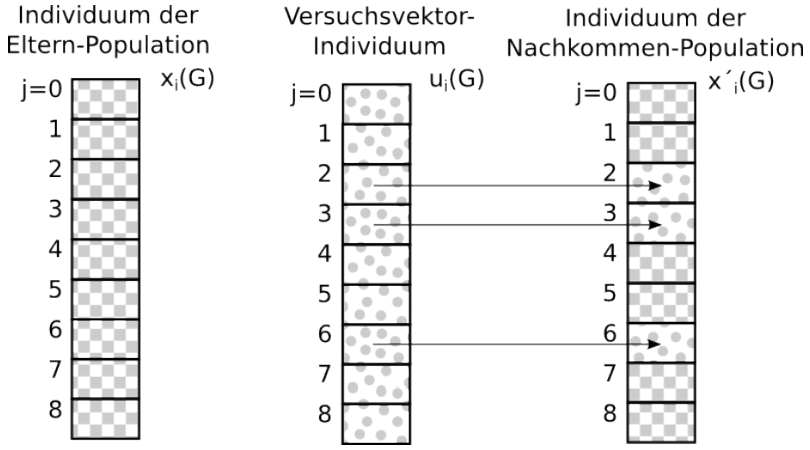


Abbildung 6.3: Rekombination von Individuen

Algorithmus kann und wurde in vielen Bereichen verändert und verbessert, welche in zwei Kategorien eingeteilt werden können:

- Hybridisierung, vor allem mit anderen evolutionären Algorithmen
- Mechanismen, um die Evolution besser ausnützen zu können [11]

6.2 Optimierung in Matlab

Mit der Streckenunterteilung in etwa 1 Zentimeter kleine Abschnitte und einer Gesamtlänge von mehreren Metern ist es aufgrund der Komplexität der Problemstellung nicht möglich, das Problem mit der Methode des Minimum-Prinzips in annehmbarer Zeit mit einem vernünftigen Aufwand zu lösen, diese ist daher nicht für die Verwendung geeignet. Ebenso kann dieselbe Aussage für die Dynamische Programmierung getroffen werden. Das einfachste und zielsicherste Mittel für die Optimierung ist somit, einen Evolutionären Algorithmus zu verwenden, wobei auch bei diesen darauf geachtet werden muss, dass die Populationen klein gehalten werden, was

sich natürlich auf das Ergebnis der Minimierung negativ auswirken kann. In dieser Arbeit wurde der Algorithmus des Differential Evolution für die Optimierung verwendet. Die Umsetzung in Matlab erfolgte wie in Kapitel 6.1.3 beschrieben mit den drei Punkten der Selektion, Mutation und Rekombination.

6.2.1 Initialisierung

Als erster Schritt der Optimierung ist der Schritt der Initialisierung durchzuführen. Hierfür wird eine Startpopulation der Steuerung

$$U0 = 100 * ones(nDim, N\_Agents) + rand(nDim, N\_Agents) * 30;$$

mit $nDim$, der Anzahl der Samples über den Streckenverlauf und N_Agents, die Größe der Population, festgelegt. Die Größe der Population bleibt in allen Generationen gleich. Danach sind noch die obere und untere Beschränkung der Steuergröße, als auch die maximale Iterationsanzahl festzulegen, die für eine Berechnung in annehmbarer Zeit auch klein gehalten werden muss.

Matlab-Code

```
%% Optimierung -DE, Initialisierung

N_Agents = 20;        % size of population
nDim = 15;            % number of samples along the track
fh = @fopt;           % optimizaton function

U0 = 100*ones(nDim,N_Agents)+rand(nDim,N_Agents)*30;      % init popu-
    lation with random values
```

```
% set options for optimization
opt.MaxFunEvals = 1000*25;       % maximum over all allowed evaluations
opt.Xmin = zeros(size(U0(:,1)));       % minimum level for U
opt.Xmax = ones(size(U0(:,1)))*130;       % maximum level for U
opt.Display = 'iter';       % show and store best value each iteration
opt.LimMethod = 'clip';       % method for values violating limits

opt.TolX = 1E-4;
opt.MaxIter = 500;       % maximum iterations
opt.method = 'de/best/1/bin';
opt.F = 0.6;       % settings for DE algorithm
opt.CR = 0.7;       % settings for DE algorithm
opt.K = 0.6;       % settings for DE algorithm
```

6.2.2 Optimierung

Bei der Optimierung wird der exakte Ablauf wie in Kapitel 6.1.3 umgesetzt, der einzige Unterschied liegt darin, dass hier keine Abbruchbedingung mit einer Gütefunktion vorliegt, sondern nach 500 Iterationen abgebrochen und die bis dahin beste ermittelte Lösung verwendet wird. Der Code kann wiederum im Anhang genau angesehen werden, ein Auszug mit den wichtigsten Abschnitten ist im nächsten Kapitel zu sehen. Der Ablauf der Optimierung sieht wie folgt aus:

- Nach festgelegtem Schema sind als Argumente der Funktion die Streckenteile der gewünschten Strecke anzugeben. Aus diesen Streckenteilen wird ein Spline für die gesamte Strecke erstellt, der alle benötigten Streckendaten enthält (Datei „time_optimal_control.m“)
- Die Limits der Startpopulation werden geprüft und gegebenenfalls angepasst, damit jedes Individuum im vorgegebenen Raum liegt. (Datei „DEopt.m“)

- Für die Startpopulation werden die Streckenparameter, wie die Parameter für das Slot-Car und die Simulation des Systems initialisiert, die Startwerte der Zustandsgrößen sind $x_0 = [0, 0, 0, 0]$, da das Auto gerade aufgesetzt aus dem Stillstand startet. (Datei „doInit.m“)
- Mit der vorgenommenen Initialisierung wird das System nun mit einer vorgegebenen, fixen Schrittweite von $dt = 0.01s$ simuliert, bis entweder die maximale Zeitvorgabe t_{max} überschritten wird, oder das Streckenende erreicht ist. Tritt bei einer Simulation der Fall ein, dass die Endzeit t_{max} überschritten wird, wird die Simulation beendet und der Wert für T, der im Gütefunktional von Bedeutung ist, auf den Wert von t_{max} gesetzt. Dadurch weist das daraus berechnete Gütefunktional mit Sicherheit einen größeren Wert auf, als eine Simulation, bei der die Berechnung der Zustandsgleichungen bis zum Streckenende ausgeführt wird. (Dateien „doSim.m“und „f_absq2.m“)
- Mit dem Gütefunktional $J = (3 * T)^3 + (3 * w)^2$ wird dem Resultat der Simulation ein Wert zugewiesen, um es beurteilen zu können. Hierbei ist T die Endzeit, wo das Slot-Car die Ziellinie überquert, $w = max(|\phi|)$ der maximal auftretende Driftwinkel und $|\phi(t = T)|$ der Absolutwert des Winkels bei der Zieldurchfahrt. Der erste Term ist hierbei für die Minimierung des Endzeitpunktes, der zweite Term soll den maximalen Winkel minimieren, der dritte Term soll sicherstellen, dass das Fahrzeug am Ende der Runde wieder möglichst gerade durch das Ziel fährt, was für die Möglichkeit einer Fahrt über mehrere Runden von Vorteil ist. (Datei „fopt.m“)
- Es werden so lange neue Populationen, mit den Schritten der Selektion und Variation, wie in Kapitel 6.1.3 beschrieben, erstellt, bis entweder die maximale Anzahl der Iterationen erreicht wird, oder die Anzahl der gesamten, maximal erlaubten Evaluierungen erreicht ist. Dabei wird für jede Population die Simulation erneut ausgeführt und mit dem Gütekriterium ein Wert errechnet. (Datei „DEopt.m“)
- Mit dem besten ermittelten Ergebnis wird die Simulation noch einmal

ausgeführt und die gesamten, ermittelten Werte der Zustandsgrößen x, der Zeit t und der Stellgröße u gespeichert und zurückgegeben. (Datei „time_optimal_control.m“)

Matlab-Code

In diesem Abschnitt werden werden die wichtigsten Code-Abschnitte der verwendeten Dateien gezeigt. Die Initialisierung der Funktion time_optimal_control und die Funktion f_absq2 wurden bereits in früheren Abschnitten dargestellt.

```
function [X_best, T_best, U_best, spline_track] = time_optimal_control(varargin)

%% Erstellung des splines mit der 1.Ableitung und der Krümmung durch die
    Streckenvorgabe
spline_track = Bspline_curve(varargin:);
spline_track = spline_track.compute_derivate(1);
spline_track = spline_track.compute_tangent;
save('spline_track','spline_track');
⋮
Initialisierung
⋮
[U,T_] = DEopt(fh,Uo,opt);      % call optimization algorithm

% final simulation
[par, xo] = doInit();       % init the simulation and track param.
[X_best, T_best, U_best] = doSim( xo, U, par, 1 );      % simulate the result
    and plot
end
```

```
function [xopt, fopt] = DEopt(fh, x0, opt, varargin)
% differential evolution algorithm for stochastic optimazation
% fh..... function handle or string of target function
% x0..... starting vector or array of starting vectors [x0_1, x0_2, ... x_0N]
% opt ... structure holding special settings
% varargin .... arguments to pass to the function to be optimized
⋮
% allocate variables
[n_x, N_a] = size(x0);        % extract dimensions of x and number of agents
if N_a<4, error('at least 4 agents are needed'); end
X = x0;
X = checkLimits( X, o,'RANDOM');
F = evalAll( fh, X, varargin );
⋮
while (iGen<=o.MaxIter && iFev<o.MaxFunEvals)
XU = X;        % clone from parent
FU = F;
for iInd = 1:min(o.MaxFunEvals-iFev,N_a)        % for all agents
% MUTATION (DE) AND RECOMBINATION (CROSSOVER)
% pick indizes for mutation
cIdx = pick_ProbN( o.CR, n_x );
% set at least one crossover index
if isempty(cIdx), cIdx=pick_iFromN(1,n_x); end
% pick agents for crossover
rIdx = pick_iFromN( 2, N_a, iInd );
% calculate child vector for indizes cIdx
XU(cIdx,iInd) = bestX(cIdx) + o.F * ( X(cIdx,rIdx(1)) - X(cIdx,rIdx(2)) );
% check limitations
XU(:,iInd) = checkLimits( XU(:,iInd), o, o.LimMethod );
% SELECTION (AGRESSIVE)
% evaluate new child
```

```
FU(:,iInd) = evalAll( fh, XU(:,iInd), varargin );
iFev = iFev+1;
if isBetter(FU(:,iInd),F(:,iInd)),        % child < parent
    X(:,iInd) = XU(:,iInd);
    F(:,iInd) = FU(:,iInd);
    if nObj==1 && isBetter(FU(:,iInd),bestF), % child < best
        idxBest = iInd;
    end
end
iGen = iGen+1;
end
end %end generations
⋮
end       %end function

function idx = pick_ProbN(p,n)
idx = find(rand(n,1)<p);
end
```

```
function f = evalAll( fh, x, varg )
  if exist('varg','var'),
      varg(2:end+1) = varg;
  end

  for i=size(x,2):-1:1
      varg1 = x(:,i);
      f(:,i) = feval( fh, varg: );
  end
end
```

```
function [ X,T,UT ] = doSim( x0, U, par, do_plot )

Un = numel(U);        % number of actuator levels
X = zeros(ceil(par.tmax/par.dt),numel(x0));        % allocate space for states
T = zeros(ceil(par.tmax/par.dt),1);        % allocate space for time
UT = T;        % allocate space for actuator

t = 0;        % inital time
n = 1;        % inital step counter
x = x0;        % initial state
X(n,:) = x;
T(n) = t;
UT(n) = U(1);

while x(1)<par.gesamt && t<par.tmax,
      L = x(1) - floor(x(1)/par.gesamt)*par.gesamt;
      idxU = floor(L/par.gesamt*(Un-1)+1);
      uL = U( idxU );
      xdot = f_absq2( t, x, uL, par, 0 ); % selbe Funktion wie f_absq2(v),
            % nur werden hier die Parameter mit übergeben
      x = x + xdot*par.dt;
      t = t + par.dt;
      n = n + 1;

      X(n,:) = x';
      T(n) = t;
      UT(n) = uL;
end
T(n+1:end)=[];
X(n+1:end,:)=[];
UT(n+1:end)=[];
```

```
function J = fopt( U )

[par, x0] = doInit();       % init track parameter
[ X,T] = doSim( x0, U, par, 0 );       % simulate

[psi, ~ ] = par.spline_track.get_PsiRho(X(:,1));       % absolute tangent angle
      of track and curvature
phi = X(:,3) - psi;       % relative angle between track and car
for j=1:length(phi)
      if phi(j) > pi
            phi(j) = phi(j) - 2*pi;
      end
      if phi(j) < -pi
            phi(j) = phi(j) + 2*pi;
      end
end

w = (max(abs(phi)));       % maximum absolute angle
J = (3*T(end))^3 + (3*w)^2 + (2*abs(phi(end)))^2;
end
```

6.3 Vergleichsversuche

Um die Funktionsfähigkeit der ermittelten optimalen Steuerung veranschaulichen zu können, wurden Testfahrten mit der Bahn für verschiedene Steuerungen durchgeführt. Einerseits wurde mit dem Controller per Hand gefahren und über dessen Taster die Bit-Zahl vorgegeben, andererseits wurden zwei Steuerungungen mit zwei unterschiedlichen Gütefunktionalen getestet.

6.3.1 Die Strecke

Für die Strecke wurde ein Layout verwendet, das in Abbildung 6.4 dargestellt ist: Es soll ein schnelles Fahren aufgrund des Profils schwierig machen, um die Vorzüge der Berechnung der optimalen Steuerung aufzuzeigen. Gefahren wurde immer auf der Außenbahn, die eine Länge von 6.625m hat. Die Erhöhung der Strecke, wo sich die Abschnitte kreuzen, wurde in der Steuerungsberechnung nicht berücksichtigt, was keinen großen Unterschied ausmachen kann, da die Änderung des Splines nur in z-Richtung statt findet und keinen Einfluss auf die Querbeschleunigungen hat, welche das Slot-Car aus der Bahn werfen können.

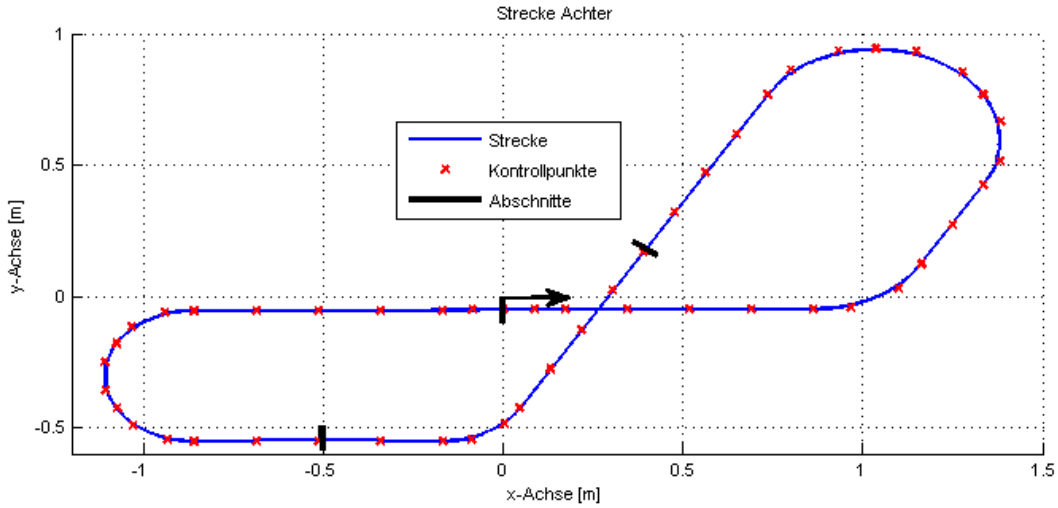


Abbildung 6.4: Strecke für Vergleichsversuche

6.3.2 Steuerungsvorgaben

Bei den zwei Steuerungsvorgaben wurde einmal mehr Wert darauf gelegt, dass der maximale Winkel klein bleibt, weshalb auch bei der Steuerungsvorgabe in Abbildung 6.7 die Bit-Werte im Allgemeinen kleiner ausfallen als

bei der zweiten Variante in Abbildung 6.9, wo die bestmögliche Zeit mehr im Vordergrund stand. Dies kann mit einer Änderung des Gütefunktionals erreicht werden. Bei der Berechnung der Steuerungen wurden kleine Populationen verwendet, um die Zeit der Berechnung kurz zu halten.

6.3.3 Versuche

Versuch 1: händische Steuerung

Bei dem Versuch der händischen Steuerung, fällt sofort auf, dass es keinesfalls eine einfache Aufgabe darstellt, das Slot-Car möglichst schnell zu bewegen. In einigen der ersten Runden wurde das Slot-Car zu schnell bewegt und ist dadurch aus der Bahn gefallen. Mit ein wenig Übung ist es schließlich gelungen, ein paar Runden hintereinander ohne Ausfall zu absolvieren. In Abbildung 6.6 kann bei genauer Betrachtung sofort erkannt werden, dass die Kennlinie von einer manuellen Fahrt stammt, da die Querbeschleunigungen an derselben Stelle jede Runde stark unterschiedlich sind. Die maximale Geschwindigkeit war mit $3.11\frac{m}{s}$ (berechnet) die höchste erzielte, jedoch flog das Slot-Car in der folgenden Kurve wegen der zu hohen Geschwindigkeit aus der Bahn. Die maximale Geschwindigkeit, wo das Fahrzeug in der Spur bleiben konnte wurde mit $2.65\frac{m}{s}$ berechnet.
Die Rundenzeiten lassen ebenfalls auf diese Tatsache schließen, die beste Rundenzeit ist mit 3.641 Sekunden im Vergleich zu der langsamsten Rundenzeit mit 18.620 Sekunden um 14.979 Sekunden schneller, wobei in dieser Runde das Slot-Car aus der Bahn fiel und erst wieder händisch in die Spur gesetzt werden musste. Daher ist die langsamste Zeit ohne Ausfall repräsentativer, diese beträgt 9.038 Sekunden, was immer noch einen Zeitunterschied von 5.397 Sekunden ausmacht. Der große Unterschied kommt dadurch zustande, dass das Fahrzeug zumindest einmal einen zu großen Driftwinkel besaß und erst durch die Federkraft und die Gleitreibung wieder gerade wurde. Zum Vergleich mit den optimalen Steuerungen wird

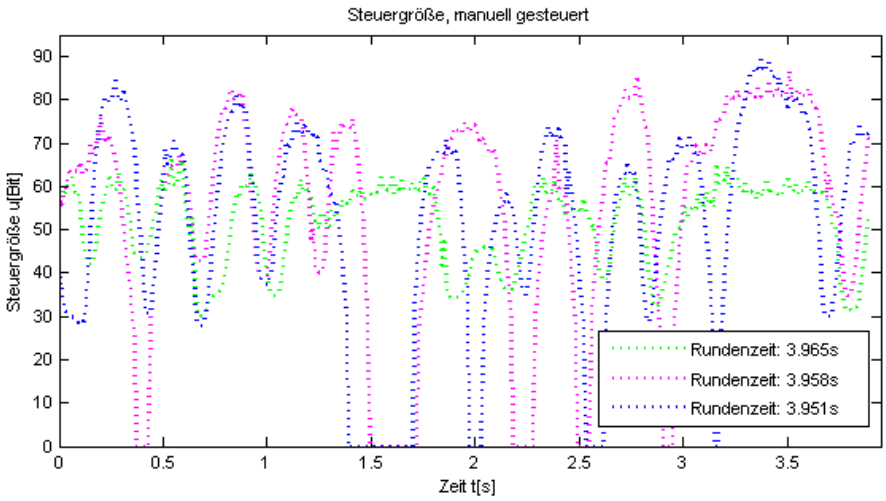


Abbildung 6.5: Steuerung, händische Vorgabe

zusätzlich die mittlere Rundenzeit aufgezeigt, diese beträgt 5.729 Sekunden über alle Runden und 4.491 Sekunden bei den Runden ohne Ausfall des Slot-Cars.

Versuch 2: erste optimale Steuerung

Bei der ersten berechneten optimalen Steuerung in Abbildung 6.7 fällt im Gegensatz zu der händischen Steuerung in Abbildung 6.5 auf den ersten Blick auf, dass die Steuerung immer dieselbe ist. Die Unterscheide kommen durch die Abtastrate, bzw. die Positionsbestimmung zustande. Dasselbe Bild zeigt sich bei den Querbeschleunigungen in Abbildung 6.8, es gibt nur sehr kleine Unterschiede, die von den Tatsachen stammen, dass die Bahn leicht uneben ist und das Modell nicht ideal ist. Auch die maximale, berechnete Geschwindigkeit ist mit $2.45\frac{m}{s}$ einiges langsamerer als die der händischen Steuerungsvorgabe.
Die Rundenzeiten geben ebenfalls Auskunft darüber, dass es sich hierbei

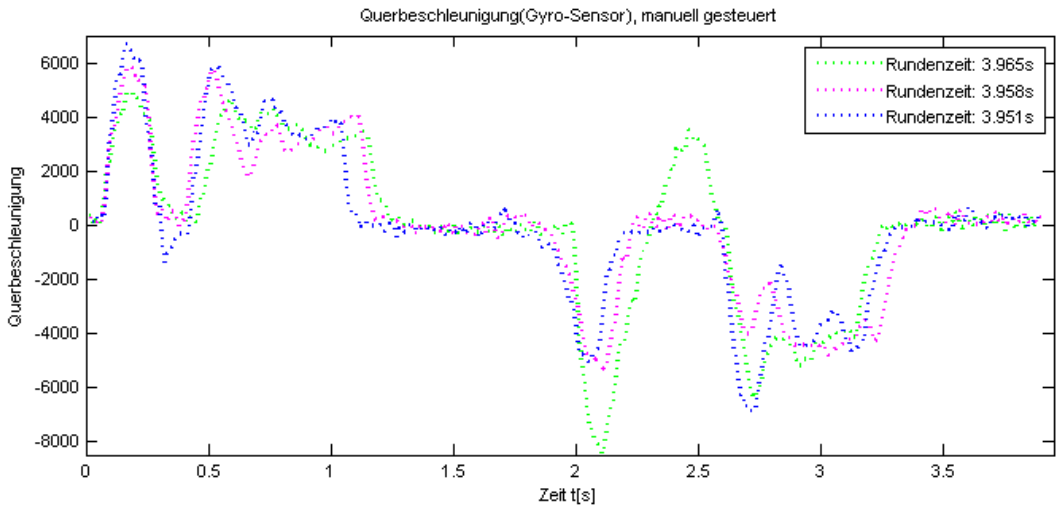


Abbildung 6.6: Querbeschleunigung, händische Vorgabe

um eine berechnete Steuerung handelt. Die schnellste Runde ist mit 4.124 Sekunden viel langsamer als die schnellste Runde der händischen Steuerung, was durch den konservativen Ansatz dieser Steuerung zu erklären ist. Das Slot-Car ist bei diesem Versuch kein einziges Mal von der Bahn abgekommen, es hatte auch keine großen Driftwinkel durch zu schnelle Kurvenfahrten, daher ist die langsamste Runde mit 4, 863 Sekunden auch nur um 0.739 Sekunden langsamer als die schnellste Runde. Der Mittelwert aller Runden beträgt 4.420 Sekunden.

Versuch 3: zweite optimale Steuerung

Mit dem Wunsch eine schnelleren Rundenzeit zu erzielen, wurde der Term des minimalen Driftwinkels kleiner gehalten und dafür der Term der minimalen Endzeit wichtiger. In Abbildung 6.9 sind die höheren Bit-Werte zu erkennen. Auch hier kann die optimale, berechnete Steuerung klar erkannt werden. Bei der Querbeschleunigung in Abbildung 6.10 ist ersichtlich, dass

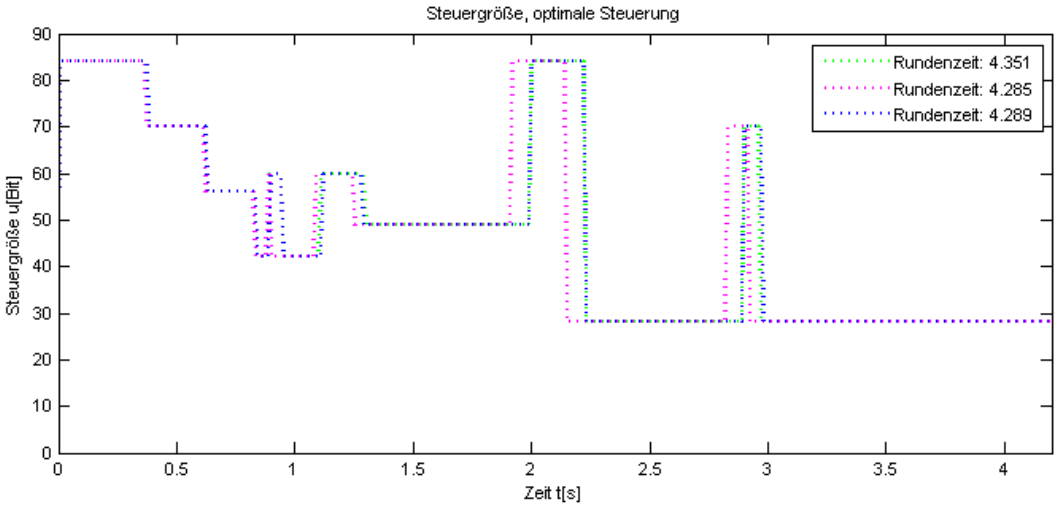


Abbildung 6.7: Steuerung, optimale Steuerung (kleiner Winkel ϕ)

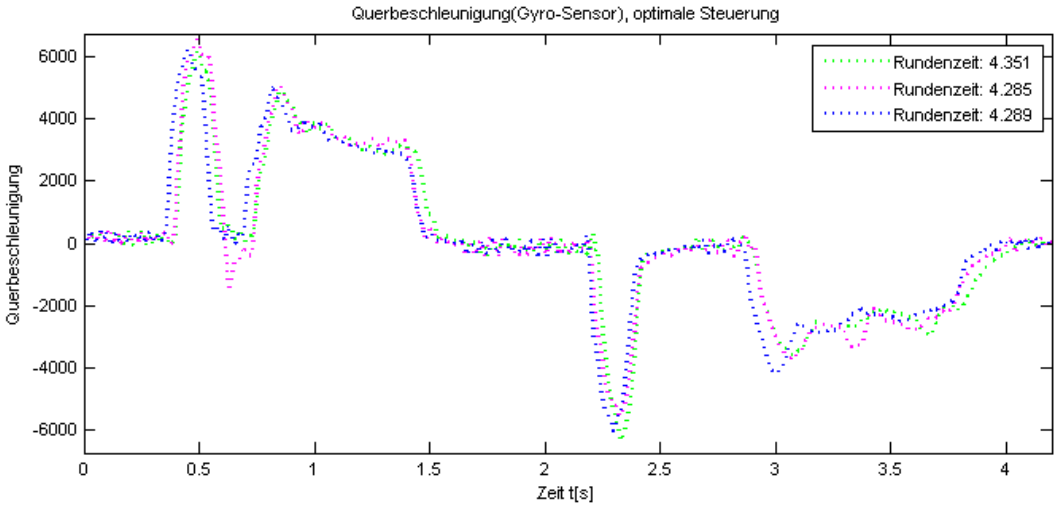


Abbildung 6.8: Querbeschleunigung, optimale Steuerung (kleiner Winkel ϕ)

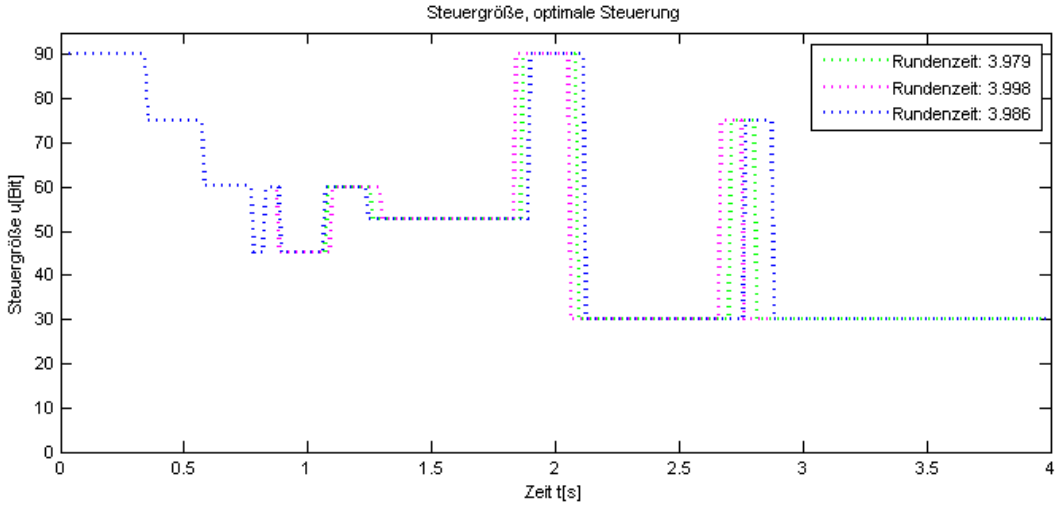


Abbildung 6.9: Steuerung, optimale Steuerung (kleine Endzeit)

das Slot-Car schneller unterwegs ist, da die Werte deutlich angestiegen sind. Die berechnete Geschwindigkeit ist mit $2.62\frac{m}{s}$ praktisch ebenso schnell wie die der händischen Vorgabe.
Bei den erzielten Rundenzeiten zeigt sich dasselbe Bild wie bei der maximalen Geschwindigkeit. Die schnellste Runde ist mit 3.979 Sekunden schneller als der erste Versuch der optimalen Steuerung. Die langsamste Runde ist mit 5.092 Sekunden hingegen langsamer, wobei der hohe Wert der Tatsache geschuldet ist, dass das Slot-Car einen zu großen Driftwinkel bekam und dadurch stark gebremst wurde, da der größte Anteil der Antriebskraft den Leitkiel in die Spurschiene drückt und nur ein kleiner Teil der Kraft für Vortrieb sorgt. Der Mittelwert ist mit 4.314 Sekunden der schnellste aller drei Versuche, jedoch ist der Unterschied zum zweiten Versuch minimal.

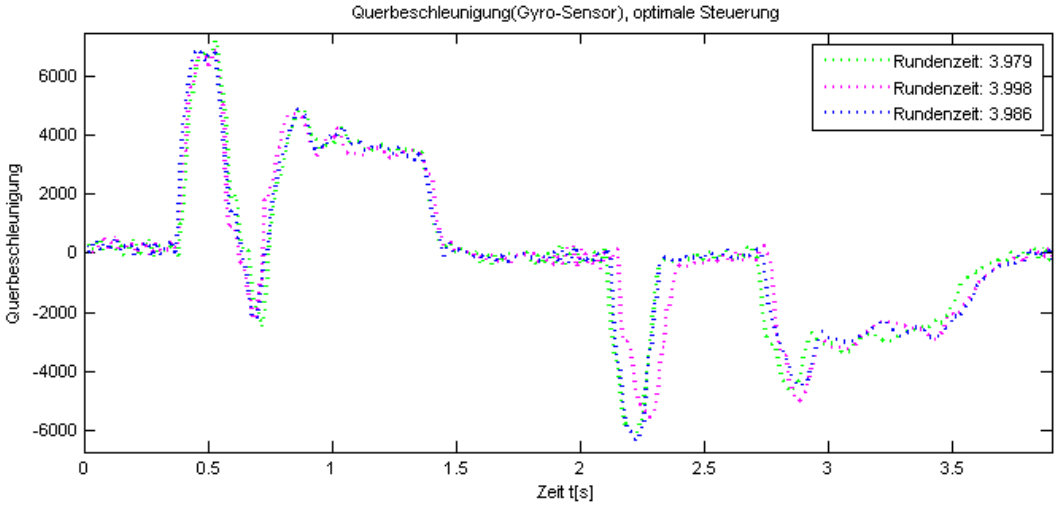


Abbildung 6.10: Querbeschleunigung, optimale Steuerung (kleine Endzeit)

6.3.4 Erkenntnisse

Mit den Ergebnissen der Versuche ist ersichtlich, dass die optimale Steuerung gut funktioniert. Auch wenn die schnellste Runde per Handsteuerung erzielt wurde, kann durch die optimale Vorgabe eine viel gleichmäßigere und im Mittelwert auch schnellere Zeit erzielt werden. Kann ein besserer Ansatz für das Gütefunktional gefunden werden, oder wird die Anzahl der Iterationen und die Anzahl der Populationen weiter erhöht, so kann die optimale Steuerung noch besser funktionieren. Zu beachten ist die Tatsache, dass aufgrund von Unebenheiten der Strecke und dem verwendeten Modellansatz nicht die eigentlichen Werte der berechneten Steuerungen verwendet werden konnten, sondern nur ein skalierter Wert davon, der etwa 75% der eigentlichen Werte darstellt.

6.4 Aufteilung der Strecke in mehrere Abschnitte

Das Ergebnis der Optimierung kann noch weiter verbessert werden. Dafür wird die Strecke, wie in Abbildung 6.4 dargestellt, in 3 Abschnitte unterteilt, die jeweils mindestens eine Kurve enthält, da die Optimierung auf einer Geraden immer die maximal mögliche Bit-Zahl als Vorgabe ergeben würde. Die Abschnitte beginnen und enden für eine einfachere Optimierung immer mit Geradenstücken. Die Optimierung wird nun für jeden Abschnitt einzeln durchgeführt, es muss nur darauf geachtet werden, dass bei der Initialisierung des folgenden Sektors der letzte Geschwindigkeitswert der vorangegangenen Optimierung als Startwert verwendet wird. Bei der Aufteilung der Sektoren wird im Gegensatz zu der Gesamtoptimierung für jeden Abschnitt die Anzahl der Individuen pro Population von 20 auf 50 erhöht.
Ein aussagekräftiger Wert, um beide Optimierungen miteinander vergleichen zu können, ist die Endzeit, bei der das Fahrzeug die Ziellinie durchquert. Diese beträgt bei der Gesamtoptimierung $4.05s$, hingegen ist der Wert der Optimierung der einzelnen Abschnitte $3.99s$. Als Erkenntnis erhält man die Tatsache, dass wenn die Optimierungsaufgabe in kleinere Segmente unterteilt wird, sie schneller arbeitet und ein besseres Ergebnis liefern kann, was vor allem bei großen Strecken von großem Vorteil ist.

7 Zusammenfassung

Ein geübter Fahrer kann im Vergleich zu einem ungeübten Fahrer deutlich schnellere Rundenzeiten auf einer Strecke einer Carrera-Bahn erreichen. Den Vergleich mit einer zeitoptimalen Steuerung, die mit Hilfe eines Modells der Strecke und der Fahrzeuge berechnet wird, kann allerdings auch ein solcher nicht gewinnen.
Um diese Steuerung zu bewerkstelligen, wurde ein Baukastensystem in Matlab für die Modellierung der Strecke entworfen, mit dem es möglich ist, mit einer Angabe des Streckenteils die nötigen Ableitungen der Strecke zu berechnen und zur Verfügung zu stellen. Das Fahrzeugmodell wurde auf Basis des linearen Einspurmodells entworfen, in dem zwei Massenpunkte mit einer festen Verbindung verwendet wurden, um damit alle nötigen Energien, Kräfte, etc. zu berechnen. Mit den Modellen der Strecke und der Fahrzeuge konnte eine (sub-)optimale Steuerung in Matlab entworfen werden, die mit dem Differential Evolution Algorithmus arbeitet, um zu einem möglichst guten Ergebnis in annehmbarer Zeit und mit beschränktem Speicher zu kommen.
Die Teile der Carrera-Digital 124-Bahn werden in zwei unterschiedliche Arten unterteilt. Einerseits gibt es Geraden mit drei verschiedenen Längen, andererseits Kurven in negative, sowie in positive Richtung, mit 30°-Stücken und 60°-Stücken und jeweils drei unterschiedlichen Radien. Ein weiteres Element, das in der Modellierung zwar berücksichtigt wurde, aber bei den Tests nicht verwendet wurde, ist das Stück, bei dem sich die Spuren kreuzen, damit es auf einer Bahn mit zwei Gleisen möglich ist, auch zu

überholen. Des weiteren besitzen die Fahrzeuge unterschiedliche Daten, die berücksichtigt werden müssen, da sie das Modell erheblich beeinflussen. In der Optimierung wurde das Modell Falken-Porsche verwendet und mit diesem die Anpassung der Modelldaten vorgenommen. Für andere Fahrzeuge muss die empirische Anpassung erneut durchgeführt werden. Der Controller wurde durch eine eigene Ansteuerung ersetzt, damit es möglich ist, direkt aus dem Programm Matlab die Steuerbefehle in Bit an den pulsweitenmodulierten Motor vorzugeben.
Der Aufbau eines Baukastensystems in Matlab ist mittels einer eigenen Klasse mit B-Splines als Grundlage erfolgt. Diese eignen sich daher sehr gut für diese Aufgabe, da die Ableitungen von B-Splines mit der Ordnung n wiederum B-Splines mit einer Ordnung niedriger, $n\text{-}1$, ergeben. Da man für die Differentialgleichungen die Beschleunigung, bzw. die Winkelbeschleunigung benötigt, ist es essentiell, die erhaltene Funktion zwei mal ableiten zu können, weshalb B-Splines 4.Ordnung verwendet werden. Bei den Versuchsvergleichen mit Kreisen, Ovalen und Geraden ist aufgefallen, dass die zweite Ableitung der erhaltenen Funktion einer Strecke so große Ungenauigkeiten aufweist, dass sie nicht mehr zu verwenden ist. Aus diesem Grund musste ein anderer Vorteil der Carrera-Bahn ausgenützt werden. Es gibt nur Geraden- und Kreisbogenstücke, weshalb die zweite Ableitung auch analytisch erfolgen kann, da die Ableitung einer Sinus- und Cosinusfunktion wiederum eine Cosinus- und Sinusfunktion ergibt. Deshalb wurde ein neuer Parameter eingeführt, der eine Unterscheidung zwischen Geraden und Kurven darstellt. Mit diesen Parametern werden alle nötigen Werte für die Modellierung bereit gestellt.
Das Fahrzeug wurde als Zwei-Massen-System modelliert, ähnlich einem Pendel mit einem beweglichen Drehpunkt. Als Basis für die auftretenden Kräfte, sowie die Massenpunkte wird das lineare Einspurmodell mit seinen Vereinfachungen verwendet. Die wichtigsten Punkte dabei sind vor allem, dass die Räder einer Achse in der Mitte als ein Rad zusammengefügt werden, was die Berechnungen erheblich vereinfacht, und dass der Schwer-

punkt der Massen auf Bodenniveau betrachtet wird, damit die Reifenkräfte keinen Einfluss haben. Durch die niedrigen Geschwindigkeiten der Carrera-Fahrzeuge kann außerdem der Luftwiderstand vernachlässigt werden, da er erst ab höheren Geschwindigkeiten an Einfluss gewinnt. Für die Beschreibung dieses Massensystems eignen sich die Lagrange Gleichungen 2.Art ideal, weil sie im Vergleich zur Newtonschen Mechanik besser für die Berücksichtigung holonomer Zwangsbedingungen geeignet sind. Weiters sind sie forminvariant, was von großem Vorteil sein kann. Ihr Geltungsbereich ist auf holonome Zwangsbedingungen und Kräfte, die sich aus einem Potential (Gravitationskräfte, Federkräfte) abgeleitet werden können, sowie auf allgemeine Kräfte, die durch Q_j berücksichtigt werden, beschränkt. Nach der Bestimmung der generalisierten Koordinaten, der Ortskoordinaten der zwei Massenpunkte und der eingeprägten Kräfte, sowie der Berechnung der verallgemeinerten Kräfte, der kinetischen Energie, der potentiellen Energie und der Dissipationsfunktionen können sowohl die Lagrange-Funktion als auch die Lagrange Gleichungen 2. Art aufgestellt werden. Mit einem einfachen Trick können diese Gleichungen in ein System erster Ordnung überführt werden. Somit erhält man anstatt von zwei Differentialgleichungen zweiter Ordnung vier Differentialgleichungen erster Ordnung, die sich in Matlab/ Simulink leicht integrieren lassen.
Für die Optimierung der Steuerung wurden drei verschiedene Arten in Betracht gezogen. Die erste Möglichkeit, das Prinzip der minimalen Wirkung in einer optimalen Steuerung zu verwenden, kann für die Problemstellung in dieser Arbeit nicht verwendet werden, da die Problemstellung zu komplex ist. Es kann keine Steuerungsfunktion u aus den Hamilton-Gleichungen gefunden werden kann, damit auch die anderen beiden Gleichungen gelöst werden könnten. Bei dem zweiten Ansatz, der dynamischen Programmierung, besteht ebenso das Problem, dass die Aufgabenstellung zu komplex ist und eine zu große Dimensionalität besitzt, dass dieses Problem mit der vorhandenen Rechenleistung und dem Speicherangebot gelöst werden kann. Somit wird in diesem Fall der Differential Evolution Algorithmus aus der

Klasse der Evolutionären Algorithmen verwendet. Mit diesem ist es möglich in akzeptabler Zeit und mit akzeptablem Aufwand eine suboptimale Lösung zu finden, indem eine Population von Individuen über viele Generationen mit den vorhandenen Werkzeugen der Selektion, Mutation und Rekombination soweit verbessert wird, bis eine vorher benannte Abbruchbedingung erfüllt wird. Mit einer entsprechenden Umsetzung in Matlab erhält man einen Steuervektor, der im Vergleich zu einer manuellen Steuerung des Fahrzeugs auf derselben Bahn um einiges näher an einer zeitoptimalen Lösung liegt.

Literatur

[1] Richard H. Bartels, John C. Beaty und Brian A. Barsky. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann Publishers, INC., 1987. ISBN: 0-934613-27-3 (siehe S. 9).

[2] Richard Bellman. *Dynamic Programming*. 1. Edition. Princeton Univ. Press, 1957 (siehe S. 100, 101).

[3] Carrera. *Streckenvorschläge für Carrera-Bahnen*. 2014. URL: `http://www.carrera-toys.com/de/live-racing/streckenvorschlaege/` (siehe S. 28).

[4] Carl deBoor. *A practical guide to splines*. Rev. ed., 1. hardcover print. Springer-Verlag New York Berlin Heidelberg, 2001. ISBN: 0-387-95366-3 (siehe S. 9, 22).

[5] Anton Hofer. »Vorlesungsskriptum der TU Graz: Computerunterstützte Modellbildung und Simulation«. 2004 (siehe S. 47).

[6] Anton Hofer. »Vorlesungsskriptum der TU Graz: Entwurf optimaler Systeme«. 2004 (siehe S. 98, 99).

[7] Thomas Jansen. *Analyzing Evolutionary Algorithms*. Springer-Verlag Berlin Heidelberg, 2013. ISBN: 978-3-642-17338-7 (siehe S. 102).

[8] Friedhelm Kuypers. *Klassische Mechanik*. 5., überarbeitete Auflage. WILEY-VCH Verlag, 1997. ISBN: 3-527-29269-1 (siehe S. 41, 42, 47, 48).

[9] Manfred Mitschke und Henning Wallentowitz. *Dynamik der Kraftfahrzeuge*. 4. Auflage. Springer-Verlag Berlin Heidelberg New York, 2004. ISBN: 3-540-42011-8 (siehe S. 35, 36).

[10] Markos Papgeorgiou, Marion Leibold und Martin Buss. *Optimierung, statistische, dynamische, stochastische Verfahren für die Anwendung*. 3., neu bearbeitete und ergänzte Auflage. Springer Verlag, 2012. ISBN: 978-3-540-34012-6 (siehe S. 98–100).

[11] Harish Sharma, Jagdish Chand Bansal und K.V. Arya. »Self Balanced Differential Evolution«. In: *Journal of Computational Science* (2012) (siehe S. 107).