Paul Karoshi

Entwicklung einer Methodik zur Optimierung von Mehrkörpersystemen

Diplomarbeit

Diplomingenieur

Maschinenbau

Technische Universität Graz

Fakultät für Maschinenbau

Institut für Fahrzeugtechnik

Member of Frank Stronach Institute

Institutsleiter: Univ.-Doz. Dipl.-Ing. Dr. techn. Arno Eichberger

Betreuer: Dr. techn. Andrés Eduardo Rojas Rojas

Graz, Jänner 2013

Danksagung

Ich möchte mich bei allen Personen bedanken welche mich bei dieser Arbeit unterstützt haben.

Bei meinen Kollegen am Institut für Fahrzeugtechnik für die freundschaftliche Athmosphäre und große Hilfsbereitschaft.

Bei Andrés Eduardo Rojas Rojas für die gute Betreuung, den steten Ansporn und kritischen Blick auf diese Arbeit.

Bei Gintarė Šiaulytė die mir in dieser anstrengenden Zeit, mit viel Kraft und Verständnis, zur Seite gestanden ist.

Bei meinen Eltern.

Danke.

Eidesstattliche Erklärung

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: Unterschrift:

Abstract

Due to more and more functionalities integrated in todays vehicles, multibody systems (MBS) become more and more important. Generally this becomes a challenge for design and calculation, as several characteristic parameters, describing the systems characteristic behavior, have to be considered and optimized simultaneously. In this thesis a method for optimizing multibody systems, such as a wheel suspension and a steering, is introduced. However, the established program is kept universal, so that any multibody-system can be analyzed.

As a co-simulation between Catia V5® and Matlab/Simulink®, a Matlab® based program, MuBOS (**M**ulti**B**ody **O**ptimization **S**ystem), was developed. It supports the design and simulation process in an early stage of product development. Therefor CAD-data is automatically transmitted to an MBS-model modelled in Matlab/Simmechanics®. Using a Nelder - Mead simplex algorithm the system geometry can be optimized with respect to characteristic parameters. Any number of characteristic parameters can be freely defined. Subsequent the optimized geometry of the multibody system is transferred to the CAD-model.

With this method a MBS-model can be quickly evaluated and optimized, when it comes to changes in the design process.

Two examples of use are presented. The optimization of the camber and toe angle is demonstrated using a control blade suspension system for the rear axle of a passenger car. The optimization of the four-wheel-steering, with recpect to the ackermann criterion, is shown for a special purpose mining vehicle.

Kurzfassung

Im heutigen Fahrzeugbau werden immer mehr Funktionalitäten in verschiedene Baugruppen integriert. Dies stellt für die Konstrukion und Auslegung von Mehrkörpersystemen eine komplexe Aufgabe dar, da immer mehr Parameter berücksichtigt und optimiert werden müssen. In der hier vorliegenden Arbeit wird eine Methodik zur Optimierung von Mehrkörpersystemen, insbesondere Radaufhängungen und Lenkungen, im Fahrzeugbau vorgestellt. Dabei wurden die Problemstellungen so allgemein definiert, dass auch beliebige Mehrkörpersysteme betrachtet werden können.

Im Sinne einer Co-Simulation zwischen Catia V5® und Matlab/Simulink® wurde ein Matlab basiertes Programm, MuBOS (**M**ulti**B**ody **O**ptimization **S**ystem), entwickelt, das den Konstruktions- und den Simulationsprozess in einer frühen Produktentwicklungsphase sinnvoll unterstützt. Dabei werden Geometriedaten vom CAD-System automatisiert auf ein MKS-Modell in Matlab/Simmechanics® übertragen. Dort kann das Modell simuliert und anhand charakteristischer Parameter optimiert werden. Im Optimierungsprozess werden, unter Verwenung eines Nelder-Mead Simplex Algorithmus, die Gelenkpositionen des Systems variiert und so nach einem Optimum der charakteristischen Parameter gesucht. Diese können für jede Anwendung frei definiert werden. Um die Integration möglichst vieler Funktionen sicherzustellen, können beliebig viele charakteristische Parameter definiert und optimiert werden. Im Anschluß an die Optimierung werden die Geometriedaten, ebenso automatisiert, wieder in das CAD-Modell übertragen. Damit ist es möglich auch sehr kurzfristige Änderungen in der Konstruktion rasch mit entsprechenden Berechnungen zu untermauern.

In zwei Anwendungsbeispielen wird die Optimierung von Sturz- und Vorspurwinkel einer PKW Schwertlenker Achse und die Auslegung einer Allradlenkung nach Ackermann eines Sonderfahrzeuges betrachtet.

Inhaltsverzeichnis

Abkürzungen

MKS	Mehrkörpersystem
GUI	Graphical User Interface
HiL	Hardware in the loop
NMA	Nelder-Mead Simplex Algorithmus

Formelzeichen

Koordinatensystem

$\mathcal{O}$	Koordinatenursprung
$\mathcal{W}$	Radaufstandspunkt
$\mathcal{EG}$	Lenkachse

Parameter und Konstanten

J	Trägheitstensor, Kostenwert
l	Radstand
m	Masse
n	Nachlauf
r_s	Lenkrollradius
s	Spurweite
δ	Vorspurwinkel
γ	Sturzwinkel
σ	Spreizungswinkel
τ	Nachlaufwinkel
δ	Lenkwinkel
$\Delta\delta$	Ackermannwinkel, Lenkwinkeldifferenz

1. Einführung

1.1. Simulation von Mehrkörpersystemen

Technische Produkte sind heutzutage ohne den Einsatz von rechnergestützten Entwicklungsmethoden, CAx, nicht mehr vorstellbar. Eine wichtige Rolle spielt dabei die Berechnung des dynamischen Verhaltens eines Systems. Grundlage für jede Untersuchung ist ein geeignetes physikalisches Ersatzmodell. Dieses soll die wesentlichen Eigenschaften des realen Systems darstellen. Gleichzeitig soll der Aufwand zur rechnergestützen Berechnung gering bleiben. Es gilt also der Grundsatz: So komplex wie nötig, so einfach wie möglich, [Woe11].

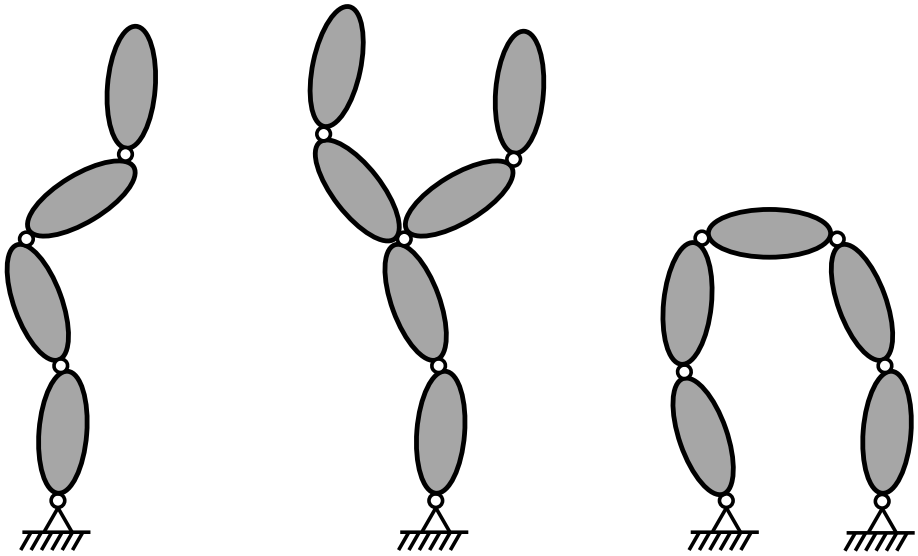
Mechanische Systeme bestehen aus massenbehafteten, elastischen Körpern, auf die über Volumen, Oberflächen und Gelenke Kräfte einwirken. Diese Systeme können mit Hilfe der Kontinuumsmechanik durch partielle Differentialgleichungen beschrieben werden. Allerdings ist eine analytische Lösung nur in einfachen Sonderfällen möglich. Im Allgemeinen werden die Systeme räumlich diskretisiert und mit numerischen Näherungsverfahren berechnet, [vS99].

Ein Mehrkörpersystem besteht aus Punktmassen und räumlichen Körpern, beschrieben durch Masse m und Trägheitstensor $\mathbf{J}$, die in ihrer Bewegung durch Bindungen (Gelenke) beschränkt sind. Die im System wirkenden Kräfte und Momente können eingeprägt, z.B. durch Lagestellglieder (rheonome Bindung) oder in Gelenken (skleronome Bindung) und Körpern hervorgerufene Schnittkräfte sein. Weitere typische Komponenten sind Federn, Dämpfer, Kraftstellglieder, [vS99].

Grundsätzlich lassen sich Mehrkörpersysteme anhand ihrer Topologie einteilen. Es wird zwischen offenen und geschlossenen Systemen unterschieden. Wird bei einem offenen System ein Gelenk entfernt, zerfällt das System in zwei Teile, Abbildung 1.1(a) bis 1.1(c), [vS99].

- Offene Systeme in Kettenform
- Offene Systeme in Baumform
- Geschlossene Systeme

Die Bewegung eines Mehrkörpersystems wird durch seine Bindungen eingeschränkt. Die Anzahl der Freiheitsgrade einses Systems ist abhängig von der Anzahl der Körper, sowie Anzahl und Art der Gelenke und Stellglieder.



(a) Offen (b) Offen (c) Geschlossen

Abbildung 1.1.: Topologien von Mehrkörpersystemen

$$n_{dof} = 6 \cdot n_b - n_\lambda \tag{1.1}$$

In Gleichung 1.1 ist n_{dof} die Anzahl der Freiheitsgrade und n_b die Anzahl der beteiligten Körper. Ohne Gelenke hätte das System $6 \cdot n_b$ Freiheitsgrade, für jeden Körper drei translatorische und drei rotatorische. Durch die im System vorhandenen Gelenke wird der Freiheitsgrad um die Anzahl der Gelenkbedingungen n_λ verringert,[vS99].

1.1.1. Typische Anwendungen von Mehrkörpersimulation

Kinematische Berechnung: Hier wird das Mehrkörpersystem unter Vernachlässigung von Kräften und Massen der Körper berechnet. Die Problemstellungen sind daher rein geometrischer Natur.

- Vorwärtssimulation: Ausgehend von Lagestellgliedern werden die Positionen aller Elemente im gesamten Simulationszeitraum berechnet. Diese Methode wird am häufigsten verwendet.
- Rückwärtssimulation: Hier werden Positionen, Geschwindigkeiten und Beschleuni-

gungen von Lagestellgliedern berechnet, die gewünschte Positionen, Geschwindigkeiten und Beschleunigungen im Mehrkörpersystem hervorrufen.

Im Allgemeinen wird die kinematische Berechnung als Unterfunktion der dynamischen Simulation durchgeführt.

Dynamische Berechnung: In dieser komplexen Simulation werden zusätzlich Kräfte, Massen, Trägheitstensoren und Schwerpunktlagen berücksichtigt. Vorausgesetzt wird, dass die Kinematik des Mehrkörpersystems lösbar ist. Es treten oft Diskontinuitäten der Zustandsvariablen auf, was eine schwierige Aufgabe in der Lösung der Gleichungssysteme darstellt. Auch hier wird zwischen Vorwärts- und Rückwärtssimulation unterschieden.

Optimierungsaufgaben: Moderne computergestützte Berechnungsmethoden werden nicht nur zur Analyse von Mehrkörpersystemen verwendet, sondern unterstützen den Anwender während des gesamten Entwicklungsprozesses. Zwei Problemstellungen sollen hier erwähnt werden, [vS99]:

- Parameteridentifikation: Bestimmte Parameter eines Modells sind zu bestimmen, um ein mit der Realität übereinstimmendes Verhalten des Systems zu erzielen. Das ist ein wichtiger Vorgang im Zuge der Validitierung dynamischer Systeme.
- Parameteroptimierung: Ein Mehrkörpersystems, das durch einen gewissen charakteristischen Parameter beschrieben wird, soll ein bestimmtes Verhalten aufweisen. Das wird durch gezielte Variation einer Gruppe von Parametern erreicht.

In der Fahrzeugtechnik wird Mehrkörpersimultion zur Modellierung von Baugruppen bis hin zum Gesamtfahrzeug in unterschiedlicher Modellierungstiefe angewendet. Dabei kommen spezifische Modelle zur Beschreibung des Rad-Schiene- und Reifen-Fahrbahn Kontaktes zum Einsatz. Damit können beispielsweise Fahrverhalten, Rundenzeiten im Rennsport bis hin zu Bauteilbeanspruchung von Komponeneten berechnet werden. Im Entwicklungsprozess können damit zeit- und kostenintensive Fahrvesuche zum Teil durch Berechnungen erstetzt werden, [Dür11] [Woe11].

1.2. Problemstellung im klassischen Produktentwicklungsprozess

Mit immer steigendem Wettbewerb ist es für Automobilunternehmen stets ein Ziel rasch auf Anforderungen der Märkte reagieren zu können. Um dieses Ziel zu erreichen ist es ein Bestreben, die Produktentwicklungszeiten möglichst kurz zu halten. Der klassische Produktentwicklungsprozess läuft seriell ab, wobei jeder Entwickungsschritt nach dem Ende des vorigen beginnt. Diese Vorgehensweise führt zu langen Entwicklungszeiten. Finden simultane Prozessabläufe (Simultaneus Engineering) statt, dann kann Potential zur Verkürzung des Produktentwicklungsprozesses genutzt werden. Dabei beginnt jeder Einzelschritt eines Prozesses zum frühestmöglichen Zeitpunkt. Die Prozesse werden so weit wie möglich simultan durchgeführt. Abbildung 1.2 zeigt den Vergleich und das Einsparungspotential dieser beiden Prozesse.

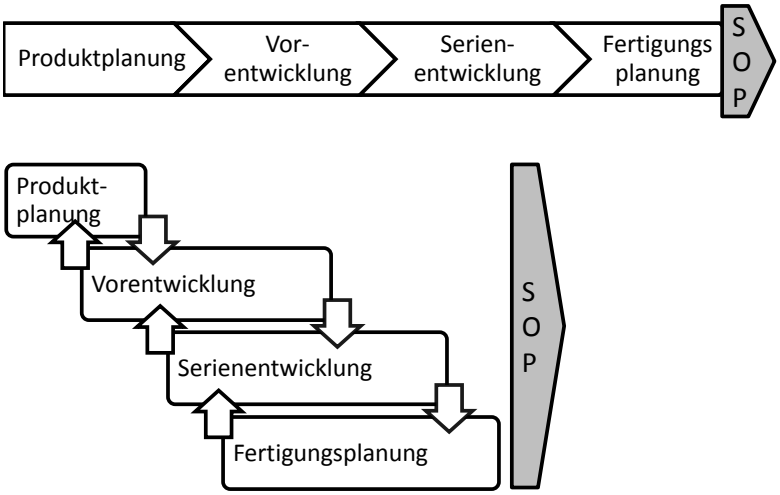


Abbildung 1.2.: Koordinatensystem nach ISO 8855, nach: [BS07]

Diese Zeitersparnis ist allerdings nur möglich, wenn zwischen den einzelnen Prozessen eine gute Kommunikation stattfindet. Der Austausch von Informationen zwischen Entwicklungs und Fertigungsabteilung, aber auch innerhalb dieser, ist der entscheidende Vorteil des Simultaneus Engineering, [BS07].

Im Entwicklungsprozess des Fahrwerkes wird nach dem Festlegen des Fahrwerkkonzeptes und des Lastenheftes die Kinematik der Radaufhängungen und der Lenkung ausgelegt, bzw. mit Mehrkörpersimulationen optimiert. Im nächsten Schritt werden im System wirkende Kräfte berechnet und die Bauteile und deren Steifigkeiten entsprechend ihren Belastungen dimensioniert. Im Simultaneous Engineering wird dieser Prozess in mehreren Iterationen durchlaufen, [HE07].

1.3. Einfluss des Fahrwerks auf Fahrdynamik und Komfort

Das Fahrwerk bildet das Verbindungsglied zwischen Fahrbahn und dem Aufbau (PKW, Busse) bzw. dem Rahmen (Nutzfahrzeuge). Die Radaufhängung hat als Verbindung des Aufbaues zum Rad die Aufgabe, Kräfte zwischen Rad und Aufbau zu übertragen, aber auch das Rad exakt zu führen. Allgemeine Anforderungen an eine Radaufhängung sind, [Wol09]:

- Geringes Gewicht
- Kostengünstig
- Haltbarkeit

- Wartungsfreiheit
- Geringer Platzberdarf

Die vielfältigen Anforderungen an das Fahrwerk lassen sich grob in drei Kategorien unterteilen, [HE07].

- Fahrdynamik
- Fahrsicherheit
- Fahrkomfort

Im Folgenden werden grundlegende charakteristische Parameter von Radaufhängung und Lenkung definiert, soweit sie diese Arbeit betreffen und die oben erwähnten Punkte beeinflussen.

1.3.1. Definition des Koordinatensystems

In MuBOS wird ein globales Koordinatensystem nach ISO 8855, Abbildung 1.3 verwendet.

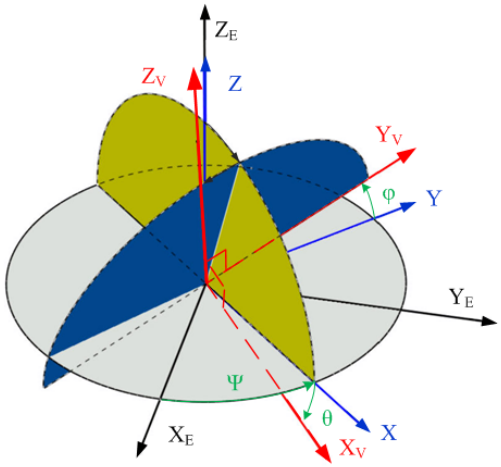


Abbildung 1.3.: Koordinatensystem nach ISO 8855, Quelle: [Wil10]

Dabei sind:

- X_E, Y_E, Z_E: Koordinaten des ortsfesten Inertialsystems
- X_V, Y_V, Z_V: Fahrzeugfestes Koordinatensystem. Der Koordinatenursprung befindet sich auf der Fahrbahn in Mitte der Vorderachse.
- X, Y, Z: Horizontiertes Koordinatensystem. Der Koordinatenursprung liegt auf der Fahrbahnoberfläche. Die X-Achse weist gegen die Fahrtrichtung, die y-Achse nach rechts und die z-Achse nach oben.

- φ: Rollwinkel, Drehung des Fahrzeuges um die X-Achse.
- Θ: Nickwinkel, Drehung des Fahrzeuges um die Y-Achse.
- Ψ: Gierwinkel, Drehung des Fahrzeuges um die Z-Achse

1.3.2. Radaufhängung

Die räumliche Bewegung des Rades beim Einfedern und Lenken wird durch die kinematischen Eigenschaften der Radaufhängung bestimmt. Durch das Aufhängungskonzept wird die Topologie der kinematischen Kette festgelegt. Durch die Anordnung der Aufhängungspunkte am Aufbau werden Größen wie Radstand und Spurweite sowie Rad- und Felgengrößen bestimmt. Im folgenden werden daher die wichtigsten fahrwerkspezifischen Kenngrößen beschrieben, [HE07].

Radstand l: Abstand zwischen den Radaufstandspunkten W der Vorder- und Hinterräder, in der Projektion auf die x-y Ebene, Abbildung 1.4, [HE07].

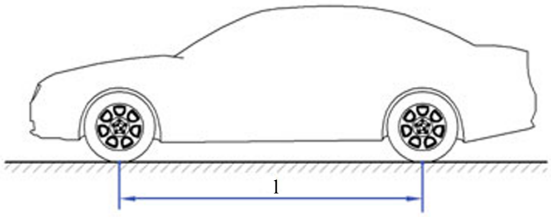


Abbildung 1.4.: Radstand l nach ISO 612/DIN70000, Quelle: [HE07]

Ein langer Radstand bedeutet:

- Geringes Nicken, damit guten Fahrkomfort
- Gute Fahrsicherheit

Ein kurzer Radstand bedeutet:

- Handlich in Kurvenfahrt und beim Parken
- Geringe Kosten und Gewicht

Das Verhältnis von Radstand zu Fahrzeuglänge soll möglichst groß sein. Durch die Einfederbewegung der Räder können sich die Positionen der Radaufstandspunkte und damit der Radstand ändern, [HE07].

Spurweite s: Abstand von linkem und rechtem Radaufstandspunkt W einer Achse, in der Projektion auf die x-y Ebene, Abbildung 1.5, [HE07].

Eine große Spurweite bewirkt:

- Gutes Fahrverhalten

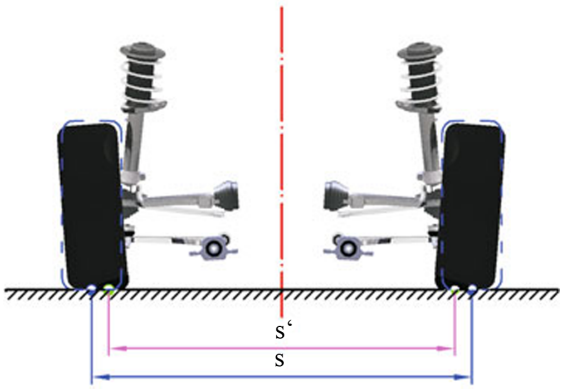


Abbildung 1.5.: Spurweite s, Quelle: [HE07]

- Geringes Wanken und damit guten Fahrkomfort
- Große Fahrzeugbreite
- Verziehen bei ungleich verteilten Bremskräften

Die Spurweite ist dabei natürlich durch die Fahrzeugbreite begrenzt. Spurweitenänderung entsteht durch Verschiebungen der Radaufstandspunkte, z.B. durch die Sturzänderung beim Einfedern des Rades. Spurweitenänderung führt zu:

- Querschlupf in der Radaufstandsfläche
- Gestörter Geradeausfahrt
- Reifenverschleiß
- Erhöhtem Rollwiderstand

Im Allgemeinen soll die Spurweitenänderung um die Konstruktionslage möglichst gering sein, [HE07].

Radhub s: Verschiebung des Radauftandspunktes W von der Konstruktionslage aus. Einfedern ist als postiver Radhub definiert. Übliche Werte sind für PKW bei Einfedern 60 bis 100 mm und bei Ausfedern 70 bis 120 mm, [HE07].

Spurwinkel δ: Winkel zwischen den Schnittlinien der Radmittelebenen der Räder einer Achse, Abbildung 1.6, [HE07].

Der Spurwinkel ist als Vorspur definiert, wenn $C < B$ gilt und als Nachspur, wenn $C > B$ gilt, Abbildung 1.6. Für einen geringen Reifenverschleiß ist ein Vorspurwinkel von $\delta = 0\,^\circ$ anzustreben. Der Spurwinkel beeinflusst das Fahrverhlaten sehr stark und wird je nach Bauform der Radaufhängung und Einsatzzweck durch Kinematik und Elastokinematik gesteuert. Um das dynamische Fahrverhalten zu beeinflussen werden Spurwinkel von $\delta = -30\,’$ bis $+30\,’$ gewählt, [HE07] [Wal06].

Sturzwinkel γ: Winkel zwischen einer senkrecht zur Fahrbahn liegenden Ebene und

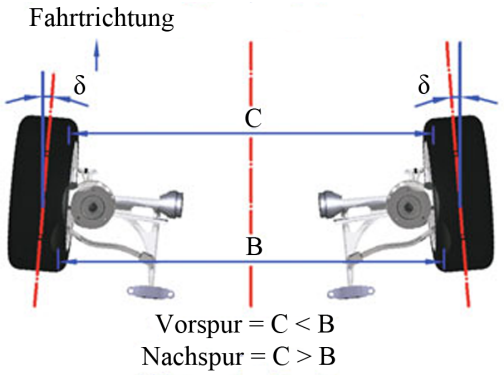


Abbildung 1.6.: Spurwinkel δ, Quelle: [HE07]

der Radmittelebene, Abbildung 1.7, [HE07].

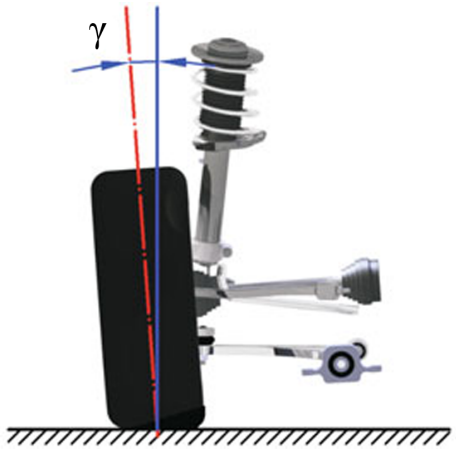


Abbildung 1.7.: Sturzwinkel γ, Quelle: [HE07]

Der Sturzwinkel ist als positiv definiert, wenn das Rad nach außen geneigt ist. Der Sturzwinkel beträgt in der Kostruktionslage $\gamma = -2$ bis $0\,°$. Negative Sturzwinkel verursachen Sturzseitenkräfte welche die Querführung der Achse positiv beeinflussen. Positive Sturzwinkel können Lenkungsflattern entgegenwirken, sollen aber wegen der damit verbundenen schlechten Seitenkraftübertragung vermieden werden, [HE07].

1.3.3. Lenkung

Lenkachse: Jene Achse, Achse EG in Abbildung 1.8, um die sich der Radträger beim Lenkvorgang dreht. Die Lenkachse ist in der Regel nach innen und hinten geneigt. Das führt zu hoher Fahrstabilität und bewirkt die Lenkungsrückstellung. Die Lenkachse sollte nah an der Radmittelebene liegen, damit Hebelarme von am Rad angreifenden Kräften

gering bleiben, [HE07].

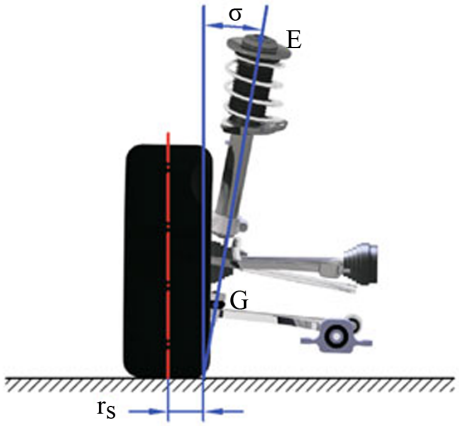


Abbildung 1.8.: Lenkachse, Quelle: [HE07]

Lenkachsenspreizung σ: Winkel zwischen einer zur Fahrbahn senkrecht liegenden Ebene und der Lenkachse, Abbildung 1.8. Die Spreizung ist als positiv definiert, wenn die Lenkachse nach innen geneigt ist. Sie bestimmt den Lenkrollradius und unterstützt damit die Lenkungsrückstellung.

Lenkrollradius r_s: Abstand zwischen der Schnittgeraden aus Fahrbahn mit Radmittelebene und dem Durchstoßpunkt der Lenkachse mit der Fahrbahn, Abbildung 1.8. Der Lenkrollradius ist als positiv definiert, wenn der Durchstoßpunkt der Lenkachse mit der Fahrbahn innerhalb der Radmittelebene liegt. Typische Werte in der Konstruktionslage sind $r_s = -20$ bis $+80$ mm. Der Lenkrollradius wird auf 0 mm ausgelegt, um den Einfluß von Regelsystemen (ABS) auf das Lenkmoment zu verringern. Bei μ-Split Bremsung erzeugt ein negativer Lenkrollradius ein Lenkmoment, das dem Giermoment entgegenwirkt, [HE07].

Nachlaufwinkel τ: Winkel zwischen der x-y-Ebene des Fahrzeugkoordinatensystems und der Lenkachse, 1.9.

Der Nachlaufwinkel ist als positiv definiert, wenn die Lenkachse nach hinten geneigt ist. Durch Nachlauf und Spreizung hebt sich der Aufbau bei der Lenkbewegung an. Dadurch wird ein Lenkungsrückstellmoment erzielt, [HE07].

Nachlauf n: Abstand des Durchstoßpunktes der Lenkachse mit der Fahrbahn in Fahrzeuglängsrichtung, Abbildung 1.9. Der Nachlauf ist als positiv definiert, wenn der Durchstoßpunkt der Lenkachse vor dem Radaufstandspunkt liegt. Der Nachlauf beeinflusst maßgeblich die Spurstabilität und die Lenkungsrückstellung, infolge am Reifen wirkender Seitenkräfte, [HE07].

Radlenkwinkel δ: Winkel zwischen der Schnittachse der Radmittelebene mit der Fahrbahn und der Fahrzeuglängsachse. Wird das Rad um den Winkel δ verdreht, entsteht durch den Reifenschräglauf eine Reifenseitenkraft, welche die Quersteuerung des Fahr-

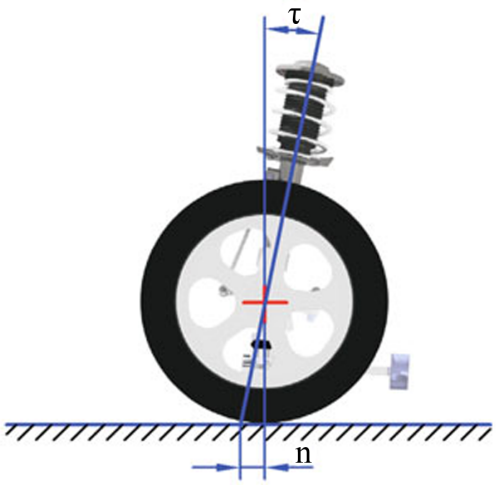


Abbildung 1.9.: Nachlaufwinkel und Nachlauf, Quelle: [HE07]

zeuges ermöglicht. Die Bewegungsrichtung des Radaufstandpunktes ist daher um den Winkel α zum Rad geneigt, Abbildung1.10. Die gelenkten Räder befinden sich, [Wol09]:

- an der Vorderachse (Standard)
- oder an allen vier Rädern (Allradlenkung)

Lenkgeometrie der Ackermann Lenkung

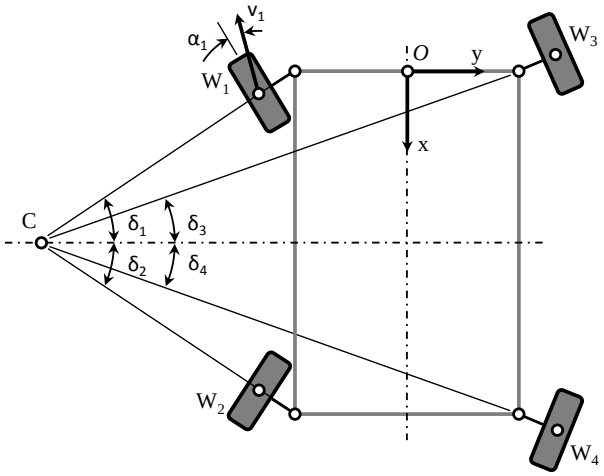


Abbildung 1.10.: Ackermann Geometrie einer Allrad Lenkung

Eine Ackermann Lenkung ist dadurch gekennzeichnet, dass bei geringen Geschwindigkeiten und Fahrt auf einem Kreisbogen alle Räder schräglauffrei rollen, [Wal06]. Das bedeutet, dass sich die Drehachsen aller Räder im Bahnmittelpunkt C treffen müssen, Abbildung 1.10, damit die Räder kräfte- und schlupffrei auf der Fahrbahn rollen. Als

Ackermannwinkel wird die Lenkwinkeldifferenz zweier Räder einer gelenkten Achse definiert:

$$\Delta\delta = \delta_1 - \delta_3 \tag{1.2}$$

Aus fahrdynamischen Gründen werden PKW oder Rennfahrzeuge meist nicht exakt nach Ackermann ausgelegt. LKW und Busse werden allerdings exakt nach Ackermann ausgelegt, um Verschleiß, Kraftstoffverbrauch und Kosten gering zu halten, [Wol09, S. 101 ff].

1.4. Motivation und Ziel der Arbeit

Ziel dieser Arbeit ist es, eine Methodik zu entwickeln, bei der in einer frühen Phase der Produktentwicklung Mehrkörpersysteme berechnet und optimiert werden können. Der Informationsfluss zwischen Konstruktions- und Berechnungsingenieur soll dabei möglichst unterstützt werden. Dies geschieht dadurch, dass Arbeitsabläufe zwischen Konstruktion und Berechnung bzw. Simulation effizient gestaltet werden.

Die Vorgehensweise soll sich nicht auf bestimmte Arten von Mechanismen, wie z.B. einzelne Typen von Radaufhängungen beschränken. Vielmehr ist die Methodik möglichst allgemein zu halten, um beliebige fahrzeugspezifische Mehrkörpersysteme, wie Lenkungen und Radaufhängungen, untersuchen und optimieren zu können.

Das entwickelte Programm ist anhand einer gängigen Radaufhängung, welche in einem vorangegangenen Projekt beschrieben wurde, zu validieren. Weiters ist eine atypische Allradlenkung eines Sonderfahrzeuges nach den Gesichtspunkten Reifenverschleiß und Kurventauglichkeit zu optimieren.

1.5. Gliederung der Arbeit

Nach der Einführung wird in Kapitel 2 die bestehende modulare Simulationssoftware MOVES^2 vorgestellt und der Bedarf eines neuen Moduls zur Optimierung von Mehrkörpersystemen, MuBOS, gezeigt. Danach wird das Modul MuBOS im Detail erläutert. Es wird die Struktur der Modelle in Catia V5® und Matlab/Simmechanics® sowie der Datentransfer zwischen Catia V5® und Matlab® beschrieben. Weiters wird gezeigt wie mit MuBOS im Sinne einer Co-Simulation zwischen Catia V5® und Matlab/Simulink® charakteristische Parameter eines Mehrkörpersystems optimiert werden können.

In Kapitel 3, werden zwei Anwendungsbeispiele von MuBOS beschrieben. Beim ersten Beispiel handelt es sich um die Optimierung charakteristischer Parameter einer Schwertlenker Radaufhängung nach dem Einbau eines Radnabenmotors. Im zweiten Beispiel wird das Lenksystem eines allradgelenkten Sonderfahrzeuges auf eine gute Kurventauglichkeit und geringen Radschlupf bei Kurvenfahrt ausgelegt. Dabei wird im Optimierungsprozess eine Lenkgeometrie nach Ackermann angestrebt.

In Kapitel 4 erfolgen Zusammenfassung und Ausblick der Arbeit.

2. Multibody Optimization System - MuBOS

In diesem Kapitel wird die Methodik von MuBOS (**M**ulti**B**ody **O**ptimization **S**ystem), welche in dieser Arbeit entwickelt wurde, vorgestellt. Danach wird der Aufbau des Programms mit seinen wichtigsten Funktionen, Simulations- Sensitivitäts- und Optimierungsfunktion, beschrieben. Mit der Simulationsfunktion lassen sich das kinematische Verhalten eines Mehrkörpersystems (MKS), beispielsweise einer Radaufhängung, berechnen und daraus charakteristische Parameter bestimmen.

Als charakteristische Parameter sind vom Benutzer festgelegte Parameter definiert. Diese beschreiben das Mehrkörpersystem in charakteristischer Weise.

Mit der Sensitivitätsanalyse wird die Empfindlichkeit charakteristischer Parameter auf eine Verschiebung einzelner Gelenkpunkte berechnet. In der Optimierungsfunktion wird die Geometrie eines Mehrkörpersystems so optimiert dass die charakteristischen Parameter einem erwünschten Verhalten möglichst nahe kommen.

Als erstes wird jedoch ein kurzer Überblick über MOVES^2 gegeben, da MuBOS als ein Modul dieser Simulationsumgebung konzipiert wurde.

2.1. MOVES^2

MOVES^2 (**MO**dular **VE**hicle **S**imulation System) wurde als flexible und erweiterbare Simulations- und Entwicklungsmethodik in Matlab / Simulink® entwickelt. Es besteht aus einem **G**raphical **U**ser **I**nterface (GUI), Fahrdynamikmodellen, [Roj12], unterschiedlicher Modellierungstiefe sowie den Modulen KOS (**K**inematic **O**ptimization **S**ystem), [Wil10], und dem hier vorgestellten Modul MuBOS, [Dür11, S. 7].

GUI Dieses Tool wurde mit Matlab Guide® entwickelt und dient dazu, die Abläufe von MOVES^2 zu steuern und hat mehrere Funktionen:

- Darstellung von Daten einer Matlab® Structure als Strukturbaum und in Diagrammform
- Datenverwaltung in Parameter files
- Aufruf der Main Functions des gewählten Moduls

In Abbildung 2.1 ist die GUI mit dem Bereich zur graphischen Ausgabe der Daten einer Matlab® Structure dargestellt.

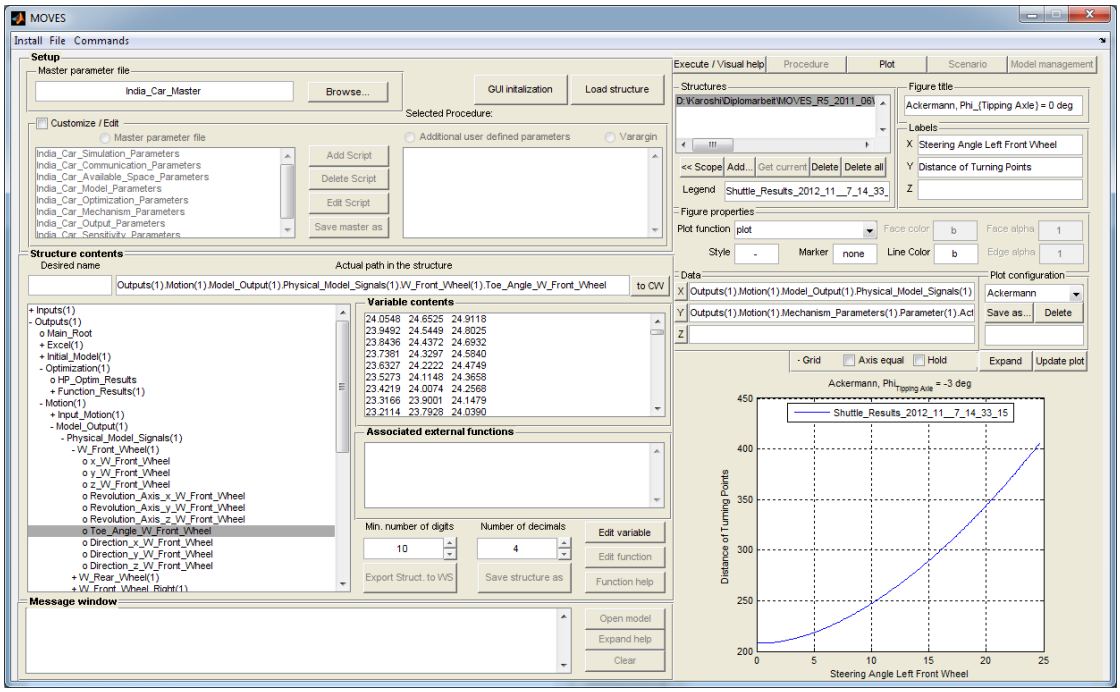


Abbildung 2.1.: GUI von MOVES[2]

Fahrdynamikmodelle Die Fahrdynamikmodelle umfassen das Fahrzeugmodell an sich, das Fahrermodell, Scenario-Builder sowie Auswertung und Visualisierung der Berechnungsergebnisse. Sie bilden die Fahrdynamik des Gesamtfahrzeuges ab. Die Modellierungstiefe reicht dabei vom einfachen Zwei-Massen-Schwinger für ein Viertelfahrzeugmodell über das Einspurmodell bis zum nichtlinearen Vollfahrzeugmodell. Die Fahrzeugmodelle sind für Hardware in the Loop Einsätze (HiL) gedacht, weswegen sehr auf Echtzeitfähigkeit geachtet wird. Außerdem sind sie adaptier- und erweiterbar. Somit können Module leicht ersetzt bzw. eingebaut werden. Kennfelder, die mit den Modulen KOS oder MuBOS erstellt wurden, können in MOVES[2] übernommen und zur Simulation der Fahrdynamik in Echtzeit verwendet werden, [Dür11, S. 7].

Kinematic Optimization System - KOS Das Programm KOS (Kinematic Optimization System) dient der kinematischen Analyse und Optimierung von Radaufhängungen. Es verfügt über eine Schnittstelle zwischen Matlab® und Catia V5®. Dabei fungiert Catia V5® als Server und Matlab® als Automation Client. Da eine direkte Kommunikation zwischen Matlab® und Catia V5® nicht möglich ist, wird, wie in Abbildung 2.2 dargestellt, die Visual Basic Application (VBA) von Excel als Bridge Connection

verwendet. Daten können von Matlab in Catia V5® gelesen und geschrieben werden. Somit ist eine Co-Simulation der beiden Programme möglich.

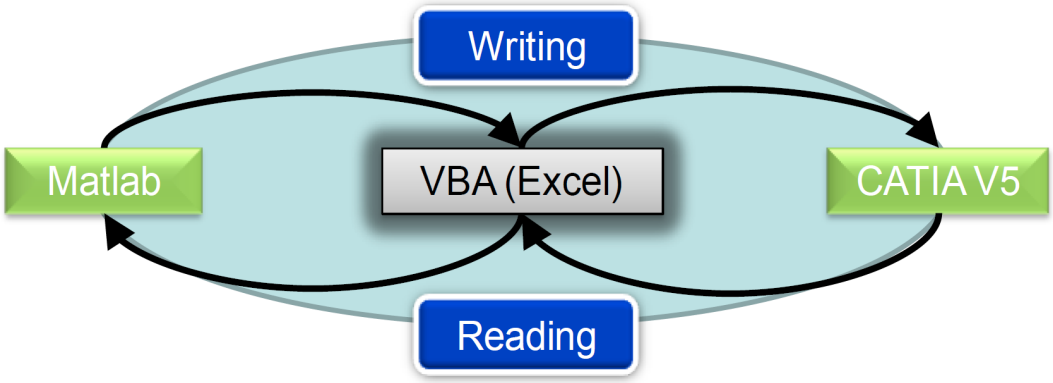


Abbildung 2.2.: Informationsfluss zwischen KOS und Catia V5®, Quelle: [Wil10]

Simulationen für Analyse und Optimierung einer Radaufhängung werden von KOS aufgerufen und in Catia V5® durchgeführt. KOS hat zwei Funktionen:

- Mit dem Analysetool wird das kinematische Verhalten, einschließlich der charakteristischen Parameter, siehe Abschnitt 2.3.3, einer bestimmten Radaufhängung, das in Catia V5® vorliegt, bestimmt.
- Das Optimierungstool verwendet das Analysetool als Subprozess in einem Optimierungsalgorithmus. Dieser verändert die bestehende Geometrie der Radaufhängung, um ein bestimmtes erwünschtes Verhalten eines charakteristischen Parameters zu erreichen.

Vorteilhaft ist bei dieser Vorgehensweise, dass die Kommunikation zwischen CAD System und Optimierungstool sehr schnell funktioniert, da nur ein CAD Modell existiert. Somit müssen Änderungen der Konstruktion nicht vom CAD Modell auf ein MKS Modell übertragen werden bevor das System analysiert und optimiert werden kann.

Während des Optimierungsprozesses werden in jeder Iteration die Positionen der Anlenkpunkte in Catia V5® neu geschrieben und die Aufhängung neu analysiert. Dies wirkt sich nachteilig auf die Rechenzeit aus. Da in Catia V5® nur kinematische Simulationen möglich sind, lassen sich keine dynamischen Bewegungen und Kräfte berechnen. KOS ist außerdem nur für eine beschränkte Anzahl von Einzelradaufhängungen konzipiert:

- McPherson Achse
- Mehrlenker Achse
- Schwertlenker Achse
- Doppel Querlenker Achse

Will man ein neues Konzept einer Radaufhängung, oder einen beliebig angeordneten Mechanismus berechnen, ist der Code des Programms selbst anzupassen, [Wil10] [Roj12].

2.2. Zielsetzung

In Kapitel 1.3 wurde der klassische Produktentwicklungsprozess dargestellt. In dieser Arbeit wird eine Methode vorgestellt, bei der in einer frühen Phase der Produktentwicklung, Zeit und Kosten gespart werden. Ziel ist es, Konstruktions- und Berechnungsingenieur stärker zu vernetzen. Dies geschieht dadurch, dass Arbeitsabläufe zwischen Konstruktion und Berechnung bzw. Simulation effizient gestaltet werden. Die Vorgehensweise beschränkt sich dabei nicht auf bestimmte Arten von Mechanismen, wie z.B. einzelne Typen von Radaufhängungen. Vielmehr ist es möglich beliebige fahrzeugspezifische Lenk- und Achskinematiken zu untersuchen und zu optimieren.

Im Sinne einer Co-Simulation zwischen Catia V5® und Matlab / Simulink® werden folgende Anforderungen an das Programm gestellt:

- Automatisierte Datenverwaltung: Vergleichbar zu KOS fungiert das CAD System als Datenbasis. Änderungen der Konstruktion werden automatisiert in Matlab / Simulink® übernommen.
- Berechnung: In einer Simulationsfunktion wird eine vom Benutzer definierte Bewegung simuliert. Alle berechneten Größen und die Werte der charakteristischen Parameter sollen automatisch gespeichert werden.
- Graphische Ausgabe: Alle Simulationsergebnisse können nach einer Simulation graphisch dargestellt werden. Für die charakteristischen Parameter werden automatisiert Diagramme erstellt.
- Sensitivitätsanalyse: Sie zeigt welche Geometrieelemente besonderen Einfluss auf die charakteristischen Parameter ausüben.
- Optimierungsfunktion: Dieser Algorithmus verändert die Geometrie des untersuchten Mehrkörpersystems so, dass die charakteristischen Parameter einem Zielwert bzw. einem Zielverhalten möglichst entsprechen.

Im Unterschied zu KOS erfolgt die Simulation des Mehrkörpersystems in Matlab / Simmechanics®. Dadurch lassen sich, unter Berücksichtigung von Masse und Trägheitstensoren der Körper, dynamische Simulationen durchführen und so Kräfte, Geschwindigkeiten und Beschleunigungen berechnen. Weiters sollen kürzere Rechenzeiten erzielt werden, da die Kommunikation zwischen Matlab® und Catia V5® in jedem Iterationsschritt entfällt.

2.3. Methodik

Da mit dem Programm KOS bereits eine Schnittstelle zwischen Matlab® und Catia V5® zur Verfügung steht, bietet es sich an, die Mehrköpersimulation mit Simmechancis durchzuführen. Damit lassen sich die Konstruktions- und Darstellungsmöglichkeiten von Catia V5® und die großen Berechnungsmöglichkeiten von Matlab / Simulink® nutzen und kombinieren. In dieser Arbeit werden mit Simmechanics nur kinematische Berechnungen durchgeführt. Es lassen sich aber auch dynamische MKS Berechnungen durchführen. Die Schnittstelle zwischen Matlab® und Catia V5® wird ohne große Veränderungen vom Programm KOS übernommen. Genauere Informationen finden sich dazu unter [Wil10].

Wie für die Programme MOVES[2] und KOS können alle Funktionen von MuBOS vom Graphical Benutzer Interface (GUI) aufgerufen werden. Main Functions können aber auch in Matlab® selbst ausgeführt werden. Sämtliche Daten werden in Parameter Dateien gespeichert und können mit der GUI als Matlab Structure dargestellt werden. In Abbildung 2.3 ist der Informationsfluss zwischen GUI, MuBOS und Catia V5® dargestellt.

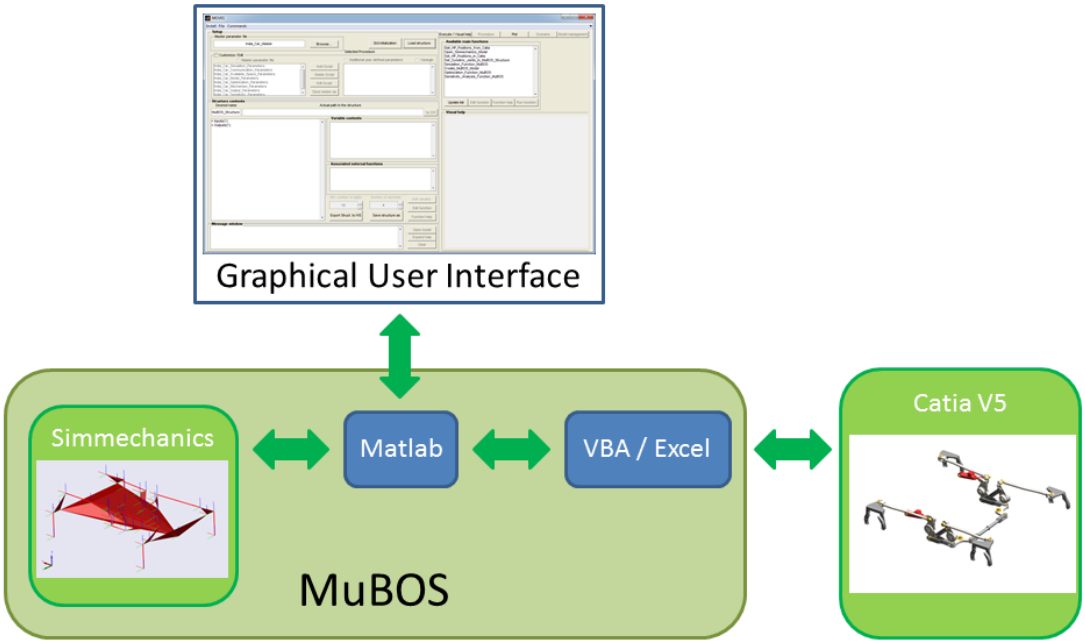


Abbildung 2.3.: Informationsfluss zwischen MuBOS und Catia V5®

In Abbildung 2.4 ist eine mögliche Vorgehensweise für eine Simulation oder einen Optimierungsprozess dargestellt. Diese Vorgehensweise ist nicht fest vorgeschrieben, vielmehr kann jede dargestellte Funktion auch einzeln ausgeführt und die Reihenfolge beliebig

kombiniert werden. Der dargestellte Prozess wird im Folgenden kurz zusammengefasst.

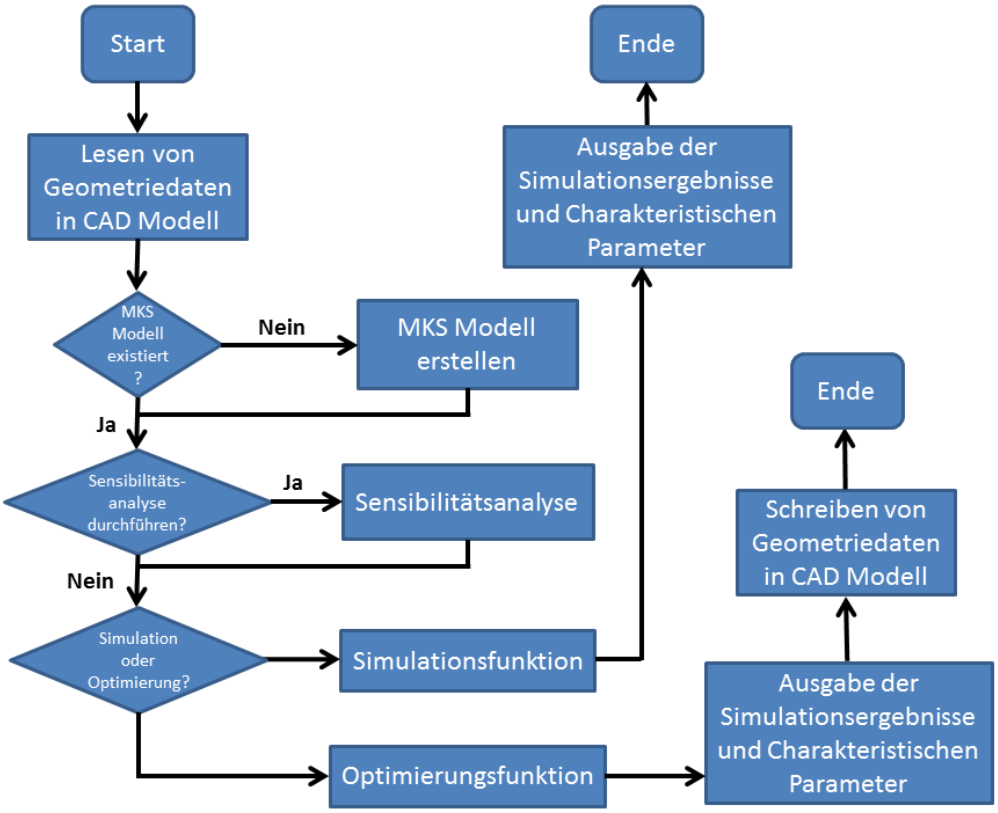


Abbildung 2.4.: Simulations- und Optimierungsvorgang MuBOS

In Catia V5® wird in einem Produkt ein Part als Skelett definiert. Dieses Skelett stellt ein vereinfachtes Modell des Mechanismus dar. Über die VBA / Excel Schnittstelle liest MuBOS die Geometriedaten des Skeletts und speichert sie in einer Parameter Datei.

Im nächsten Schritt wird ein MKS-Modell in Simmechanics erstellt. Dies ist für jede Anwendung nur einmal notwendig. Änderungen in der Geometrie, im Zuge des Konstruktionsprozesses, können jederzeit im Sinne einer Co-Simulation auf die Parameter Dateien übertragen werden. Gelenkpositionen werden vor jeder Simulation automatisch vom MKS-Modell übernommen.

Ist ein MKS-Modell bereits vorhanden, kann der Mechanismus sofort simuliert werden. Dabei kann der Benutzer charakteristische Parameter frei definieren und berechnen lassen. Diese werden von der Simulationsfunktion von MuBOS berechnet und mit einer Kostenfunktion bewertet.

Um ein Verständnis des untersuchten Mechanismus zu erlangen, kann eine Sensibilitätsanalyse durchgeführt werden. Dabei wird ermittelt, wie stark die Auswirkung einer

bestimmten Variation der Position eines Gelenkpunktes auf die charakteristischen Parameter ist.

In der Optimierungsfunktion wird Matlab® der Mechanismus als eine mehrdimensionale Funktion vorgegeben. In einer Kostenfunktion wird bewertet wie gut die charakteristischen Parameter einem erwünschten Verhalten entsprechen. Dabei wird nach dem Minimum der Kostenfunktion des Mechanismus gesucht.

Ist der Optimierungsprozess abgeschlossen und das Ergebnis zufriedenstellend, können die optimierten Positionen der Gelenkpunkte wieder in Catia V5® geschrieben werden. Damit steht eine optimierte Geometrie in Catia V5® zur Verfügung, die für die weitere Konstruktion verwendet werden kann.

MuBOS wird mit folgenden Versionen von Matlab® und Catia V5® entwickelt:

- Matlab: R2008b
- Catia: V5.16

Werden andere Versionen dieser Programme verwendet, kann unter Umständen unerwünschtes Verhalten auftreten.

2.3.1. Struktur von MuBOS

Zwischen MuBOS und Catia V5® müssen grundlegende Geometriedaten in Catia V5® gelesen, in einer MuBOS Parameter Datei gespeichert bzw. umgekehrt in Catia V5® geschrieben werden.

Dieser Datentransfer zwischen CAD System und MKS Simulationstool existiert bereits im Programm KOS und wird daher, mit kleineren Anpassungen, übernommen.

Die Kommunikation basiert auf dem OLE (Object Linking and Embedding) Automatisierungtool. Das ist ein Windows Protokoll welches es Programmen erlaubt Daten auszutauschen bzw. andere Programme zu steuern. Da praktisch alle Funktionen von Catia V5® automatisierbar sind, wird Catia V5® als Server und Matlab® als Client verwendet. Eine direkte Kommunikation zwischen den Programmen ist nicht möglich, da Matlab® nicht über die „Type Library (TLB)“ verfügt. Wie in Abbildung 2.3 dargestellt, wird die Kommunikation durch eine VBA Schnittstelle bewerkstelligt. Dabei sind die VB Functions in einer Excel Datei gespeichert. Diese Datei wird von MuBOS als Objekt geöffnet. Damit ist MuBOS in der Lage Daten mit Excel auszutauschen bzw. VB Funtions aufzurufen.

Auf dieselbe Art und Weise wird Catia V5® als Objekt einer VB Function geöffnet. Von den verschiedenen VB Functions können dann alle Catia V5® Befehle, die auch als Makro programmiert sind, ausgeführt werden. Detaillierte Informationen zu dieser Schnittstelle sind unter [Wil10] zu finden.

2.3.1.1. CAD Modell in Catia V5®

Das CAD System, in diesem Fall Catia V5®, dient MuBOS als Datenbasis. Insofern ist es notwendig festzulegen wie diese Daten strukturiert und aufgebaut sind. Auf der anderen Seite soll mit denselben Daten der Konstruktionsprozess stattfinden. Diesen soll MuBOS möglichst wenig einschränken. Um diese beiden Forderungen zu erfüllen wird im Catia V5® Hauptprodukt, welches das gesamte Mehrkörpersystem enthält, ein Catia V5® Teil als Skelett definiert und an erste Stelle im Strukturbaum gestellt. In diesem Teil wiederum wird ein geometrisches Set eingefügt das alle nötigen Geometriedaten enthält. In diesem geometrischen Set sind die Gelenkpunkte des Systems als Catia V5® Punkte definiert. Weitere Daten, die von MuBOS gelesen und geschrieben werden können, sind Catia V5® Parameter des Hauptprodukts.

Diese beiden Datentypen, Punkte und Parameter, reichen aus um die Co-Simulation zwischen Catia V5® und Matlab® durchzuführen. Dabei werden die Daten einerseits von MuBOS über die VBA Schnittstelle gelesen und geschrieben. Wurden in der Optimierungsfunktion optimierte Gelenkpositionen berechnet, können diese in Catia V5® geschrieben und in der Konstruktion verwendet werden.

Alle Namen des Produktes, des Parts, des geometrischen Sets und der Punkte sind frei wählbar.

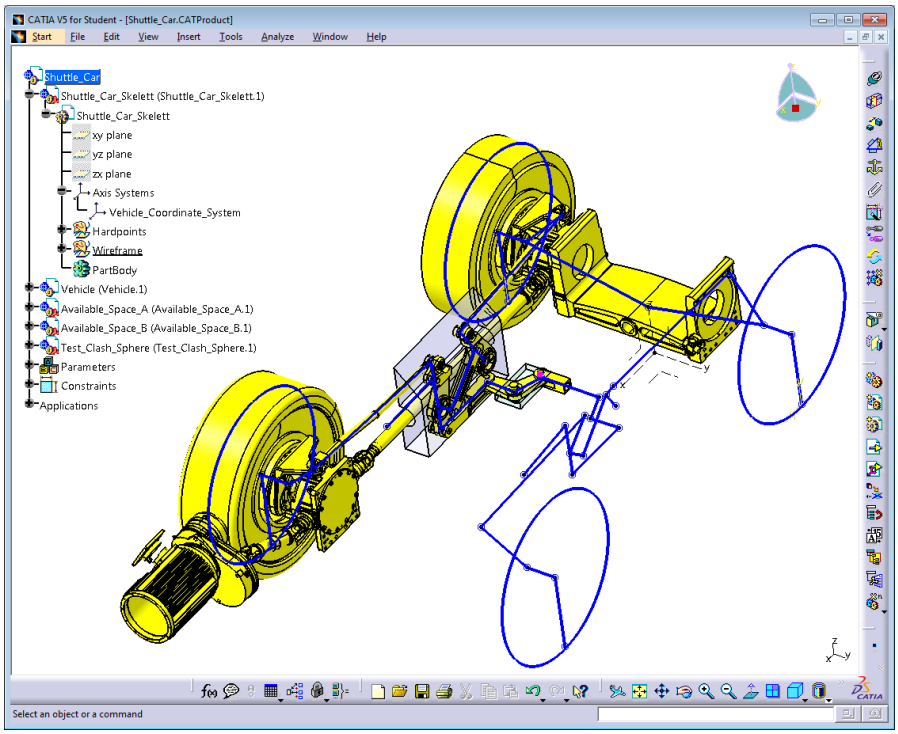


Abbildung 2.5.: CAD Modell in Catia V5®

Strukturbaum: In Abbildung 2.5 ist das CAD Modell am Beispiel des Anwendungsfalles in Abschnitt 3.2 dargestellt. Der Strukturbaum ist folgendermaßen aufgebaut:

```
Part Number des Produkts
├── Part Number des Skeletts
│   ├── xy plane
│   ├── yz plane
│   ├── zx plane
│   ├── Axis Systems
│   │   └── Vehicle_Coordinate_System
│   ├── Geometrisches Set Hardpoints
│   └── Geometrisches Set Wireframe
├── Beliebige Produkte und Teile der Konstruktion
├── Teile der Bauraumkontrolle
├── Test_Clash_Sphere
└── Catia Parameter
```

An erster Stelle steht das Skelett mit dem Fahrzeugkoordinatensystem und dem geometrischen Set „Hardpoints“, das die Gelenkpunkte enthält. Aus praktischen Gründen beinhaltet das Skelett ein weiteres geometrisches Set „Wireframe“. Darin befinden sich Linien und Skizzen, in blau, welche die Gelenkpunkte verbinden um das System anschaulich darzustellen.

Danach kann der Strukturbaum frei gestaltet werden. In Abblildung 2.5 beinhaltet das Produkt Vehicle die Konstruktionsdaten des Fahrzeugs. In Abbildung 2.5 sind die linke Hälfte der Lenkung und des Antriebsstranges sowie die Kippachse gelb dargestellt. Zwischen diesem Produkt und MuBOS findet kein Datenaustausch statt. Hier kann ungestört von der Simulation die Konstruktion durchgeführt werden. Es können allerdings Eltern/Kind Beziehungen zum Skelett bestehen. Im Sinne einer gemeinsamen Datenbasis für Konstruktion und Simulation ist es sinnvoll, dass Elemente des Skeletts immer Eltern anderer Elemente, und nicht umgekehrt, sind.

Teile der Bauraumkontrolle können ebenfalls beliebig im Strukturbaum platziert und benannt werden. Sie dürfen sich nur nicht in einem Unterprodukt befinden.

Test_Clash_Sphere ist ein Teil der automatisch während der Bauraumkontrolle, siehe Abschnitt 2.3.3, erzeugt wird.

Die Parameter, die vom MuBOS gelesen werden können, sind Parameter des Hauptprodukts. Sie können derzeit im MKS System, nicht aber im Optimierungsprozess, verwendet werden.

Einstellungen in Catia V5®: Damit MuBOS und Catia V5® überhaupt kommunizieren können, müssen folgende Einstellungen in Catia V5® getroffen werden:

- Tools / Customize / User Interface / Language: English

- Tools / Options / Infrastructure / Part Infrastructure / General / Update: Synchronize all external references when updating
- Tools / Options / Infrastructure / Product Structure / Tree Customization / Parameters: Yes

2.3.1.2. Programmablauf einer Main Function

Für jede Konfiguration eines Mehrkörpersystems wird in MuBOS eine Master Parameter Datei angelegt. Diese beinhaltet, wie in Matlab® Code 2.1 dargestellt, eine Liste von Parameter Dateien des Formats *.m. Alle Daten werden in diesen Parameter Dateien in Form einer Matlab® Structure abgelegt. Mit dieser Vorgehensweise lassen sich verschiedene Konfigurationen des Mehrkörpersystems darstellen, wobei eine Parameter Datei von mehreren Master Parmeter Dateien aufgerufen werden kann. Abbildung 2.6 zeigt die Auswahl der Master Parameter Datei der Schwertlenker Achse in der GUI.

Matlab Code 2.1: Master Parameter Datei des Sandvik Shuttle Car

```
% Paramater-files to be loaded

User_defined_Parameters = {
'India_Car_Simulation_Parameters';...
'India_Car_Communication_Parameters';...
'India_Car_Available_Space_Parameters';...
'India_Car_Model_Parameters';...
'India_Car_Optimization_Parameters';...
'India_Car_Mechanism_Parameters';...
'India_Car_Output_Parameters';...
'India_Car_Sensitivity_Parameters';...
};
```

Eine Main Function ist eine Matlab® Function die vom Benutzer über die GUI oder von der Kommandozeile aus aufgerufen wird. Sie initialisiert die MuBOS Structure und führt danach eine Reihe anderer Matlab® Functions aus.

Die Auswahl und Steuerung einer Main Function über die GUI ist in Abbildung 2.7 dargestellt. Der Button „Update list“ aktualisiert die Liste der Main Functions. Mit „Edit function“ wird die Main Function im Matlab® Editor geöffnet. Diese beiden Funktionalitäten sind vor allem bei der Entwicklung neuer Main Functions hilfreich. Wie im Matlab® Code 2.2 zu sehen, ist eine Main Function durch den auskommentierten Text „Main Function“ am Beginn der Datei definiert. Danach folgt, wieder auskommentiert, eine Beschreibung der Function. Diese kann in der GUI durch klicken des Buttons „Function help“ unter „Message window“ angezeigt werden. Mit „Run function“ wird die Main Function ausgeführt.

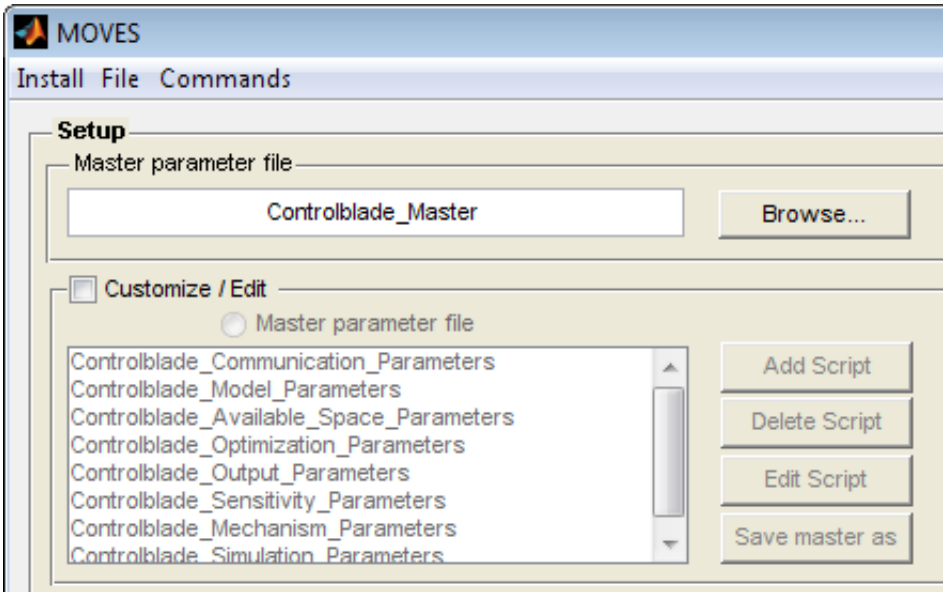


Abbildung 2.6.: Auswahl einer Parameter Datei in der GUI

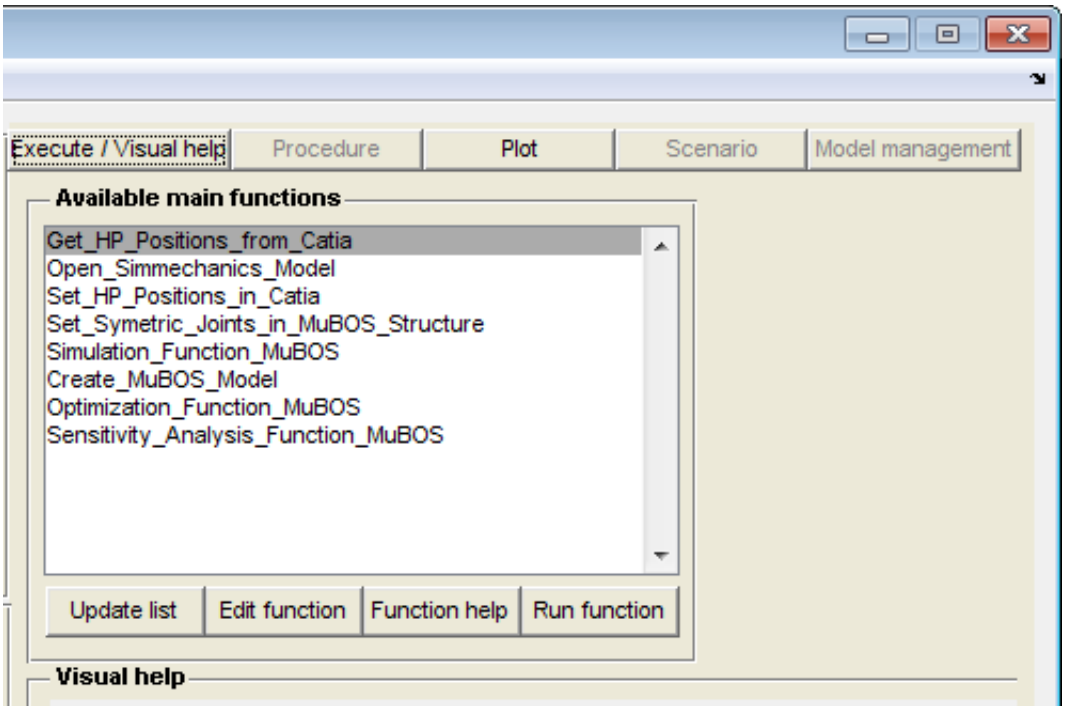


Abbildung 2.7.: Auswahl einer Main Function in der GUI

Matlab Code 2.2: MuBOS Main Function

```
function MuBOS_Structure = Get_HP_Positions_from_Catia(...
    Master_Param_Init_MuBOS,Add_User_defined_Parameters_MuBOS,varargin)
% Main function
% Function description
%
% This Function initializes the MuBOS_Structure, opens Excel and Catia and
% reads the coordinates of the joints defined in the Model_Parameter file.
% The coordinates are saved in the MuBOS_Structure and in the
% Model_Parameter file.
```

Jede Main Function hat einen festgelegten Ablauf. In der Preprocessing Phase werden die Parameter Dateien gelesen und die Daten im Workspace der Main Function als Structure initialisiert. In einigen Parameter Dateien sind Preprocessing Functions angegeben. Diese werden aufgerufen und geben grundlegende Größen und Daten, die ebenfalls in der Structure abgelegt werden, zurück. Danach folgt die eigentliche Berechnung. Hier werden z.B. Simulationen oder Optimierungsprozesse durchgeführt. Im Postprocessing werden die Berechnungsergebnisse in der Structure abgelegt. Danach wird die Structure im Ordner Current_Directory\MuBOS\Simulation_Results_MuBOS gespeichert. Pre- und Postprocessing sind für alle Main Functions gleich. Abbildung 2.8 zeigt ein Flussdiagramm des Ablaufs einer Main Function.

In jeder Main Function werden die Daten der Parameter Dateien als Structure initialisiert. Im Laufe der Berechnung werden Berechnungsergebnisse zur Structure hinzugefügt. Die Structure hat einen festgelegten Aufbau:

```
MuBOS_Structure
├── Inputs
│   └── Parameters
│       ├── PreProcessing
│       ├── Communication
│       ├── Model
│       ├── Available_Space
│       ├── Optimzation
│       ├── Outputs
│       ├── Sensitivity
│       ├── Mechanism_Parameters
│       └── Simulation
└── Outputs
    ├── Main_Root
    ├── Excel
    ├── Initial_Model
    ├── Optimization
    └── Motion
        └── Input_Motion
```

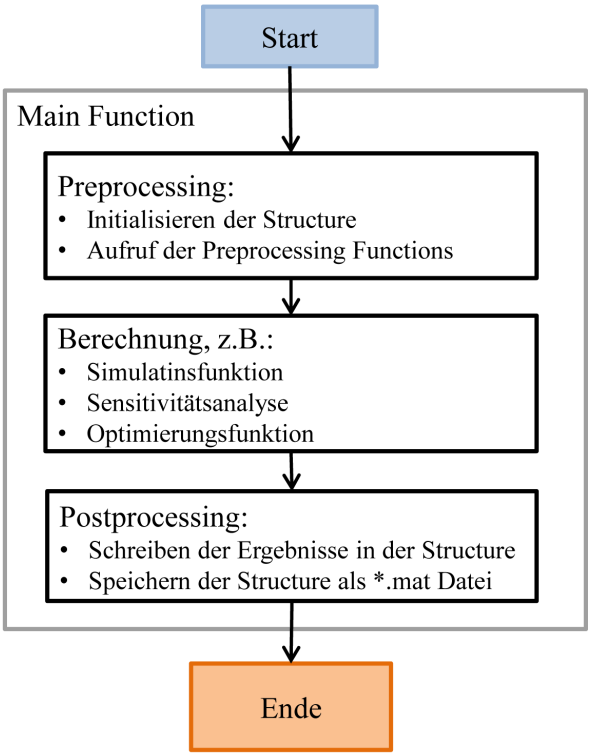


Abbildung 2.8.: Flussdiagramm einer Main Function

```
    Model_Output
        Physical_Model_Signals
        Char_Paramters
    Mechanism_Parameters
Simmechanics_Model_Info
Results_Directory
```

Unter MuBOS_Structure.Inputs.Parameter.PreProcessing werden Daten der Preprocessing Functions abgelegt. In den darauf folgenden Feldern werden die Daten der Parameter Dateien gespeichert.

Unter MuBOS_Structure.Outputs werden alle in den Main Functions berechneten Daten gespeichert:

- Main_Root: Beinhaltet den Dateipfad von MuBOS.
- Excel: Beinhaltet den Dateipfad zum COM Objekt der VBA Schnittstelle.
- Initial_Model: Wird die Optimization Function ausgeführt, werden hier die ursprünglichen Gelenkpositionen gespeichert.

- Model_Output: Hier werden alle Signale des Simmechanics Modells gespeichert.
- Mechanism_Parameters: Hier werden die Werte der charakteristischen Parameter abgelegt.
- Simmechanics_Model_Info: Liste aller Blöcke des Simmechanics Modells.
- Results_Directory: Verzeichnis in dem die gesamte Structure gespeichert wird.

2.3.1.3. Datenstruktur in Parameter Dateien

Jede Parameter Datei besteht aus zwei Bereichen. Den Kopf der Datei bildet die *ParameterFieldsList*. Sie beinhaltet allgemeine Informationen zur Datei. Der Parameter *Structure_Position* gibt die Position in der MuBOS Structure an, wo bei der Initialisierung die Daten eingefügt werden. *Structure_Fields* enthält eine Liste der Felder in der Structure der Parameter Datei. *Preprocessing_FCN* gibt den Namen der Preprocessing Function an, die bei der Initialisierung aufgerufen wird. Die *ParameterFieldsList* soll vom Benutzer nicht verändert werden. Der Matlab® Code 2.3 zeigt ein Beispiel.

Matlab Code 2.3: Parameter Field List

```
%% Parameter Fields List
Structure_Position = 'Parameters';      % <string> Field in MuBOS Structure
                                        % where data is inserted
Structure_Fields   = {'Optimization'    % <string> List of structure fields
                      };
PreProcessing_FCN  = 'Optimization_Parms_PreProcFCN_MuBos'; % <string> Name
                                        %of the parameter files
                                        % preprocessing function
```

Den zweiten Bereich bildet die Parameters Definition. Hier sind die Daten der Parameter Datei als Structure abgelegt. Im Folgenden werden die Parameter Dateien kurz beschrieben. Detailiert kommentierte Beispiele befinden sich im Anhang.

Communication_Parameter Datei: Diese Datei enthält Informationen welche zum Aufbau der VBA Schnittstelle zu Catia V5® benötigt werden:

- Name des Ordners in dem Ergebnisse gespeichert werden. Um ein Überschreiben alter Berechnungen zu verhindern setzt sich der Ordnername aus dem anzugebenden String und Datum und Uhrzeit zum Zeitpunkt des Speichervorganges zusammen.
- Pfad des Skelett-Teils im Catia V5® Strukturbaum.
- Pfad und Dateiname des Catia V5® Produkts.
- Flag ob Catia V5® und Excel sichtbar geöffnet werden sollen.
- Pfad und Dateiname der Excel Datei, welche die VBA Scripts enthält.

Ein Beispiel ist unter Matlab® Code A.1 zu finden.

Model Parameter: Hier werden Daten des Mehrkörpersystems selbst abgelegt. Dabei werden drei Datentypen unterschieden:

- Parameter: Catia V5® Parameter werden von der Get_HP_Positions_from_Catia Main Function im CAD Modell gelesen und können im MKS Modell verwendet werden. Diese Parameter werden im Optimierungsprozess nicht verändert. Es müssen Name, Wert und Pfad im Catia V5® Strukturbaum angegeben werden.
- Gelenke (Joint): Um mit MuBOS überhaupt Simulationen durchführen zu können müssen Gelenke definiert werden. Dabei sind verschiedene Größen, die den Optimierungsprozess, die Bauraumkontrolle und die Symmetriefunktion betreffen, zu definieren.
- Körper (Body): Diese Daten werden nur benötigt wenn die Main Function Create_MuBOS_Model, in der das MKS Modell automatisiert erstellt wird, ausgeführt wird. Nähreres dazu ist im Abschnitt 2.3.2 zu finden.

Details und Kommentare sind im Matlab® Code A.2 zu finden.

Simulation Parameter: Alle für die Simulation in Simmechanics nötigen Einstellungen werden in dieser Datei getroffen. Im wesentlichen sind das Einstellungen die direkt in Simmechanics getroffen werden, wie z.B. Simulationszeit, Ausgabeschrittweite, Solver und Analysis Mode. Weiters kann angegeben werden ob die Simulation in Simmechanics graphisch dargestellt oder ob die Gelenkpositionen in jeder Iteration in Catia V5® aktualisiert werden sollen. Als letztes werden Start- und Endwert der gesteuerten Bewegung, z.B. Lenkwinkel oder Radhub, definiert.

Details und Kommentare sind im Matlab® Code A.3 zu finden.

Mechanism Parameter: Die charakteristischen Parameter des MKS Systems werden in der Optimierungsfunktion bewertet und sollen möglichst einem Zielverhalten entsprechen. In dieser Datei werden daher Informationen zur Gewichtung, zum Zielverhalten und zur graphischen Ausgabe gespeichert.

Details und Kommentare sind im Matlab® Code A.4 und im Abschnitt 2.3.3 zu finden.

Sensitivity Parameter: Hier werden Einstellungen für die Sensitivitätsanalyse getroffen. Der Benutzer kann angeben, ob die Gelenkpunkte in einem Bereich konstanter Größe variiert werden oder in einem Bereich relativ zum erlaubten Bauraum.

Details und Kommentare sind im Matlab® Code A.5, sowie in Abschnitt 2.3.4 zu finden.

Available Space Parameter: Hier werden Einstellungen zur Bauraumkontrolle getroffen. Da diese Funktion von KOS übernommen wurde braucht der Benutzer hier keine Änderungen vornehmen.

Details und Kommentare sind im Matlab® Code A.6 zu finden.

Optimzation Parameter: Hier werden alle für den Optimierungsprozess nötigen Einstellungen getroffen. Das sind neben dem Strafwert für Überschreiten des Bauraumes

und der Ausgabe von Zwischenergebnissen die Abbruchkriterien:

- Maximale Anzahl an Simulationen $Max\_Function\_Evals$
- Toleranzwert der Kostenfunktion $TolFun$
- Toleranzwert der optimierten Parameter (Gelenkpositionen) $TolX$

Ist eines dieser Kriterien erfüllt, wird die Optimierungsfunktion abgebrochen.

Details und Kommentare sind im Matlab® Code A.7 und in Abschitt 2.3.5 zu finden.

Output Parameter: In dieser Datei kann der Benutzer die Darstellung aller automatisch erstellten Diagramme festlegen. Es können Schriftgrößen, Linienstärken, Farben und Farbpaletten festgelegt werden.

Details und Kommentare sind im Matlab® Code A.8 zu finden.

2.3.1.4. Struktur des Simmechanics Modells

Das MKS Modell in Simmechanics kann vom Benutzer beliebig aufgebaut werden. Nur zum Aufbau der Ausgangssignale bestehen Einschränkungen. Natürlich ist es zweckmäßig eine einheitliche Struktur innerhalb des MKS Modells zu verwenden. Daher wird hier das Template, das bei der automatisierten Modellerstellung verwendet wird, beschrieben.

Abbildung 2.9 zeigt die drei Blöcke Input, Physical Model und Postprocessing, aus denen das Modell aufgebaut ist.

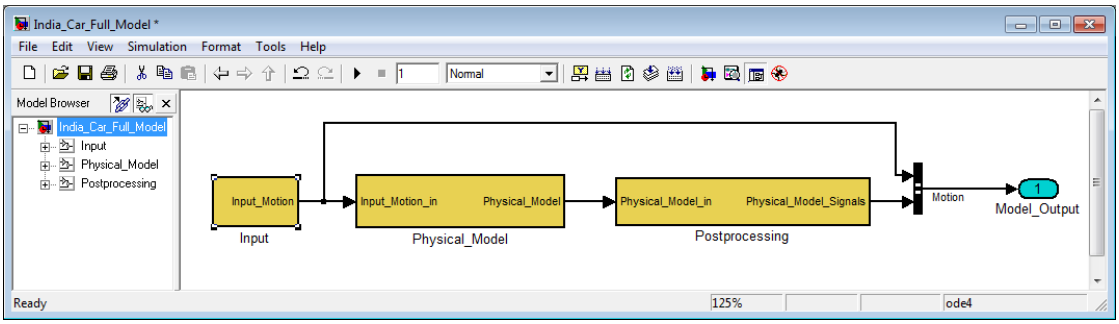


Abbildung 2.9.: MKS Modell in Simmechanics

Input: In diesem Block werden die Signale zur Steuerung der Bewegung erstellt. Ein Aktuator Block benötigt ein Bus Signal, das aus den Größen Position, Geschwindigkeit, und Beschleunigung, besteht. In der MuBOS Structure stehen in folgenden Feldern Daten zur Steuerung der Bewegung zur Verfügung:

- Erste Bewegungsrichtung:
  - Geschwindigkeit:
    MuBOS_Structure.Inputs.Parameters.Simulation.Main_Movement.Speed

  - Anfangsbedingung:
    MuBOS_Structure.Inputs.Parameters.Simulation.Main_Movement.Start

- Zweite Bewegungsrichtung:
  - Position:
    MuBOS_Structure.Inputs.Parameters.Simulation.Movement_Position

Grundsätzlich kann im Simmechanics Modell jede Größe der MuBOS Structure aus dem Matlab® Workspace gelesen werden. Abbildung 2.10 zeigt, wie diese Größen zur Bewegungsdefinition in Simmechanics verwendet werden. Die Geschwindigkeit wird mit einem Constant-Block angegeben. Die Position wird in einem Integrator-Block berechnet, wobei die Anfangsbedingung unter „Initial condition“ angegeben wird. Die Beschleunigung wird in einem Derivate-Block berechnet.

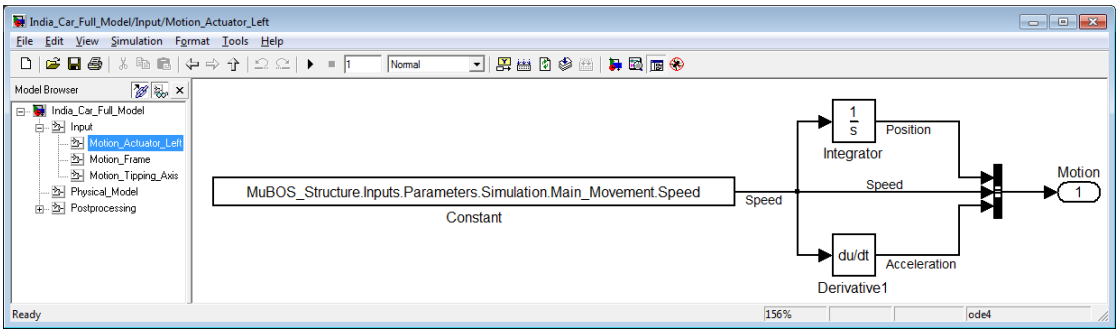


Abbildung 2.10.: Aufbau des Main Movement Signals

Die Bewegungsdefinition für die zweite Bewegungsrichtung erfolgt, wie in Abbildung 2.11 dargestellt, in ähnlicher Art und Weise. Die Geschwindigkeit wird im Constant-Block mit 0 vorgegeben. Im Integrator-Block wird unter „Initial condition“ die aktuelle Position angegeben. Die Beschleunigung wird in einem Derivate-Block berechnet.

Die Signale aller Bewegungen werden in einem Bus-Creator Block gebündelt und dem Physical_Model-Block übergeben.

Physical Model:

Im Physical_Model-Block werden die Signale der Bewegung über Source-Blöcke und Bus-Selector-Blöcke an die Actuator-Blöcke weitergeleitet. Es können beliebig viele Joint- und Body-Sensor-Blöcke eingebaut werden. Die Sensor Signale werden praktischerweise mit Goto- und From-Blöcken an einen Bus-Creator-Block weitergeleitet. Dort werden alle Signale gebündelt über einen Outport-Block an den Postprocessing-Block weitergeleitet. Abbildungen 2.12 und 2.13 zeigen Beispiele.

In den Body-Blöcken werden Angaben zu den Simmechanics Coordinate Systems gemacht. Abbildung 2.14 zeigt die Eingabemaske eines Body-Blocks. Die Gelenkpositionen werden auch hier aus dem Matlab® Workspace gelesen. Jedes Coordinate System muss sich auf das selbe Koordinatensystem wie im CAD Modell beziehen. In den in

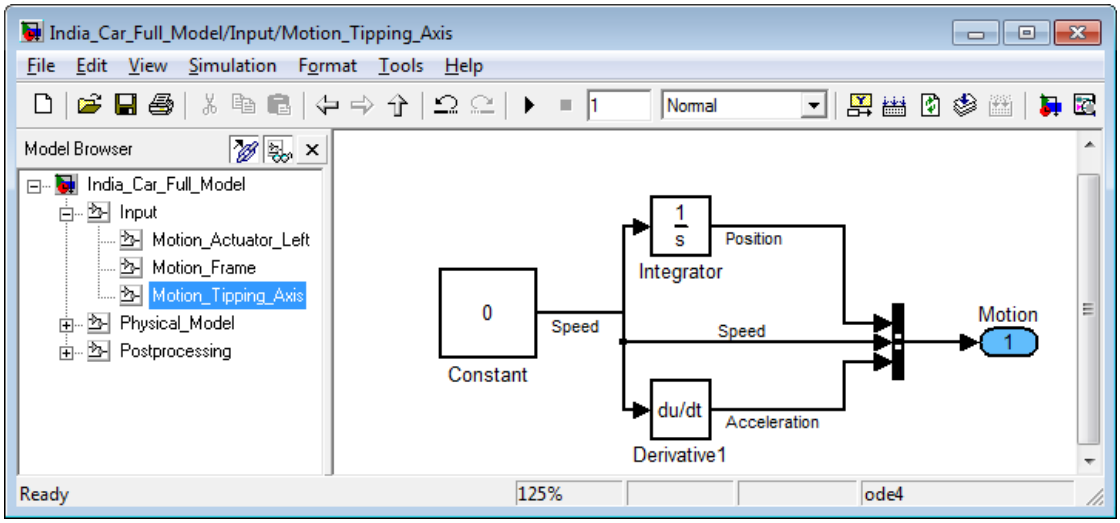


Abbildung 2.11.: Aufbau des Movement Signals

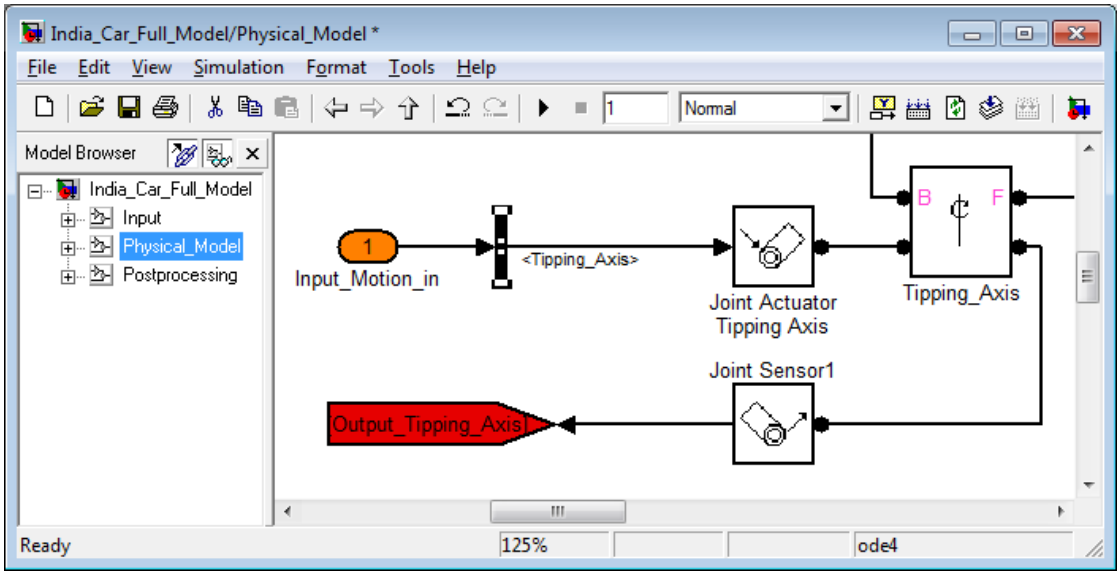


Abbildung 2.12.: Eingangssignale im Physical_Model-Block

Abschnitt 3 vorgestellten Anwendugsfällen sind alle Gelenkpunkte einheitlich auf das Fahrzeugkoordinatensystem in Simmechanics „World“ angegeben.

Postprocessing:

Der Postprocessing-Block dient dazu, die Simulink Signale so zu ordnen dass sie von MuBOS ausgewertet werden können. In Abbildung 2.15 ist der Aufbau des Postprocessing-Blockes dargestellt. Er besteht aus zwei Blöcken:

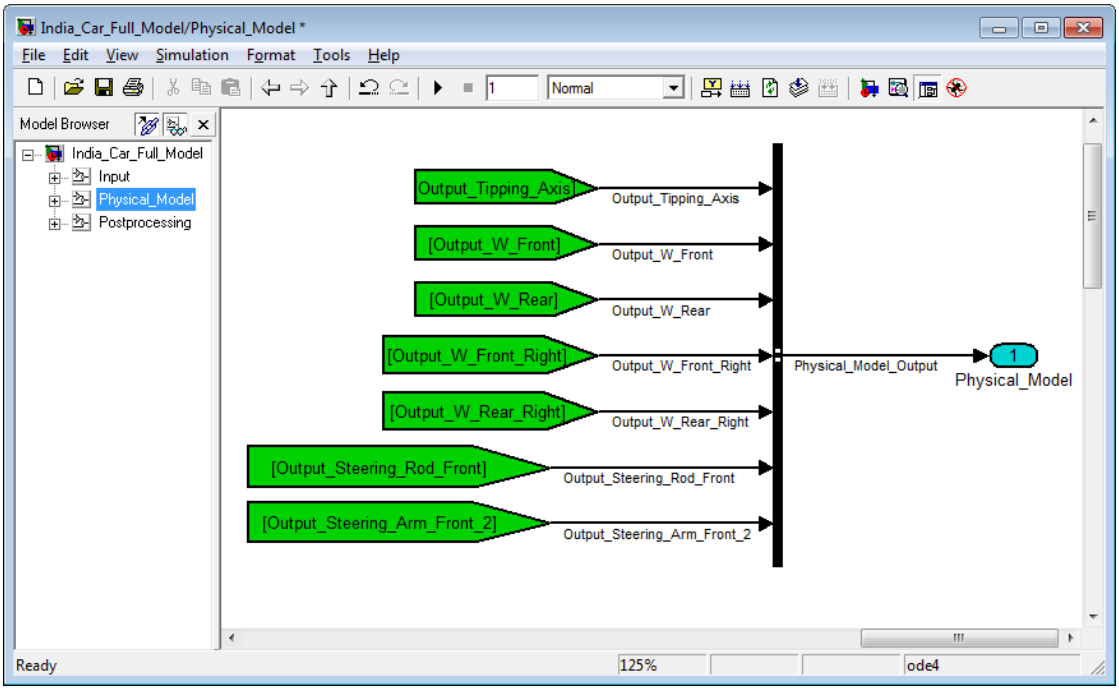


Abbildung 2.13.: Ausgangssignale im Physical_Model-Block

- Coordinate_Transformation: Hier werden alle Signale mit Bus-Selector- oder Demux-Blöcken entbündelt und benannt. Es können beispielsweise Koordinatentransformationen oder Umrechnungen der Einheiten durchgeführt werden. Danach werden alle Signale wieder mit Bus-Creator Blöcken zu Bus-Signalen zusammengefasst.
- Char_Parameter_Calculation: Hier besteht die Möglichkeit noch im Simmechanics Modell einfache charakteristische Parameter, siehe Abschnitt 2.3.3 zu berechnen.

MuBOS schreibt keinen bestimmten Aufbau der Ausgangssignale vor. Nach jeder Simulation wird das Bus-Signal des Outport-Blockes analysiert und die Signale mit demselben Aufbau in der MuBOS Structure abgelegt. Dabei werden die Bus-Signale, ausgehend vom Outport-Block, so lange weiterverfolgt bis ein einzelnes Signal oder ein Operator erreicht wird. Abbildungen 2.16 und 2.17 zeigen dieselben Signale in Simulink und in der MuBOS Structure. Die Simulationsergebnisse werden in der MuBOS Structure im Feld Output.Motion.Model_Output eingefügt.

Damit MuBOS die Ausgangssignale in der MuBOS Structure ablegen kann müssen folgende Punkte erfüllt sein:

- Die Signale müssen vollständig mit Bus-Selector- oder Demux- Blöcken entbündelt werden.
- Danach müssen die Signale zu Bus-Signalen zusammengefasst werden.

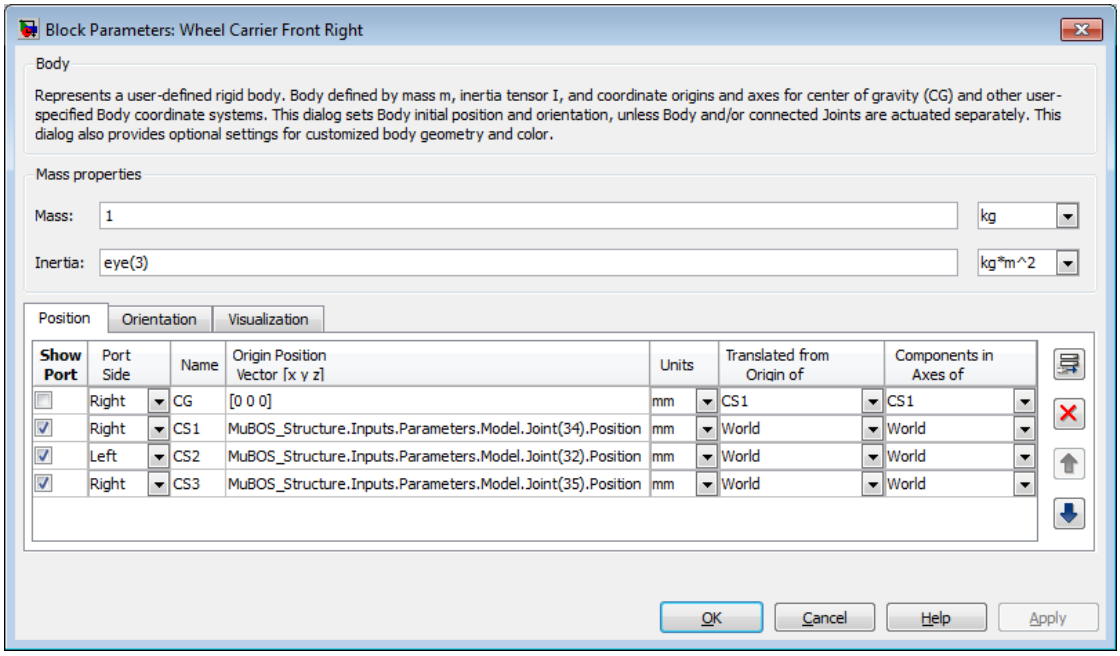


Abbildung 2.14.: Gelenkpositionen im Body-Block

- Jedes Signal und Bus-Signal muss an einer Stelle benannt werden um in der MuBOS Structure abgebildet zu werden. Wird ein Signal oder Bus-Signal an mehreren Stellen benannt, erkennt MuBOS einen Fehler und beendet die laufende Main Function.

Unter Beachtung dieser Punkte kann der Benutzer den Aufbau der Ausgangssignale und damit den Aufbau der Simulationsergebnisse in der MuBOS Structure frei bestimmen.

2.3.2. MKS Modellerstellung

Für jede Anwendung die mit MuBOS untersucht wird, muss ein MKS-Modell in Simmechanics erstellt werden. Dieses kann vom Benutzer zur Gänze selbst erstellt werden, wenn die in Abschnitt 2.3.1.4 erwähnten Regeln für die Signalstruktur eingehalten werden und die Positionen der Coordinate Systems in den Joint-Blöcken richtig definiert werden.

Um Zeit zu sparen und lästiges Fehlersuchen zu vermeiden, wird der Benutzer durch die Main Function Create_MuBOS_Model unterstützt. Dabei wird ein MKS Modell automatisiert erstellt. Allerdings wird das Modell aus zwei Gründen nicht vollständig erstellt:

- Mit dem Befehl *set_param* werden, in Matlab 2008b®, Simulink Block Parameter von der Kommandozeile aus gesteuert. Ausgenommen sind davon Simmechanics Blöcke. De facto funktioniert dieser Befehl, allerdings nur sehr eingeschränkt, [MAT08].

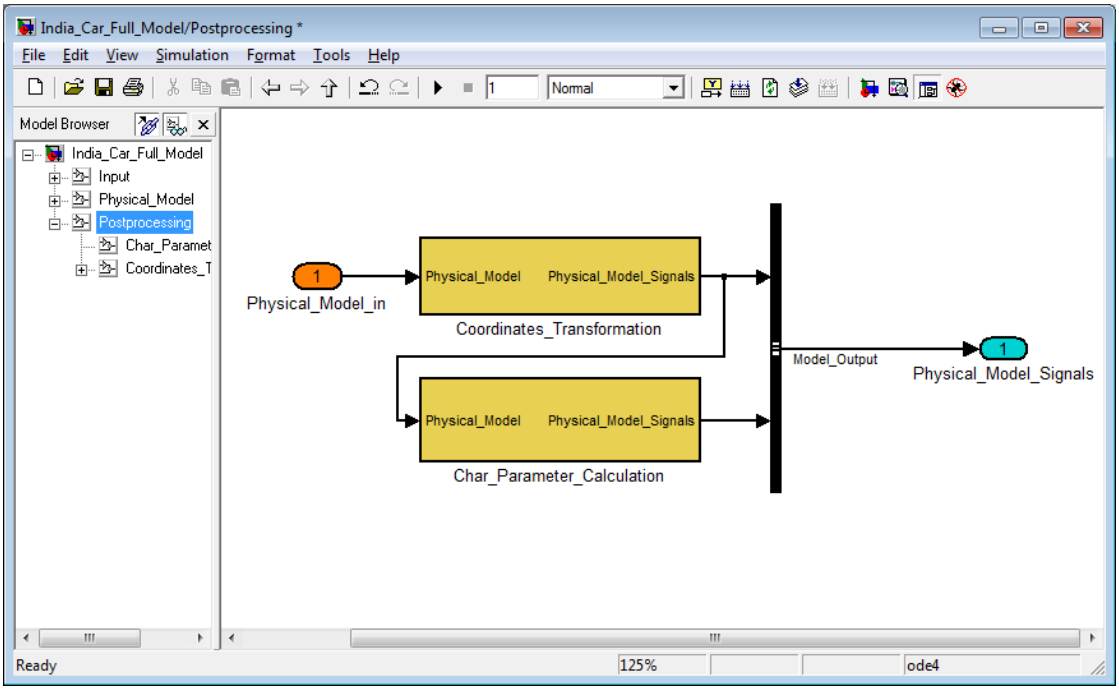


Abbildung 2.15.: Postprocessing-Block

- Dem Benutzer soll möglichst viel Freiheit in der Gestaltung des Mehrkörpersystems und der Ausgangssignale gelassen werden.

Daher beschränkt sich diese Funktion darauf, die in der Model_Parameter Datei definierten Bodies und Joints im Physical_Model-Block des MKS Modells einzufügen.

Die Funktion hat folgenden Ablauf: Nach der üblichen Preprocessing Phase wird das, in der Simulation_Parameter Datei definierte MKS Modell geöffnet. Ist diese Datei nicht vorhanden, wird ein Template geöffnet. Im nächsten Schritt werden alle Joint-Blöcke eingefügt. Danach werden die Body-Blöcke eingefügt. Alle Einstellungen der Joint- und Body-Blöcke, außer der Sichtbarkeit der Ports, werden wie in der Model_Parameter Datei angegeben, richtig getroffen. In Abbildung 2.18 ist der Prozess als Flussdiagramm dargestellt. Joint- und Body-Blöcke werden, wie in Abbilung 2.19 dargestellt, gleichmäßig verteilt platziert.

Dem Benutzer wird damit ein großer Teil der Arbeit abgenommen. Folgende Punkte müssen vom Benutzer selbst durchgeführt werden:

- Input-Block: Die Bewegungs-Signale müssen vollständig erstellt werden.
- Physical_Model-Block:
  - Alle Ports müssen sichtbar gemacht werden.
  - Joint- und Body-Blöcke müssen vebunden werden.

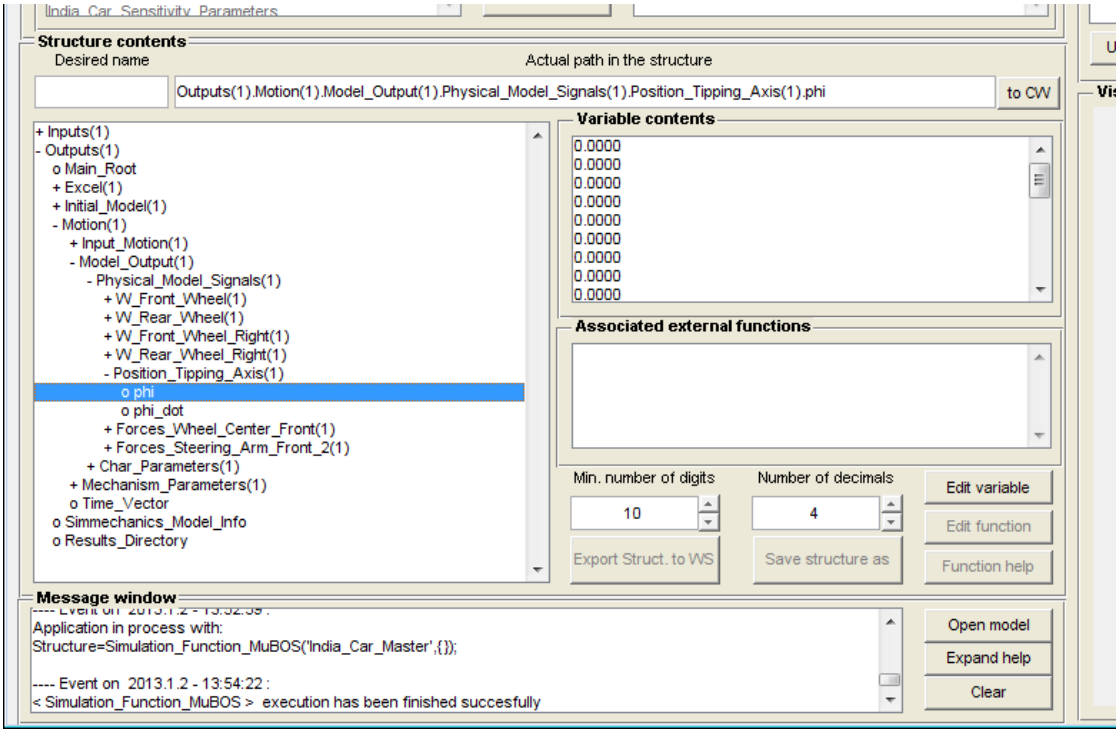


Abbildung 2.16.: Signalstruktur in MuBOS Structure

  - Ground- und Machine-Environment-Blöcke sind einzufügen.
  - Sensoren und Sensorsignale bis zum Outport-Block sind einzufügen.
- Postprocessing: Die Ausgangssignale sind wie in Abschnitt 2.3.1.4 beschrieben, aufzubauen.
- Darüber hinaus kann das MKS Modell natürlich noch beliebig beliebig erweitert werden.

2.3.3. Simulationsfunktion

Wurde ein MKS Modell in Simmechanics erstellt, kann mit der Simulationsfunktion das kinematische Verhalten des untersuchten Mechanismus berechnet werden. In Abbildung 2.20 ist der Simulationsprozess dargestellt. Modelleingang sind im wesentlichen die aktuellen Positionen der Gelenkpunkte, da die Konfiguration des Mechanismus ja schon im MKS Modell hinterlegt ist. Da die Simulationsfunktion ein Subprozess der Optimierungsfunktion ist, laufen zuerst die Symmetriefunktion und Bauraumkontrolle ab. Danach werden die Positionen der Gelenkpunkte im MKS Modell gespeichert und die Simulation durchgeführt. Nach erfolgreicher Simulation werden die charakteristischen Parameter und daraus die Kostenwerte berechnet. Diese werden in der Optimierungsfunktion, Ab-

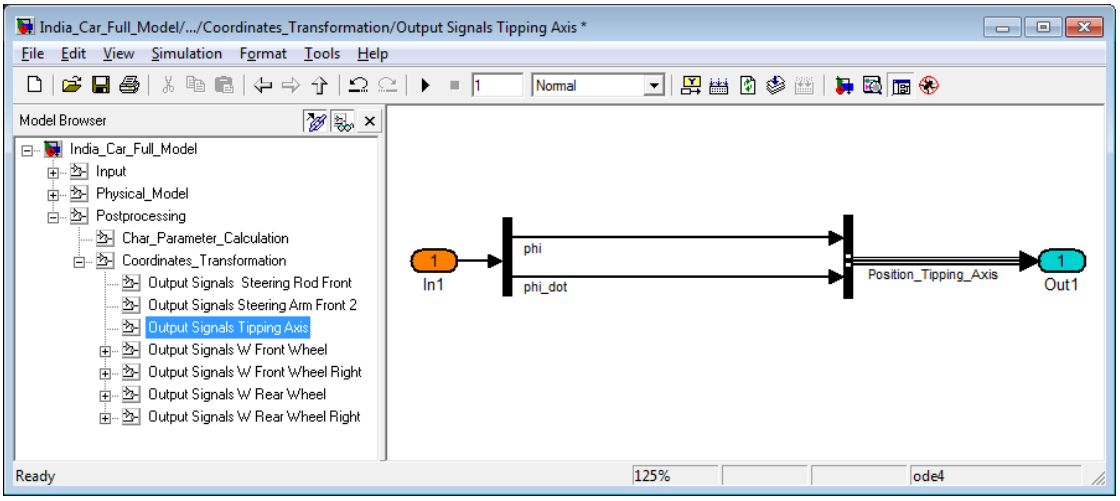


Abbildung 2.17.: Entbündelte Signale

schitt 2.3.5, benötigt. Wird die Simulation abgebrochen, wird als Fitnesswert ein vorher definierter Strafwert ausgegeben. Grund für einen Abbruch der Simulation können etwa kinematische Probleme, wie Strecklagen, Anschläge etc., sein. Im letzten Schritt werden alle Berechnungsergebnisse in einer Matlab® Structure und der Plot mit den charakteristischen Parametern in einer Matlab® Figure gespeichert.

Definition von Bewegungen

Die Zahl der Freiheitsgrade f bestimmt, wie viele unabhängige Bewegungen ein System ausführen kann, Gleichung 1.1.

Nimmt man im Fall einer Radaufhängung, alle Lagerstellen als starr an, so sind die Freiheitsgrade

- $f = 0$: Starre Radaufhängung
- $f = 1$: Gefederte und ungelenkte Radaufhängung
- $f = 2$: Gefederte und gelenkte Radaufhängung [Skript Hirschberg S. 64]

In MuBOS kann der Benutzer zwei Bewegungen definieren. Dies geschieht durch Angabe von Start- und Endwert eines Parameters. Im Anwendungsbeispiel Shuttle Car sind das:

- Position der Kippachse, angegeben durch den Drehwinkel
- Position des Hydraulikzylinders, angegeben durch die Verschiebung aus der Konstruktionslage

Diese Bewegungen werden gleichzeitig und mit konstanter Geschwindigkeit ausgeführt. Wird zusätzlich zu Start- und Endwert einer Bewegung auch eine Schrittweite angegeben,

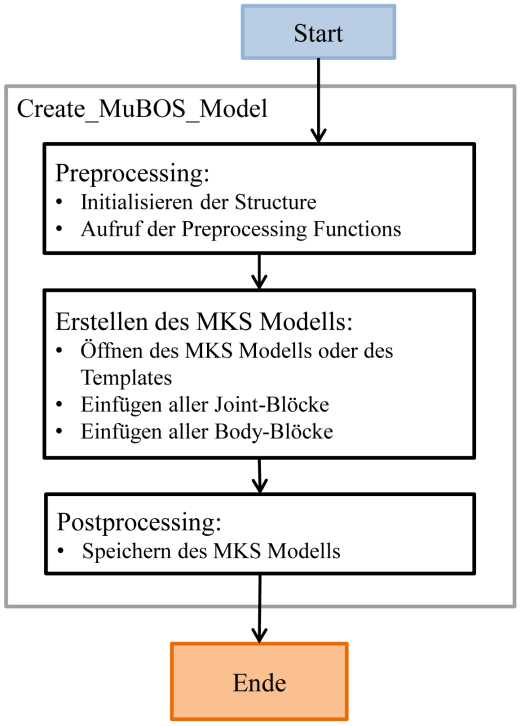


Abbildung 2.18.: Flussdiagramm der Create_MuBOS_Model Main Function

wird die Simulation iterativ für jede Stellung des einen Parameters durchgeführt. Somit lassen sich dreidimensionale Kennfelder für einen charakteristischen Parameter erstellen. In Abbildung 3.21 ist am Beispiel des Shuttle Cars die Lenkwinkeldifferenz zwischen Vorder- und Hinterrad, in Abhängigkeit der Kippachsenstellung und der Position des Hydraulikzylinders, dargestellt.

Charakteristische Parameter

Der Benutzer hat die Möglichkeit charakteristische Parameter zu definieren. Zur Verfügung stehen alle Berechnungsergebnisse aus der Simulation. Dementsprechend muss auch der Benutzer darauf achten dass im MKS Modell die benötigten Daten ermittelt werden. Diese charakteristischen Parameter werden nach der Simulation des MKS Modells ermittelt und graphisch dargestellt. Abbildung 3.2 zeigt die charakteristischen Parameter im Anwendungsbeispiel Schwertlenkerachse. Abildungen 3.21 und 3.2.5 zeigen charakteristische Parameter im Anwendungsbeispiel Lenkungsoptimierung.

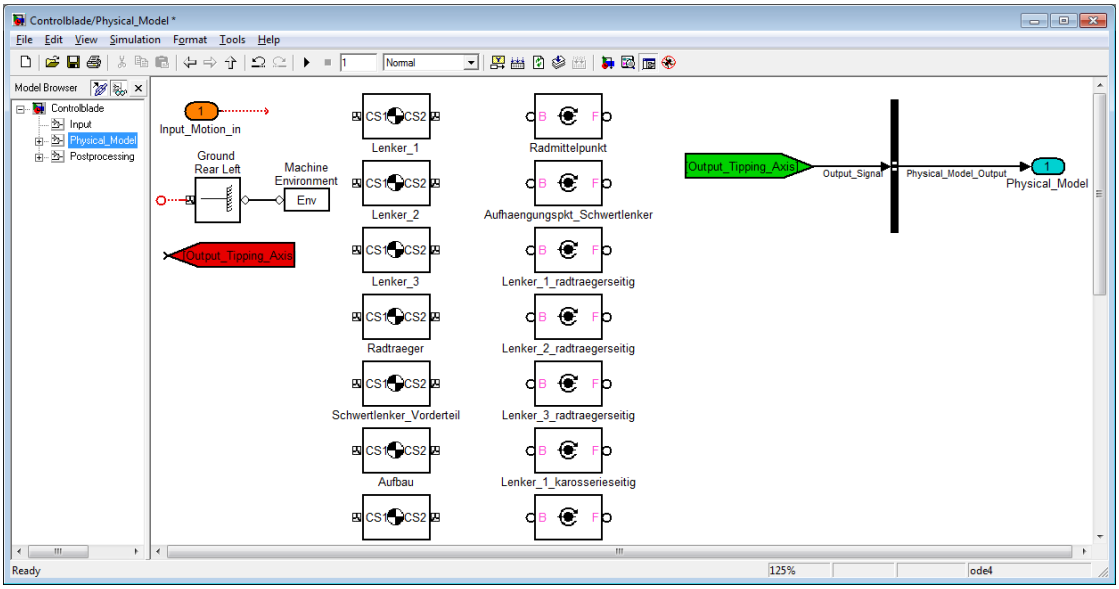


Abbildung 2.19.: Eingefügte Joint- und Body-Blöcke

Kostenfunktion

Um charakteristische Parameter zu bewerten, wird die Kostenfunktion J definiert, [Roj12]:

$$J = \begin{cases} \sum_{i=1}^{n} \left(w_{om,i} \cdot w_i \cdot \sum_{k=2}^{m} \frac{\Delta P_{ik} + \Delta P_{i(k-1)}}{2} \cdot \Delta\zeta \right) & \text{... falls Bauraumbedingung erfüllt,} \\ J_{Kollision} & \text{... falls nicht erfüllt.} \end{cases} \tag{2.1}$$

Dabei ist n die Anzahl der charakteristischen Parameter, m die Simulationszeit dividiert durch die Ausgabeschrittweite, $w_{om,i}$ der Gewichtungsfaktor für die Größenordnung eines Parmeters, w_i der Gewichtungsfaktor für einen Parameter, wobei

$$\sum_{i=1}^{n} w_i = 1$$

gilt. Die Kostenfunktion stellt, wie in Abbildung 2.21 dargestellt, die Fläche zwischen tatsächlichem und erwünschtem Verhalten eines charakteristischen Parameters dar. Die Gesamtfläche ist die Summe aus rechteckigen Einzelflächen. Die Einzelflächen werden zu den diskreten Zeitschritten aus der Ausgabeschrittweite $\Delta\zeta$ und der gemittelten und normierten Differenz aus aktuellem und Zielverhalten bestimmt:

$$P_{ik} = \frac{P_{ik} - GP_{i(k-1)}}{max\,(GP_i) - min\,(GP_i)}$$

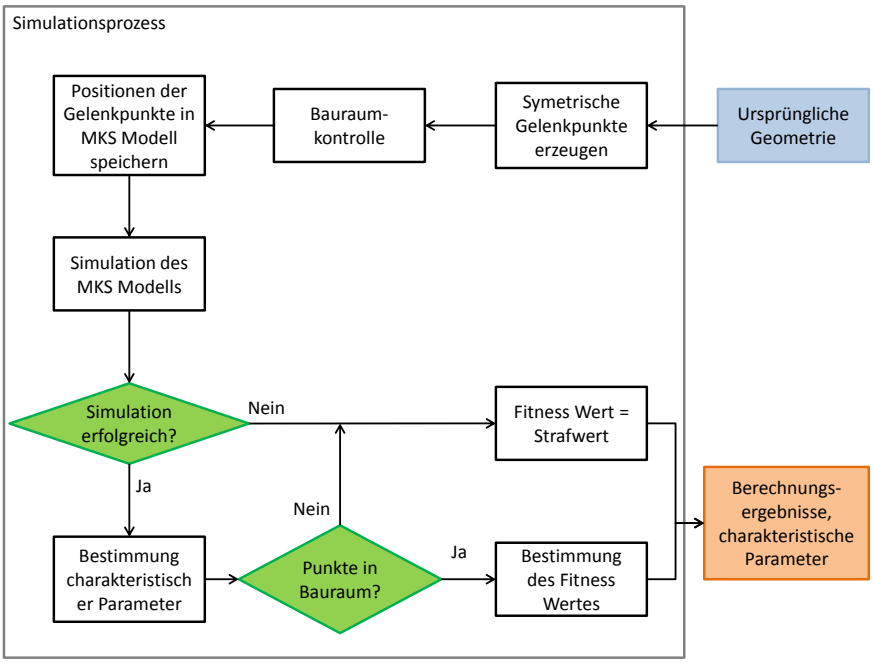


Abbildung 2.20.: Flussdiagramm Simulationsprozess

Symetriefunktion

Die Symmetriefunktion dient dazu, Punkte an einer Symmetrieebene zu spiegeln. Will man beispielsweise eine ganze Achse, also linke und rechte Radaufhängung, in einer Simulation berechnen, so muss man darauf achten, dass in der Konstruktionslage die Gelenkpunkte der linken Radaufhängung symmetrisch bezüglich der Fahrzeugmitte zu jenen der rechten Radaufhängung sind.

In der Sensibilitätsanalyse und im Optimierungsprozess werden allerdings in jeder Iteration Gelenkpositionen in der Konstruktionslage verändert. Dabei müssen die Positionen der entsprechenden symmetrischen Gelenkpunkte korrigiert werden. Diese Gelenkpunkte sind nicht Gegenstand der Optimierung da ihre Position von einem zu optimierenden Gelenkpunkt abhängig ist. In Abbildung 2.22 ist das CAD Modell der Vorderachse des Fallbeispiels Shuttle Car dargestellt. Das stark vereinfachte Skelett ist in blau dargestellt. Die rot markierten Gelenkpunkte werden an der x-z Ebene des Fahrzeugkoordinatensystems gespiegelt. Es können die anderen Hauptebenen des Fahrzeugkoordinatensystems, aber auch beliebig im Raum liegende Ebenen als Symmetrieebenen angegeben werden. Im letzten Fall wird die Ebene durch die Parameter a, b und c der Ebenengleichung $a * x + b * y + c * z = 1$ angegeben.

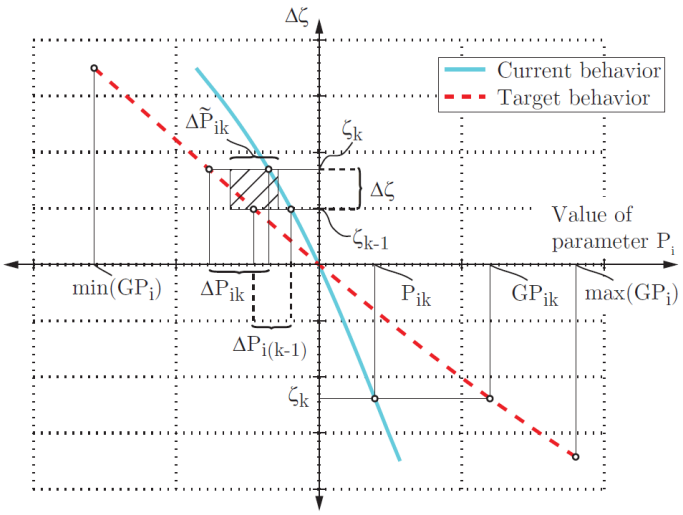


Abbildung 2.21.: Ermittlung der Kostenfunktion, Quelle: [Roj12]

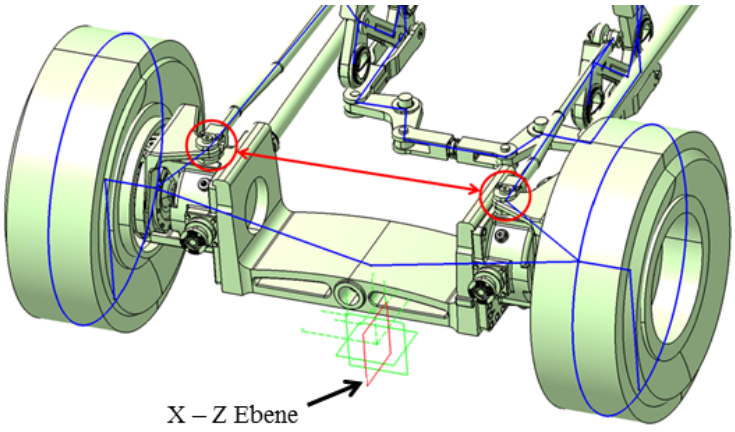


Abbildung 2.22.: Kippachse und Skelett des Shuttle Cars

2.3.4. Sensibilitätsanalyse

Um ein Verständnis des untersuchten Mechanismus zu erlangen, kann eine Sensibilitätsanalyse durchgeführt werden. Dabei wird die Position eines definierten Gelenkpunktes in jeder der drei kartesischen Koordinatenrichtungen in einem vorher definierten Bereich variiert. In jeder Position wird einmal die Simulationsfunktion aufgerufen und die charakteristischen Parameter werden berechnet. Das Ergebnis dieser Analyse ist die Variation der charakteristischen Parameter in jeder Koordinatenrichtung. Somit kann abgeschätzt werden, welchen Einfluss ein Gelenkpunkt auf einen charakteristischen Parameter hat.

In Abbildung 2.23 ist ein Flussdiagramm der Sensibilitätsanalyse dargestellt. Wie bei der Simulationsfunktion stellen die aktuellen Positionen der Gelenkpunkte den Modelleingang dar. In einer Schleife wird die Symmetriefunktion aufgerufen, die aktuellen Gelenkpositionen im MKS Modell gespeichert und der Mechanismus simuliert. Danach werden die charakteristischen Parameter in einem Diagramm dargestellt und das Abbruchkriterium geprüft. Die Analyse ist abgeschlossen wenn jeder gewünschte Gelenkpunkt analysiert wurde. Abbildungen der Sensibilitätsanalyse des Anwendungsbeispieles Schuttle Car sind in Anhnang A.2 zu finden.

In der Sensitivity_Parameter Datei, Matlab® Code A.5, kann definiert werden in welchem Volumen der jeweilige Gelenkpunkt variiert werden soll:

- Absolut in mm.
- Relativ im Verhältnis zum erlaubten Bauraum. Mit dem Aufruf einer eigenen VBA-Function werden die Dimensionen des erlaubten Bauraumes ermittelt. Der Faktor $Rel\_Variation$ gibt das Verhältnis des Volumens das in der Sensibilitätsanalyse betrachtet wird, zum erlaubten Bauraum an.

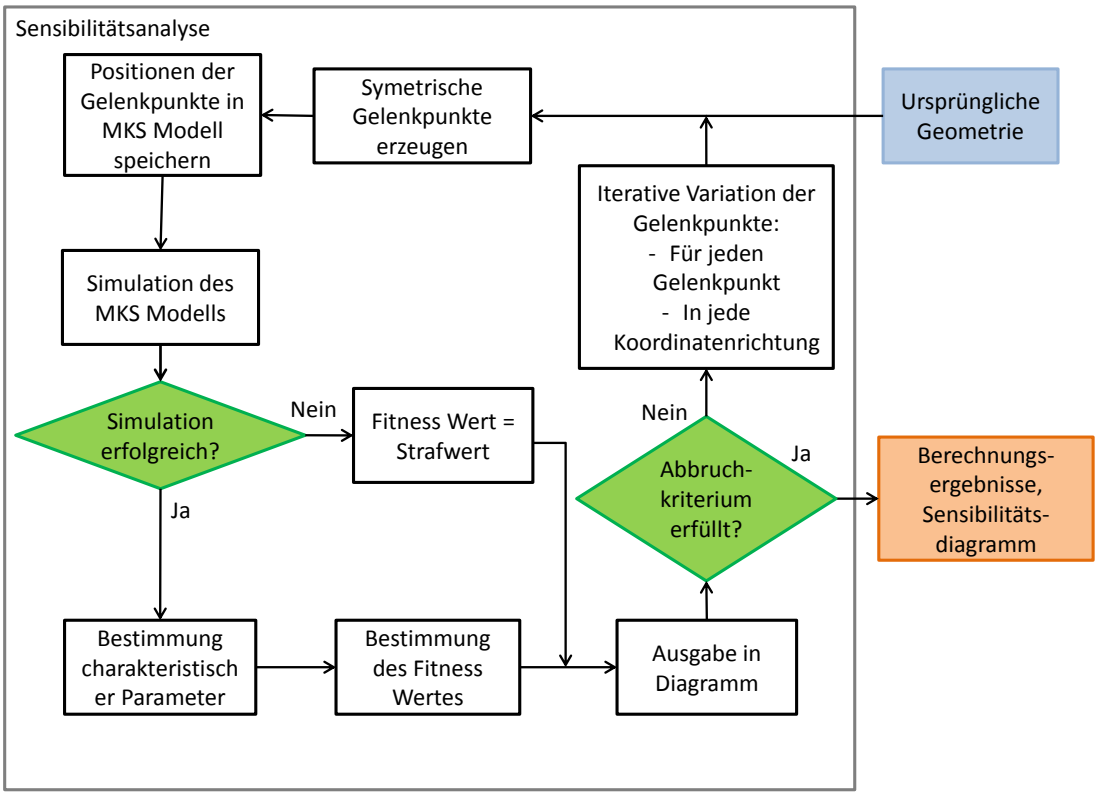


Abbildung 2.23.: Flussdiagramm Sensibilitätsanalyse

2.3.5. Optimierungsfunktion

In diesem Abschnitt wird gezeigt wie der Optimierungsprozess den Simulationsprozess verwendet wird um ein erwünschtes Verhalten charakteristischer Parameter eines Mehrkörpersysems zu erzielen. Zuerst wird ein Überblick über den Prozessablauf gegeben. Danach wird auf einzelne wichtige Funktionen eingegangen.

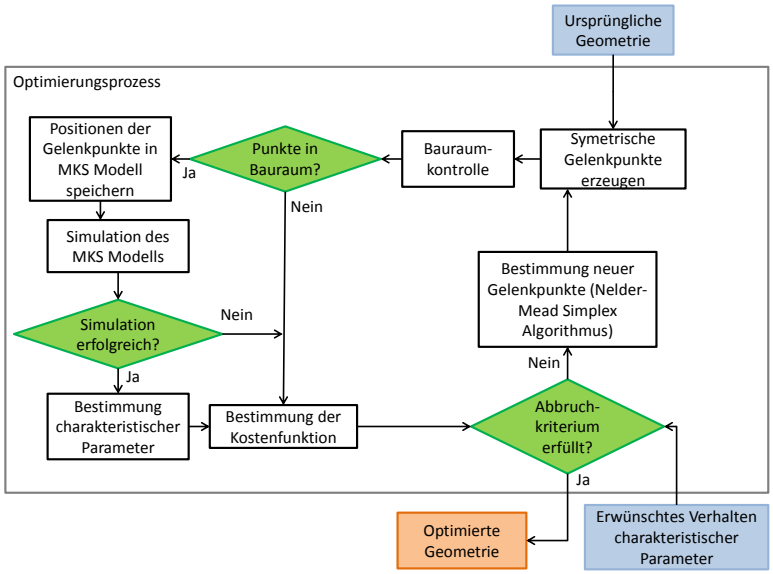


Abbildung 2.24.: Flussdiagramm der Optimierungsfunktion

In Abbildung 2.24 ist in einem Flussdiagramm der Optimierungsprozess dargestellt. Den Eingang ins System bilden einerseits die Geometrie des Mehrkörpersystems in ihrer ursprünglichen Konfiguration, andererseits das erwünschte Verhalten der charakteristischen Parameter. Im ersten Schritt jeder Iteration wird die Symmetrie definierter Gelenkpunkte bezüglich einer Symmetrieebene hergestellt. Danach erfolgt die Bauraumkontrolle, in der kontrolliert wird, ob alle Gelenkpunkte innerhalb ihres vorgesehenen Bauraumes liegen. Liegt ein Gelenk außerhalb des vorgesehenen Bauraumes, wird keine Simulation durchgeführt und in der Kostenfunktion ein Strafwert ermittelt. Andernfalls werden nach der Simulation die charakteristischen Parameter und ausgehend davon die Kostenfunktion berechnet.

Bis hierher ist der Optimierungsprozess ident mit dem Simulationsprozess. Nun wird aber der Simulationsprozess in einer Schleife wiederholt. In jeder Iteration werden die Positionen der Gelenkpunkte verändert, die Simulation neu durchgeführt und die charakteristischen Parameter und die Kostenfunktion berechnet.

In Matlab® geschieht dies mit der buildt-in Function $fminsearch$. Diese Function sucht, unter Verwendung eines Nelder-Mead Simplex Algorithmus, Abschnitt 2.3.5, nach den Parametern x einer nichtlinearen Funktion fun, bei denen der Funktionswert von fun ein

Minimum besitzt. Den Parametern x entsprechen in MuBOS die x-, y- und z-Koordinaten jener Gelenkpunkte, die optimiert werden sollen. Mit fun wird der $fminsearch$-Function die Simulationsfunktion von MuBOS übergeben. Der darin berechnete Kostenwert J, siehe Kostenfunktion in Abschnitt 2.3.3, stellt den Funktionswert von fun dar, nach dessen Minimum gesucht wird. Mit $x0$ werden die Startwerte der Parameter von fun übergeben. In MuBOS sind das die ursprünglichen Positionen jener Gelenkpunkte, die optimiert werden sollen, [MAT08].

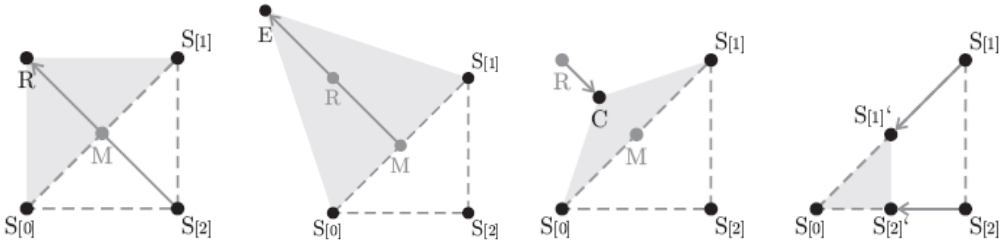
Mit *options* wird eine Matlab® Structure übergeben, in der verschiedene Optionen festgelegt werden können. Das sind einerseits Einstellungen zur Anzeige von Ergebnissen während des Optimierungsvorganges, andererseits werden hier die Abbruchkriterien der $fminsearch$-Funktion definiert. Es können folgende Möglichkeiten stehen zur Wahl:

- Maximale Anzahl an Aufrufen der Simulationsfunktion
- Maximale Anzahl an Iterationen
- Toleranzwert des Kostenwertes J
- Toleranzwert der Funktionsparameter x

Ist zwischen zwei Iterationen die Änderung des Kostenwertes J oder der Funktionsparameter x, das sind im Fall von MuBOS die Gelenkpositionen in x-, y-, und z-Richtung, geringer als der entsprechende Toleranzwert, wird die Optimierungsfunktion beendet. Der Abbruch erfolgt, sobald eines der Abbruchkriterien erfüllt ist, [MAT08].

Es existiert eine Reihe von Optimierungsalgorithmen. Sie lassen sich in folgende Kategorien einteilen, [KR11]:

- Deterministische Verfahren: Auch Hillclimbing Strategien genannt. Ausgehend von einem Startpunkt werden umliegende diskrete Nachbarpunkte berechnet. Der am besten bewertete Nachbarpunkt wird als neuer Startwert definiert. Auf diese Weise nähert sich die der Algorithmus zielstrebig einem Optimum. Vorteilhaft ist die Schnelligkeit mit der sich das Verfahren einem Optimum nähert. Allerdings ist es unwahrscheinlich dass ein globales Optimum gefunden wird, da das Ergebnis stark vom Startpunkt abhängt.
- Stochastische Verfahren: Dabei werden im gesamten Suchraum zufällige Startpunkte erzeugt. Die am besten bewerteten Punkte werden wieder als Startpunkte übernommen, von denen neue Punkte erzeugt werden. Der entschiedende Vorteil dieses Verfahrens liegt in der zufälligen Bestimmung von Startpunkten im gesamten Suchraum. Damit besteht die Möglichkeit ein globales Optimum zu finden. Allerdings erfordert diese Methode eine große Anzahl an Berechnungen, was zu langen Rechenzeiten führt.
- Evolutionäre und genetische Verfahren: Diese Methoden bilden die natürliche Evolution von Organismen nach. Es wird eine zufällige Elternmenge bestimmt und davon durch Mutation und Rekombination eine Kindermenge erzeugt. Diese wird



(a) Reflection (b) Expansion (c) Contraction (d) Shrinking

Abbildung 2.25.: Geometrische Operationen des NMS, Quelle: [MJ08]

mittels Selektion bewertet. Die besten Individuen werden als neue Elterngeneration übernommen. Diese Verfahren führen schrittweise zu einem Optimum. Nachteilig ist auch hier die große Zahl berechneten Lösungspunkten.

Nelder-Mead Simplex Algorithmus

Als Suchmethode für neue Gelenkpositionen wird in MuBOS, das deterministische Verfahren des Nelder-Mead Simplex Algorithmus (NMA) verwendet. Damit werden kurze Rechenzeiten gewährleistet. Allerdings ist immer darauf zu achten, dass evtl. kein globales Optimum ermittelt wird. Der Algorithmus berücksichtigt nur Funktionswerte selbst und keine Ableitungen der Funktion, [NM65].

Der Suchvorgang basiert auf vier geometrischen Operationen, Reflexion, Expansion, Contraction und Shrinking, an einem Polygon (Simplex) mit $n+1$ Eckpunkten in $\mathbf{R}^n$.

In $\mathbf{R}^2$ ist das beispielsweise ein Dreieck, in $\mathbf{R}^3$ ein Tetraeder. Jeder Eckpunkt dieses Polygons stellt einen Funktionswert dar. In Abbildung 2.25 sind die Operationen für $\mathbf{R}^2$ dargestellt. Die ersten Eckpunkte $S_{(1)}$ und $S_{(2)}$ des Polygons werden, ausgehend von den Anfangswerten $S_{(0)}$, willkürlich ermittelt. In jeder weiteren Iteration wird jener Punkt mit den schlechtesten Eigenschaften, im Fall von MuBOS jener mit dem höchsten Kostenwert, verworfen und seine Spiegelung R am Schwerpunkt der restlichen Punkte, M, berechnet. Wird R weder besser noch schlechter als alle bisherigen Punkte bewertet, wird R ein neuer Eckpunkt des Plygons (Reflection Parameter $\alpha = 1$, Abbildung 2.25(a)). Wird R besser als alle bisherigen Punkte bewertet, so wird R weiter in dieselbe Richtung verschoben (Expansion Parameter $\gamma = 1$, Abbildung 2.25(b)). Ist der resultierende Punkt E besser als R, so wird E neuer Eckpunkt des Polygons. Ist R schlechter als alle bisherigen Punkte, so wird der Punkt C, zwischen R und M, neuer Eckpunkt (Contraction Parameter $\rho = 0,5$, Abblildung 2.25(c)). Ist C immer noch schlechter als alle bisherigen Punkte, wird C verworfen und alle Punkte werden in Richtung des am besten bewerteten Punktes verschoben (Shrinking Parameter $\sigma = 0,5$, Abbildung 2.25(d)).

3. Anwendungen

In diesem Kapitel werden zwei Anwendungen von MuBOS beschrieben. Im ersten Beispiel werden charakteristische Parameter einer Schwertlenkeraufhängung optimiert. Dabei handelt es sich um einen Anwendungsfall der schon von [Wil10] betrachtet wurde. Sinn dieses Beispiels ist einerseits

- Validitierung der Simulationsergebnisse, andererseits
- Demonstration der Anwendung anhand einer Radaufhängung.

In der zweiten Anwendung wird die Optimierung einer Allradlenkung eines Sonderfahrzeuges gezeigt. Diese zeichnet sich durch ein ungewöhnlich komplexes Lenkgestänge aus, womit gezeigt werden soll, wie mit MuBOS beliebige Mehrkörpersysteme optimiert werden können.

3.1. Schwertlenkeraufhängung

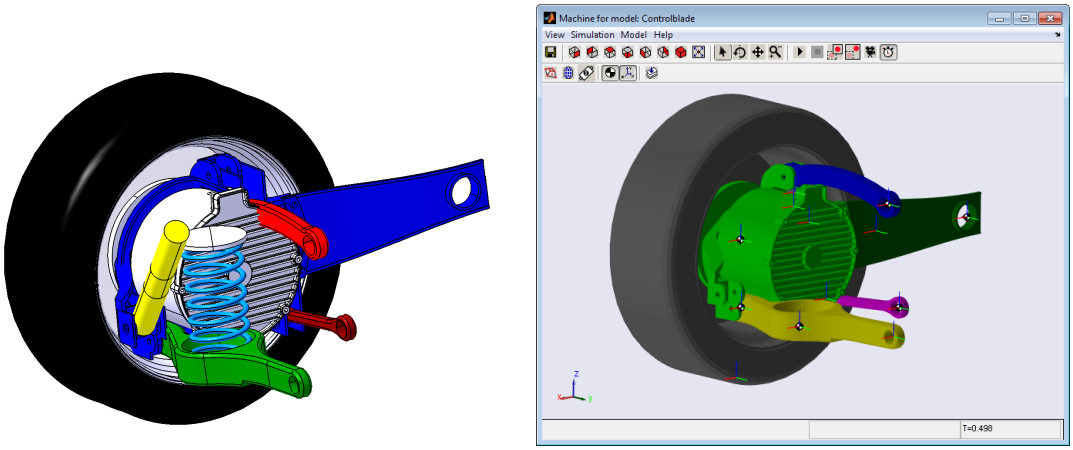
3.1.1. Beschreibung der Radaufhängung

Diese Art der Radaufhängung wurde von Ford, im Modell Focus, 1999 unter dem Namen Control Blade^{TM} entwickelt. Die Aufhängung betsteht aus einem charakteristischen Längslenker in Form eines vertikal angeordneten Schwertes aus gehärtetem Stahlblech. Dieser Lenker überträgt Längskräfte und erlaubt dem Radträger Bewegungen in y- und z-Richtung, sowie Drehung um die z-Achse aufgrund der Biegeelastizität des Bleches. Drei Querlenker übertragen Querkräfte. Mit der Anordnung dieser Lenker und ihrer elastischen Lager, können charakteristische Parameter wie Sturz und Spurwinkel δ beeinflsst werden. Zwei dieser Lenker sind unter der Radachse und einer darüber angeordnet, [HM11] [HE07]. Abbildung 3.1(a) zeigt das CAD Modell der Radaufhängung in Catia V5® und Abbildung 3.1(b) das MKS Modell in Simmechanics.

Vorteile dieser Radaufhängung sind:

- Bremsnickausgleich durch die vertikale Anordnung des Schwertes.
- Das Verhalten charakteristischer Parameter, wie Sturz und Spurwinkel, können leicht durch die drei Querträger gesteuert werden, [HM11].
- Das Fehlen eines Federbeines ergibt eine platzsparende Bauweise.

Nachteil dieser Radaufhängung ist:



(a) CAD Modell, nach [Roj12]

(b) MKS Modell

Abbildung 3.1.: Modell der Schwertlenkeraufhängung

- Elastoinematische Eigenschaften haben sehr großen Einfluss auf die Auslgeung.

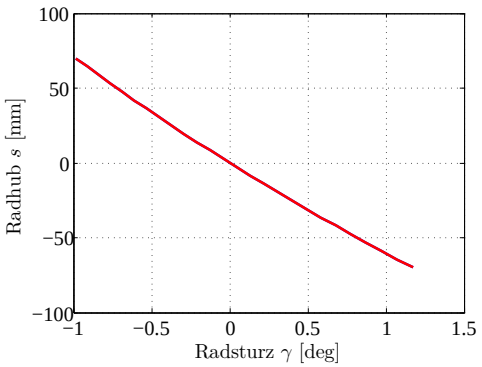
3.1.2. Problemstellung

Im Zuge immer stärkerer Elektrifizierung des Antriebstranges stellen elektrische Radnabenmotoren eine interessante Alternative zum konventionellen Antriebstrang dar. Die Integration in bestehende Radaufhängungen bringt allerdings Problemstellungen, wie erhöhte ungefederte Massen oder Bauraumprobleme, mit sich. In diesem Anwendungsfall entsteht durch den Einbau eines Radnabenmotors in eine Schwertlenkerachse eine Kollision mit dem oberen Querlenker, Abblildung 3.1, [Roj12].

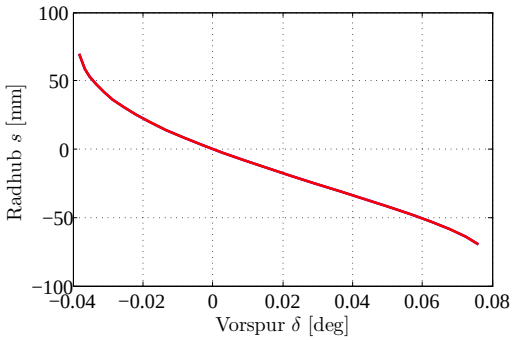
Ziel der Optimierungsaufgabe ist es nun die Gelenkpunkte dieses Lenkers so auszulegen, dass die ursprünglichen Eigenschaften der charakteristischer Parameter so gut wie möglich wieder erreicht werden. Zu diesem Zweck werden Sturz γ und Vorspurwinkel δ betrachtet. Elastizitäten in den Lagerstellen der Lenker werden nicht berücksichtigt.

Vorgehensweise: Die folgende Vorgehensweise wird angewendet:

- Ausarbeitung der charakteristischen Parameter Sturz γ und Vorspur δ.
- Berechnung dieser Parameter für die ursprüngliche Konfiguration ohne Radnabenmotor.
- Vorgabe der Simulationsergebnisse als Zielfunktion für den Optimierungsprozess.
- Verschiebung der Gelenkpunkte des Lenkers in eine Positon ohne Kollision mit Radnabenmotor, Felge oder Bremsanlage.
- Optimierung des Mehrkörpersystems.



(a) Zielfunktion Sturzwinkel



(b) Zielfunktion Spurwinkel

Abbildung 3.2.: Zielfunktionen

3.1.3. Bestimmung charakteristischer Parameter

Sturzwinkel γ: Der Sturzwinkel wird im MKS-Modell selbst berechnet und als Simulationsergebnis der MuBOS Optimierungsfunktion zurückgegeben. Er wird also in der Fitness-Funcion Datei nicht berechnet sondern nur als charakteristischer Parameter definiert. Der von der Kostenfunktion bewertete charakteristische Parameter wird daher als Integral des Sturzwinkels γ über dem Radhub s definiert:

$$J_\gamma = \int_{-70}^{70} \gamma \mathrm{d}s \tag{3.1}$$

Vorspurwinkel δ: Der Sturzwinkel wird ebenfalls im MKS-Modell selbst berechnet und als Simulationsergebnis der MuBOS Optimierungsfunktion zurückgegeben. Der von der Kostenfunktion bewertete charakteristische Parameter wird daher als Integral des Sturzwinkels γ über dem Radhub s definiert:

$$J_\delta = \int_{-70}^{70} \delta \mathrm{d}s \tag{3.2}$$

3.1.4. Ergebnisse

Eine erste Berechnung des Systems mit dem Querlenker in ursprünglicher Position liefert die Zielfunktionen der charakteristischen Parameter, Abbildung 3.2(a) und 3.2(b).

Im nächsten Schritt wird der Querlenker in eine Position verschoben, die keine Kollision mit anderen Baugruppen verursacht, blau dargestellt in Abbildung 3.4. Die Verschiebung beträgt +70 mm in Z-Richtung. Für das radträgerseitige Gelenk wird ein erlaubter Bauraum definiert, und in der Optimierungsfunktion überwacht, orange dargestellt in Abbildung 3.4.

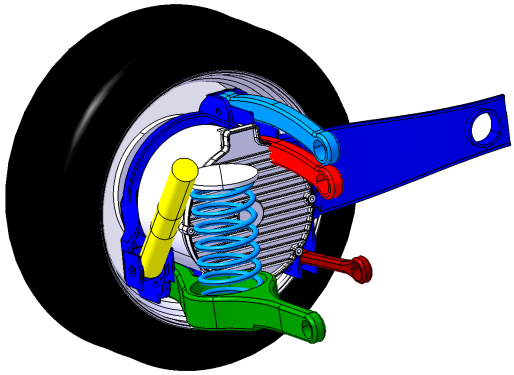


Abbildung 3.3.: Neupositionierung des Querlenkers, nach [Roj12]

Eine erste Simulation mit dieser Konfiguration ergibt folgende Werte der charakteristischen Parameter:

$$J_\gamma = 2,15 \cdot 10^{-3} \tag{3.3}$$

$$J_\delta = 1,32 \cdot 10^{-5} \tag{3.4}$$

Mit diesen Ergebnissen können die Einstellungen für den Optimierungsprozess getroffen werden, Tabelle 3.6. Mit den Faktoren $w_{om,i}$ werden beide charakteristischen Parameter auf die selbe Größenordnung gebracht. Mit $w_i = 0.5$ werden beide Kriterien als gleichwertig betrachtet, siehe auch Kostenfunktion 2.1.

	J_γ	J_δ
Wert	$2,15 \cdot 10^{-3}$	$1,32 \cdot 10^{-5}$
$w_{om,i}$	10^6	10^8
w_i	0.5	0.5

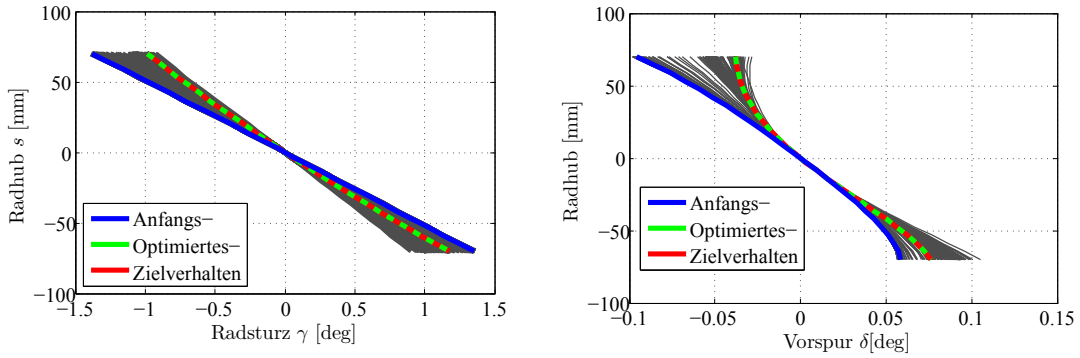
Tabelle 3.1.: Charakteristische Parameterwerte und Gewichtungsfaktoren der Schwertlenkerachse

Die Abbruchkriterien, siehe Abschnitte 2.3.5 und 2.3.1.3 und Mechanism_Parameter Datei A.4, werden wie in Tabelle 3.5 dargestellt definiert.

Abbruchkriterium	Wert
$Max\_Function\_Evals$	2000
$TolFun$	10^{-2}
$TolX$	10^{-3}

Tabelle 3.2.: Abbruchkriterien

Damit kann die Optimierung mit MuBOS durchgeführt werden. In Abbildung 3.4 ist der Optimierungsprozess beider charakteristischer Parameter dargestellt. Die blaue Li-



(a) Optimierung des Sturzwinkels

(b) Optimierung des Spurwinkels

Abbildung 3.4.: Optimierung der Schwertlenkeraufhängung

nie stellt jeweils die Zielfunktion, die rote Linie das Verhalten des Parameters nach der Optimierung und die grüne Linie das Verhalten vor dem Optimierungsprozess dar. Man erkennt, dass beide Parameter ihr ursprüngliches Verhalten vor Einbau des Radnabenmotors sehr gut erreichen.

Abbildung 3.5 zeigt die drei Schritte des Optimierungsvorganges. Die ursprüngliche Lenkerposition ist rot dargestellt. Man erkennt die Kollision mit dem Radnabenmotor, grau dargestellt. Der hellblaue Lenker stellt die Position nach Verschieben und vor Optimierung des Systems dar. Der orange Block ist der erlaubte Bauraum für das radträgerseitige Gelenk. Das Ergebnis der Optimierung ist in grün dargestellt. Das radträgerseitige Gelenk ist nicht an die Grenze des erlaubten Bauraumes gestoßen. Die rot strichlierte Linie stellt das Skelett des CAD Modells vor- und die grüne Linie nach der Optimierung dar.

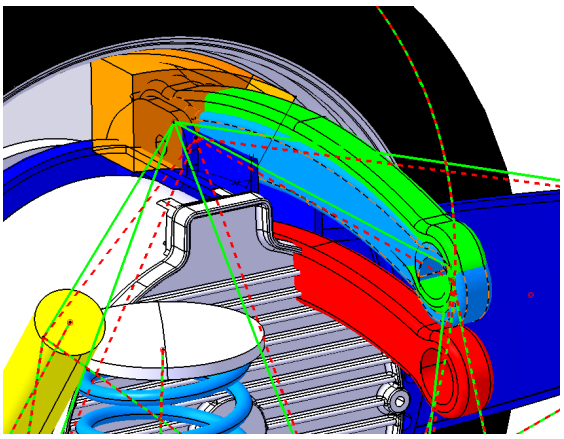


Abbildung 3.5.: Optimierungsschritte der Schwertlenkerachse

(a) CAD Modell (b) Im Betrieb

Abbildung 3.6.: Shuttle Car TC790, Quelle: [San09, San12]

3.2. Lenkungsoptimierung eines Load and Haul Shuttle Cars

3.2.1. Load and Haul Shuttle Car TC790

Dieses Sonderfahrzeug wird im Untertagebergbau verwendet. Es dient dazu geschnittenes Material, z.B. Kohle oder weiche Minerale, von einer konstant fördernden Gewinnungsmaschine aufzunehmen, zu transportieren und zu entladen. Abbildung 3.6 zeigt das Fahrzeug im CAD-Modell und im Einsatz unter Tage. Das Fahrzeug wird mit dem Transportgut über ein Förderband beladen. Das Entladen erfolgt selbsttätig durch einen Kratzboden.

Das Fahrzeug hat folgende Eckdaten, [San09, San12]:

Größe	Wert
Nutzlast	18 t
Leergewicht	30 t
Länge	92 mm
Breite	3460 mm
Höhe	2000 mm
Bodenfreiheit	300 mm
Radstand	2800 mm
Spurweite	2600 mm
Max. Fahrgeschwindigkeit	10 km/h

Die Konstruktion des Fahrzeuges wird sehr stark durch die Gegebengheiten unter Tage bestimmt, [San12]:

- Frischwetter: Um möglichst geringe Emmisionen zu verursachen, werden oft elektrische Antriebe verwendet.
- Um Schlagwetter zu vermeiden muss die Elektische Ausrüstung Sicherheitsstandards, wie MSHA/PBMS, ATEX und GOST erfüllen.

- Platzbeschränkung: Größe und Form der Stollen beeinflusst
  - Länge
  - Breite
  - Höhe und
  - Kurvenradius

  des Fahrzeuges

Da der Platz für das Fahrzeug beschränkt ist, gilt es den vorhandenen Bauraum effizient zu nutzen um eine möglichst hohe Zuladung sicherzustellen. Die Komponenten des Fahrzeuges sind seitlich angebracht, daher kann die Ladefläche sehr tief gelegt werden. Die Höhe der Ladefläche wird dabei von der Bodenfreiheit und den notwendigen mechanischen, elektischen und hydraulischen Verbindungen von linker und rechter Seite bestimmt.

Wie in Abbildung 3.7 dargestellt, besitzt das Fahrzeug zwei getrennte Antriebstränge für linke und rechte Räder. Diese werden durch je einen VFD (Variable Frequency Drive) Elektromotor angetrieben. Gschwindigkeit bzw. Antriebsmoment werden dabei rein elektronisch gesteuert. Ein Differentialgetriebe, welches in Fahrzeugmitte Bauraum beanspruchen würde, entfällt.

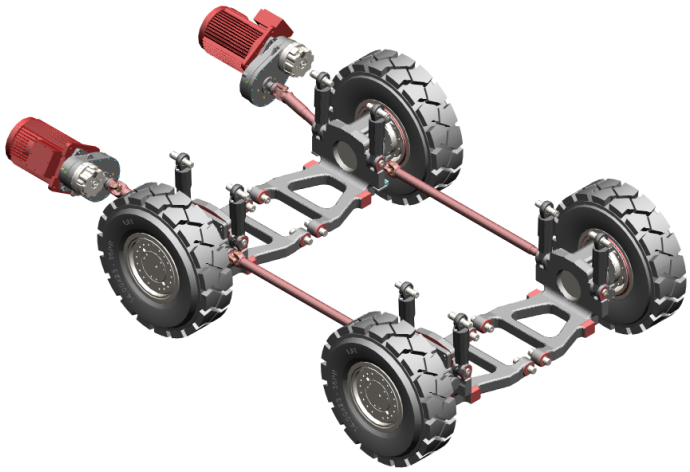


Abbildung 3.7.: Antriebstrang und Radaufhängung TC790

Das Fahrzeug wird durch ein Kabel mit elektrischer Energie versorgt. Dieses wird während der Fahrt von der Kabeltrommel auf- oder abgerollt. Die Versorgungsspannung beträgt, je nach Einsatzort, zwischen 480 und 1140V. Wegen der strengen Bestimmun-

gen für flammsichere Anlgen untertage, werden nur die Fahrmotoren, der Antriebsmotor des Kratzbodens und die Hydraulikpumpe elektrisch betrieben. Alle anderen Aggregate und Antriebe, wie

- Servicezylinder,
- Kabeltrommel,
- Kühlsystem,
- Kratzboden Hebevorrichtung und
- Lenkzylinder

werden durch das Hydrauliksystem angetrieben.

Abbildung 3.8.: Antriebstrang und Radaufhängung des TC790

Die Lenkung des Shuttle Car ist symmetrisch zur Fahrzeuglängsachse aufgebaut und bildet ein verallgemeinertes Lenktrapez. Abbildung 3.8 zeigt Antriebstrang und Lenkung.

Es werden alle vier Räder gelenkt. Die Lenkung ist symmetrisch zur Fahrzeugquerachse aufgebaut, wodurch Vorder- und Hinterräder jeweils gleich stark einlenken. Damit wird ein sehr kleiner Bahnradius bei Kurvenfahrt erreicht. Abbildung 3.9 zeigt die Platzverhältnisse in einem $6200mm$ breiten Stollen.

Die Lenkkraft wird durch zwei Hydraulikzylinder eingeleitet. Dabei steuert der Fahrer mit einem Joystick den Druck in den links und rechts angeordneten Druckzylindern, (1) in Abbildung 3.10. Über das Hebelsystem (2) (3) und (4) wird die Lenkbewegung auf die Spurstangen (6) und (7), und von diesen weiter auf die Spurhebel der Vorder- und Hinterräder übertragen. Im Gegensatz zum Antriebstrang müssen bei der Lenkung linke

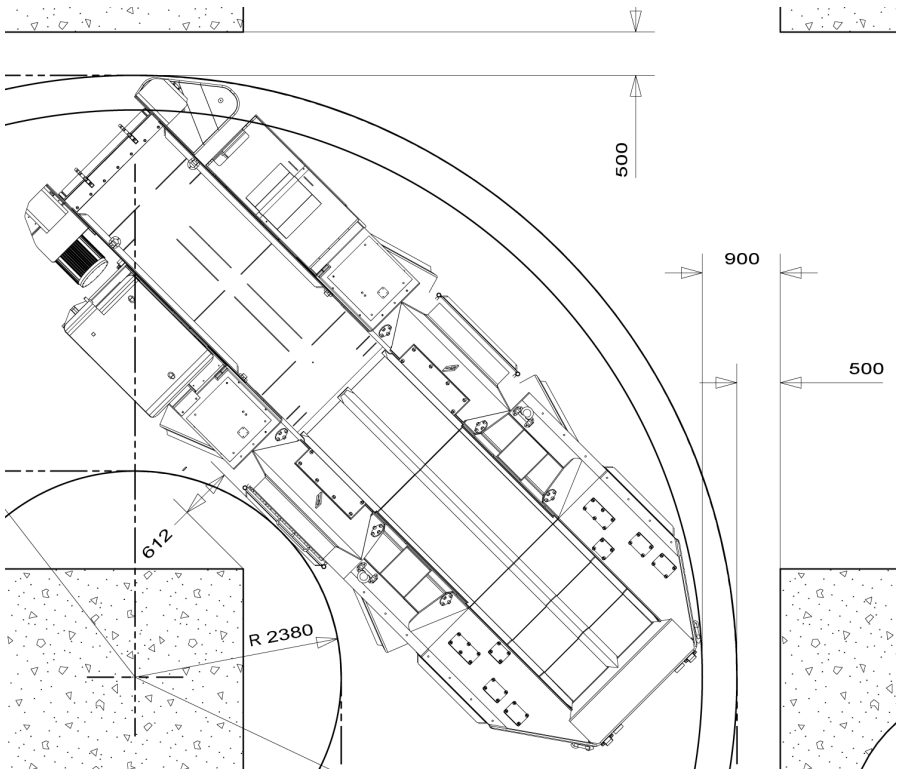


Abbildung 3.9.: Kurvenfahrt des TC790

und rechte Seite miteinander mechanisch verbunden sein, um ein sinnvolles Lenkverhalten sicherzustellen. Das geschieht mit dem Spurgestänge (9), (10) und (11), das unter der Ladefläche angeordnet ist.

Wie auch der Antriebstrang, ist die Lenkung stark im Bauraum eingeschränkt. Die wichtigsten einschränkenden Baugruppen sind:

- Rahmen
- Räder
- Getriebe und Antriebstrang
- Hydraulikkasten
- Elektrikkasten

3.2.2. Problemstellung Load and Haul Shuttle Car TC100

Im Zuge eines neuen Auftrages wurde ein Projekt gestartet, um das bestehende Fahrzeug an die Bedürfnisse eines neuen Kunden anzupassen. Die wichtigsten Anforderungen an

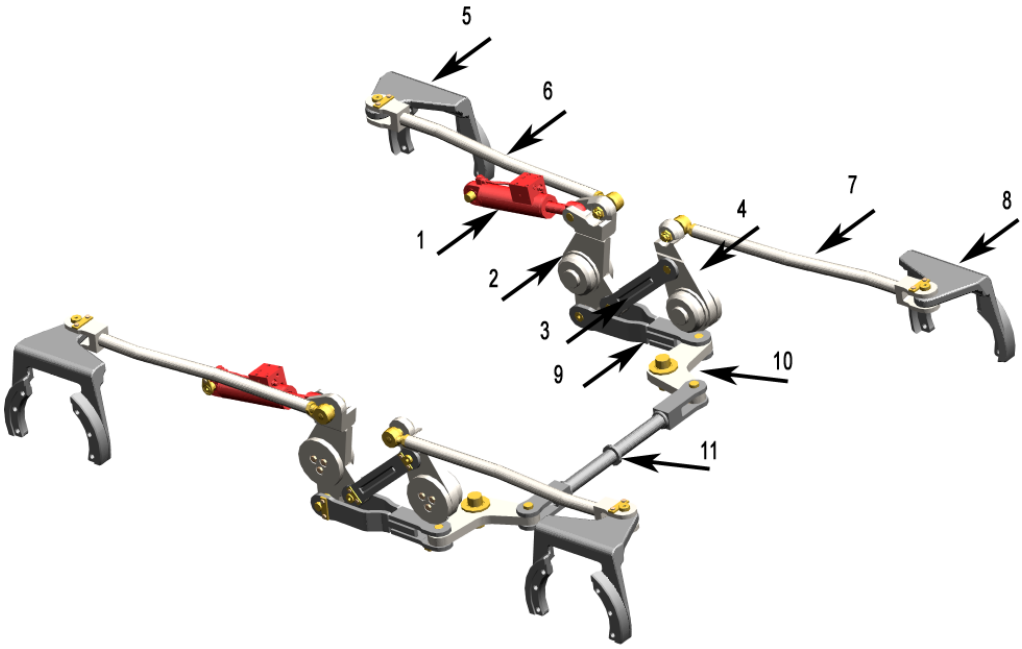


Abbildung 3.10.: Lenkung des TC790

das neue Fahrzeug TC100 sind:

- Reduktion der Stollenbreite auf $4800mm$
- Reduktion der Stollenhöhe auf $1800mm$
- Kostenbeschränkung

Wegen dieser Einschränkungen wurde eine Neukonstruktion des gesamten Fahrzeuges notwendig. Dabei wurden folgende Veränderungen am Fahrzeug durchgeführt:

- Reduktion der Fahrzeughöhe
- Reduktion der Spurweite
- Aufgrund der Kostenbeschränkung wird auf eine Einzelradaufhängung verzichtet. Wie in Abbildung 3.11 dargestellt, werden statt dessen:
  - eine Kippachse als Vorderachse, sowie
  - eine Starrachse als Hinterachse verwendet.

Diese Änderungen führen einerseits zu einer geringeren Ladekapazität und einer Neuanordnung oder Neukonstruktion aller Baugruppen mit den damit verbundenen Packaging

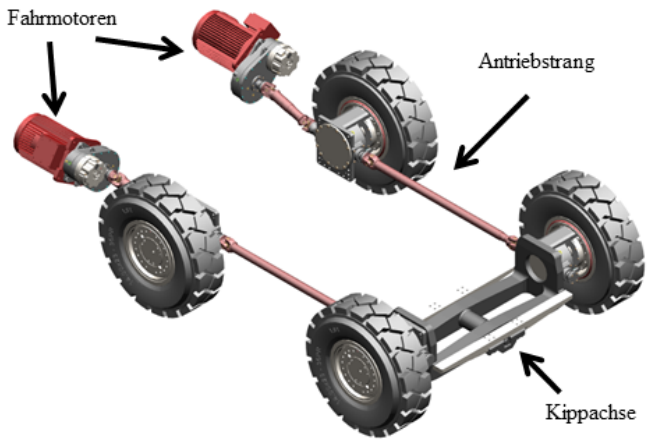


Abbildung 3.11.: Antriebstrang und Radaufhängung des TC100

Problemen. Auch die Lenkung des Fahrzeuges ist von diesen Änderungen stark betroffen. Werden die Bauteile unverändert in Richtung Fahrzeugmitte verschoben und nur die Spurstange, (11) in Abbildung 3.10, an die neue Spurweite angepasst, verändert sich das Lenkverhalten. Auch die Kräftesituation im Lenkgestänge verändert sich wegen der geringeren Fahrzeuggesamtmasse.

Ziel ist es daher, in einer frühen Entwicklungsphase die Lenkung auszulegen. Als Hauptziele werdern definiert:

- Die Lenkung des Schuttle Car TC790 ist anhand der unten angeführten Anforderungen an die neuen Randbedingungen anzupassen.
- Um die Lenkung in Zukunft rasch an neue Randbedingungen anpassen zu können, ist ein MKS-Modell zu erstellen.

Anforderungen: In einem Lastenheft werden Anforderungen definiert:

- Die Lenkung wird nach kinematischen Gesichtspunkten ausgelegt.
- Ein Festigkeitsnachweis wird in dieser Arbeit nicht erbracht. Dieser ist Teil einer weiterführenden Diplomarbeit.
- Das Fahrzeug wird mit geringen Geschwindigkeiten von max. 10 $^{km}/_{h}$ gefahren. Daher werden keine Untersuchungen des dynamischen Fahrverhaltens durchgeführt.
- Das Fahrzeug ist allradgelenkt und besitzt eine Starr- und eine Kippachse.
- Um Schlupf zu verringern ist ein Lenkverhalten nach Ackermann anzustreben.
- Die Geometrie des gesamten Lenkgestänges darf verändert werden. Ausnahmen

bilden Räder und Getriebe. Somit dürfen die Radgeometrie und die Lenkachsen nicht verändert werden.

- Der max. Einlenkwinkel der jeweils kurveninneren Räder beträgt $21,5\,°$.
- Abbildung 3.12 zeigt das Platzangebot bei Kurvenfahrt. Um eine Kollision des Fahrzeuges mit der Stollenwand zu vermeiden, ist ein möglichst geringer Bahnradius anzustreben.

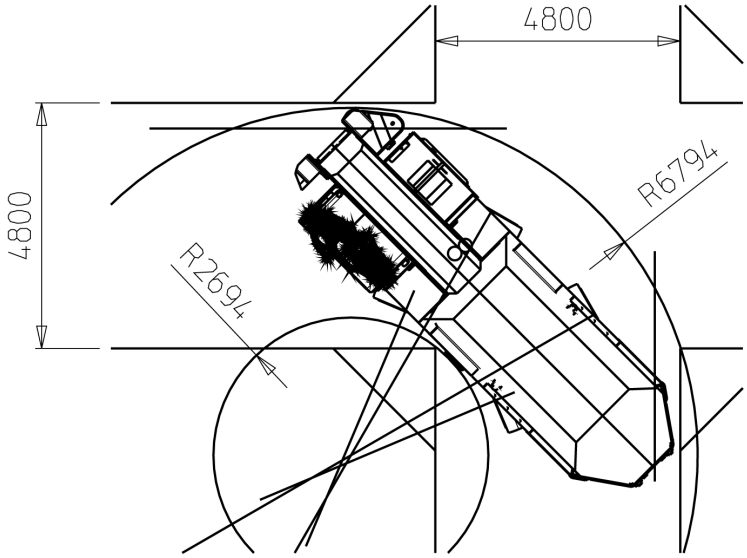


Abbildung 3.12.: Kurvenfahrt des TC100, Quelle: [San12]

Vorgehensweise: Die folgende Vorgehensweise wird angewendet, um die Lenkung auszulegen:

- Überführen der CAD Daten des Fahrzeuges von Unigraphics nach Catia V5R16® mit dem Step-Format
- Aufbau eines MuBOS tauglichen Catia V5® Produkts
- Ausarbeiten charakteristischer Parameter
- Optimierung der Lenkgeometrie mit MuBOS
- Rückführung der optimierten Lenkgeometrie nach Unigraphics

3.2.3. Bestimmung charakteristischer Parameter

3.2.3.1. Ackermann

Ein wichtiges Ziel der Neuauslegung des Shuttle Cars ist es, den Reifenverschleiß zu minimieren. Dieser entsteht durch Schlupf der Räder auf der Fahrbahn und hat folgende Ursachen:

Antriebschlupf / Bremsschlupf: Dieser kann durch die Lenkung nicht beeinflusst werden.

Lenkungsschlupf: Vorder- und Hinterräder sind, wie in Abbildung 3.13 dargestellt, durch den Antriebstrang fest miteinander verbunden. Die Lenkachse des Fahrzeuges steht senkrecht zur Fahrbahn, der Lenkrollradius r_0 beträgt $232,25\,\mathrm{mm}$. Werden die Räder eingelenkt, bewegen sich, wie in Abbildung 3.14 dargestellt, die Radaufstandspunkte der Vorder- und Hinterräder zueinander. Die Räder können diese Bewegung aber nicht durch Rollen ausgleichen, es entsteht Schlupf. Da die Lenkachse als Teil des Getriebes nicht verändert werden darf, beteht keine Möglichkeit diesen Schlupf zu minimieren. Aus Sicherheitsgründen und um den Verschleiß gering zu halten, ist der Druck in den Lenkzylindern begrenzt. Damit wird verhindert, dass ein ungewolltes Lenken im Stand stattfindet, da dem Fahrer durch die Joystick Steuerung die Geradeausstellung der Räder nicht bekannt ist. Somit findet die Relativbewegung nicht durch Gleiten der Räder auf der Fahrbahn, sondern durch Schlupf während der Fahrt statt.

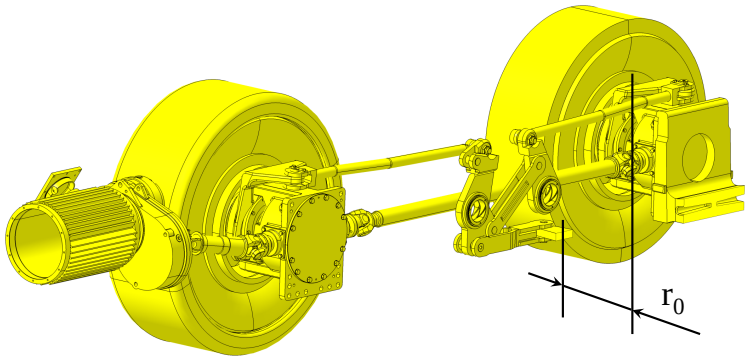


Abbildung 3.13.: Antriebstrang und Lenkrollradius

Querschlupf / Reifenseitenkraft: Die durch den Schräglaufwinkel des Rades verursachte Reifenseitenkraft wird wegen der geringen Fahrgeschwindigkeit von $10\,\mathrm{km/h}$ nicht berücksichtigt.

Schlupf durch Kurvenfahrt: Eine Ackermann Lenkung ist dadurch gekennzeichnet, dass bei geringen Geschwindigkeiten und Fahrt auf einem Kreisbogen alle Räder schräglauffrei rollen. Das bedeutet, dass sich die Drehachsen aller Räder im Bahnmittelpunkt C treffen müssen, Abbildung 1.10, damit die Räder kräfte- und schlupffrei auf der Fahrbahn rol-

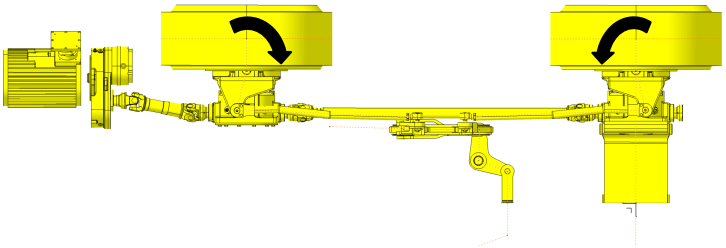


Abbildung 3.14.: Gegenläufige Lenkbewegung

len. Aus fahrdynamischen Gründen werden PKW oder Rennfahrzeuge meist nicht exakt nach Ackermann ausgelegt. LKW und Busse werden allerdings exakt nach Ackermann ausgelegt, um Verschleiß, Kraftstoffverbrauch und Kosten gering zu halten, [Wol09, S. 101 ff].

Definition für den ebenen Fall: Zur Formulierung des charakteristischen Parameters, der beschreibt wie gut die Ackermann Bedingung erfüllt ist, soll folgender Gedankengang dienen: Für ein ebenes Einspurmodell, Abbildung 3.15, ist die Ackermann Bedingung immer erfüllt, da sich die Drehachsen der beiden Räder immer im Punkt C schneiden, [RB00].

Legt man diesen Gedanken auf das ebene Modell mit vier Rädern, Abbildung 3.16, um, müssten sich linke und rechte Räder jeweils um einen eigenen Bahnmittelpunkt C_i drehen. Als Größe $L_{Ackermann}$, der die Ackermann Bedingung beschreibt, wird nun der Abstand dieser beiden Punkte C_1 und C_2 gewählt.

$$L_{\text{Achermann}} = \overline{C_1 C_2} \tag{3.5}$$

Natürlich ist diese Definition in gewisser Weise willkürlich, da auch die Schnittpunkte C_3 und C_4 der Vorder- und Hinterräder gebildet, und der Abstand dieser Punkte als charakteristischer Parameter gewählt werden kann.

Es wird die erste Definition gewählt, da angenommen wird, dass der auf diese Weise ermittelte Wert von $L_{Ackermann}$ stets der größere ist.

Als charakteristischer Parameter, der in der Optimierungsfunktion bewertet wird, wird

$$J_{\text{Ackermann}} = \int_0^{21.5} L_{\text{Ackermann}} \mathrm{d}\delta \tag{3.6}$$

definiert.

Das Zielverhalten dieses charakterischen Parameters ergibt sich aus der Überlegung, dass bei einer Ackermann Lenkung die Drehpunkte C_1 und C_1 in einem Punkt C zusammenfallen, $L_{Ackermann}$ also bei jedem Lenkwinkel null ist.

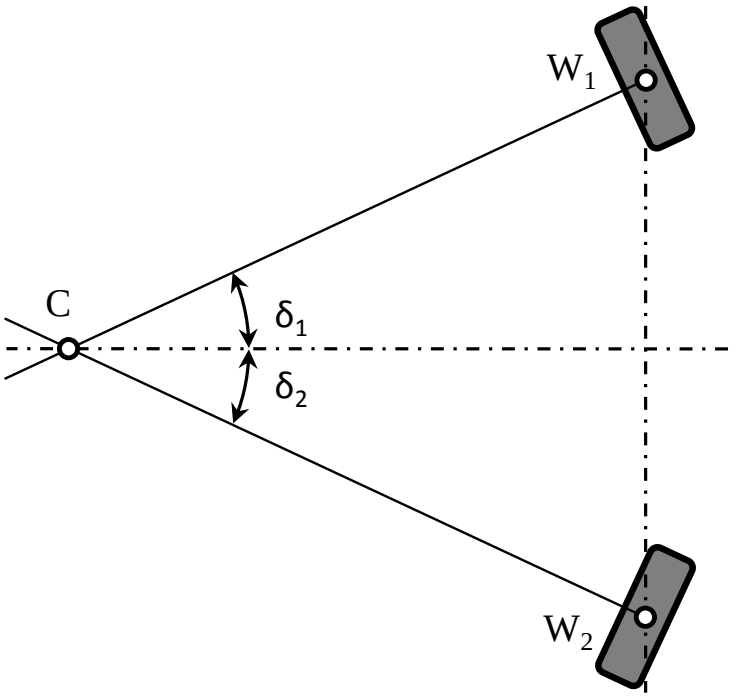


Abbildung 3.15.: Ebenes Einspurmodell

Diese Definition für $J_{Ackermann}$ lasst sich anwenden, wenn sich die Kippachse in waagrechter Position befindet. Ist die Kippachse um den Winkel φ gedreht, lässt sich diese Definition nicht mehr anwenden. Daher wird hier eine
Definition für den räumlichen Fall vorgestellt. Diese bezieht sich nicht speziell auf einen bestimmten Typ von Lenkung oder Radaufhängung. Ausgehend von den

- Geschwindigkeitsvektoren aller Radmittelpunkte $\mathbf{v}_i$ und den
- Positionen aller Radaufstandspunkte W_i

wird analog zum ebenen Fall ein charakterischer Parameter $J_{Ackermann}$ berechnet. Abbildung 3.17 zeigt die Radaufstandspunkte W_1 und W_2 in ein- bzw. ausgefedertem Zustand, außerhalb der ursprünglichen Fahrbahnebene.

Im ersten Schritt werden als gedankliche Hilfe alle Geschwindigkeiten $\mathbf{v}_i$ in die Radaufstandspunkte W_i verschoben, so als würden die Räder auf der Fahrbahn gleiten. In jeden Radaufstandspunkt W_i wird eine Ebene E_i, normal auf $\mathbf{v}_i$ gelegt. Eine Drehachse, um die sich das Rad dreht, muss immer in dieser Ebene liegen. Danach wird die Schnittgerade G_1 aus den Ebenen E_1 und E_2 ermittelt. Die Radaufstandspunkte W_1 und W_2 drehen sich um diese Gerade. Analog wird G_2 aus E_3 und E_4 ermittelt. Die Geraden G_1 und G_2 durchstoßen die ursprüngliche Fahrbahnebene, x-y-Ebene, in den Punkten C_1 und C_2. Im Idealfall einer Ackermannlenkung fallen G_1 und G_2 in einer Geraden zusammen. Im

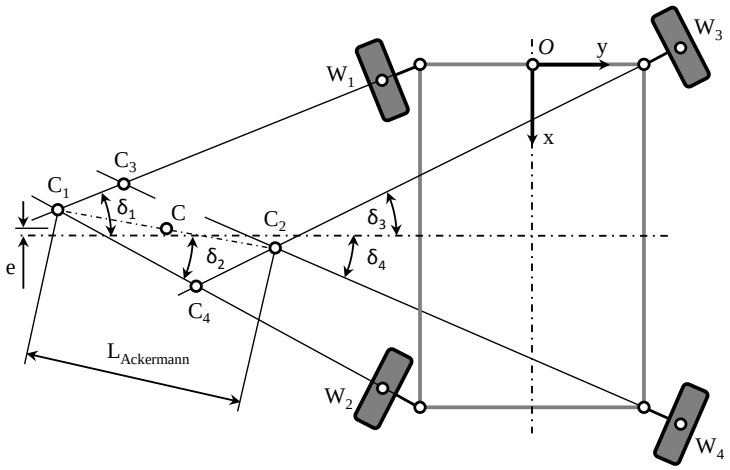


Abbildung 3.16.: Geometrie der Ackermann Bedingung

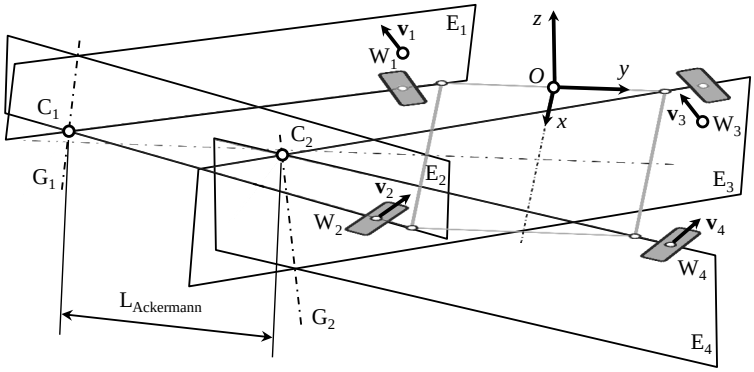


Abbildung 3.17.: Definition $L_{Ackermann}$ für den räumlichen Fall

allgemeinen wird das nicht der Fall sein und die beiden Geraden windschief zueinander liegen.

Als Größe, welche die Ackermann Bedingung für den räumlichen Fall beschreibt, wird nun der Abstand der beiden windschiefen Geraden G_1 und G_2 in der x-y-Ebebe gewählt. $L_{\text{Ackermann}}$ wird analog zum ebenen Fall definiert:

$$L_{\text{Ackermann}} = \overline{C_1 C_2}$$

Die Schnittgeraden lassen sich ebenfalls durch Verschneiden der Ebenen E_1 und E_4 bzw. E_2 und E_3 ermitteln. Analog zum ebenen Fall wird hier die erste Variante gewählt da

angenommen wird, dass der auf diese Weise ermittelte Wert von $L_{Ackermann}$ der größere ist.

Als charakteristischer Parameter, der in der Optimierungsfunktion bewertet wird, wird

$$J_{\text{Ackermann}} = \int_0^{21.5} L_{\text{Ackermann}} \mathrm{d}\delta \tag{3.7}$$

definiert.

Der ebene Fall stellt außerdem eienen Sonderfall des räumlichen Falles dar. Es wird also mit der Berechnungsmethode des räumlichen Falles jede Stellung der Kippachse berechnet.

Das Zielverhalten ist für den räumlichen Fall ebenfalls gleich wie im ebenen Fall.

3.2.3.2. Lenkwinkeldifferenz

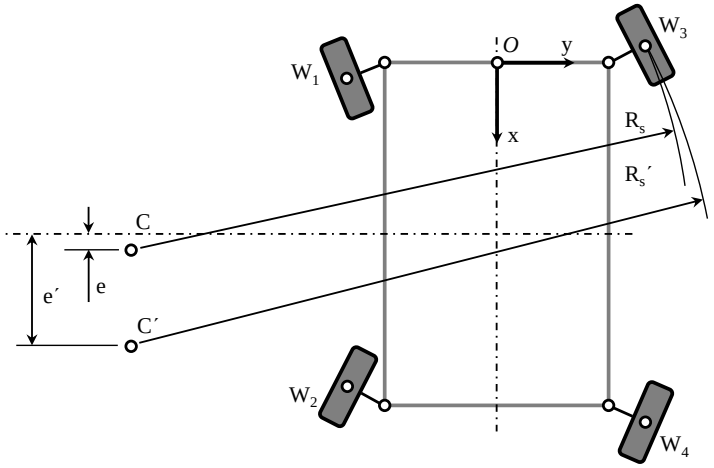


Abbildung 3.18.: Vergrößerung des max. Kurvenradius

Für die hier angestellten Betrachtungen wird angenommen, dass der tatsächliche Bahnmittelpunkt des Fahrzeuges genau in der Mitte der Strecke $\overline{C_1C_2}$ liegt, Abbildung 3.16. Die Punkte C, C_1 und C_2 liegen nicht stets auf der Fahrzeugquerachse, sondern können je nach Konstruktion davon abweichen. Insbesondere wenn in der MuBOS Optimierungsfunktion nur die Ackermann Bedingung berücksichtigt wird, ist nicht sichergestellt, dass sich der Abstand e des Punktes C_1 von der Fahrzeugquerachse nicht vergrößert. Das hat im Grunde genommen keine negativen Folgen im Sinne der Ackermann Bedingung. Allerdings vergrößern sich mit steigender Exzentrität e die maximalen Kurvenradien, Abbildung 3.18.

Der maximale Einschlagwinkel ist mit $\delta = 21,5\,^\circ$ beschränkt. Ist, wie in Abbildung 3.19 dargestellt, das Vorderrad ganz eingeschlagen und der Lenkwinkel $\delta_2 = \delta_1$, bzw. $\delta_2 > \delta_2{}'$,

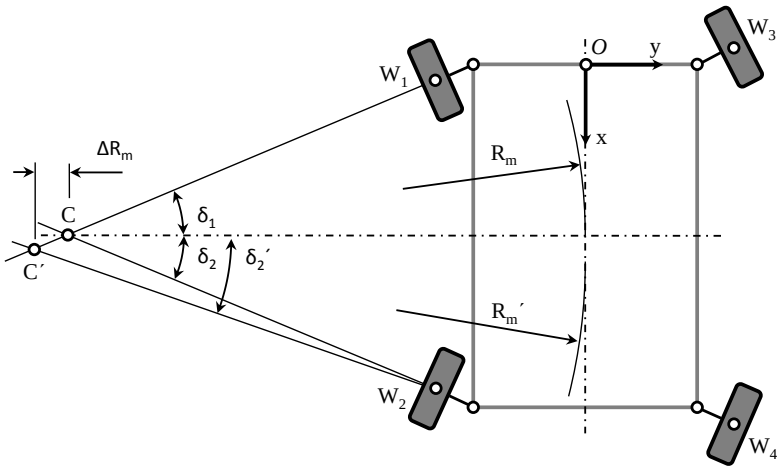


Abbildung 3.19.: Bahnradius

verschiebt sich C zu C´. Der Kurvenradius R_m vergrößert sich dabei um den Betrag ΔR_m auf R_m´. Um einen möglichst geringen Kurvenradius und die Kurventauglichkeit des Fahrzeuges sicherzustellen, wird daher gefordert dass $\delta_1 = \delta_2$ gilt. Als Größe, welche diese Forderung beschreibt, wird daher

$$\Delta\delta = \delta_1 - \delta_2$$

Als charakteristischer Parameter, der in der Optimierungsfunktion bewertet wird, wird

$$J_\mathrm{Lenkwinkeldifferenz} = \int_0^{21.5} \Delta\delta \mathrm{d}\delta \tag{3.8}$$

eingeführt. Die Zielfunktion ist analog zu $L_\mathrm{Ackermann}$ stets null.

3.2.4. Sensibilitätsanalyse

In der MuBOS Model-Parameter Datei werden die Gelenkpunkte in einer Structure, Matlab® Code A.2, abgelegt. Abbildung 3.26 zeigt die Gelenkpunkte der Lenkung die im Optimierungsprozess überhaupt verändert werden dürfen. Die Nummerierung erfolgt gemäß der Position in der MuBOS-Structure.

Die Sensibilitätsanalyse wird wie in Abschnitt 2.3.4 beschrieben durchgeführt. Die Position der Gelenkpunkte 7, 8, 9, 10, 12, 13, 14, 15, 16 und 17 wird in jeder Koordinatenrichtung von -10 bis $+10$ mm variiert. Die Ergebnisse dieser Analyse sind für jeden Gelenkpunkt im Abschnitt A.2 dargestellt. Die Simulationsergebnisse jedes Gelenkpunktes werden für jeden der beiden charakteristischen Parameter bewertet. Dabei wird ein Gelenkpunkt nach folgenden Kriterien eingestuft:

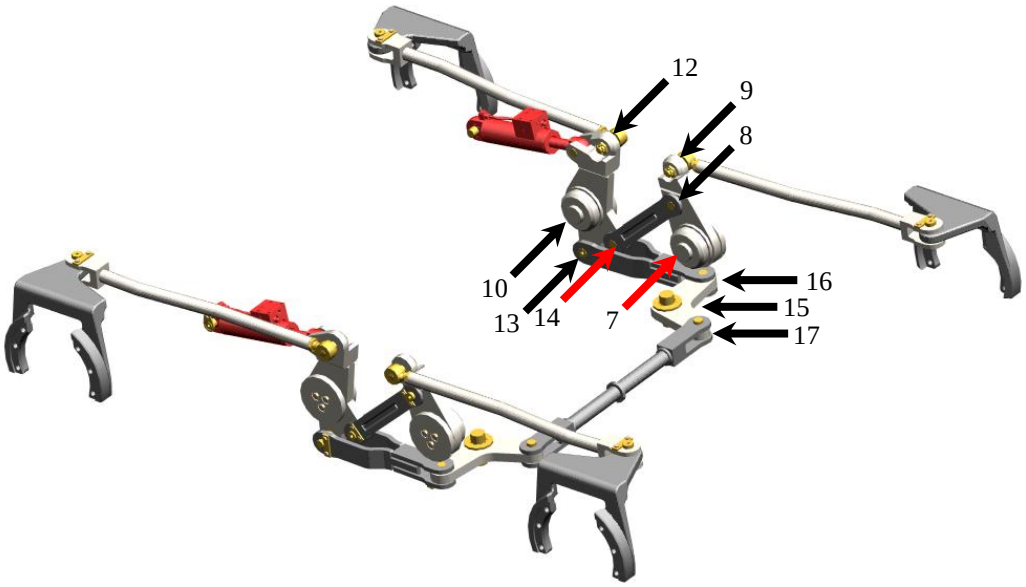


Abbildung 3.20.: Optimierbare Gelenkpunkte

- +: Eine starke Reaktion des charakteristischen Parameters ist zu beobachten.
- ∼: Eine geringe Reaktion des charakteristischen Parameters ist zu beobachten.
- -: Praktisch keine Reaktion des charakteristischen Parameters ist zu beobachten.

Tabelle 3.3 zeigt die Ergebnisse der Bewertung:

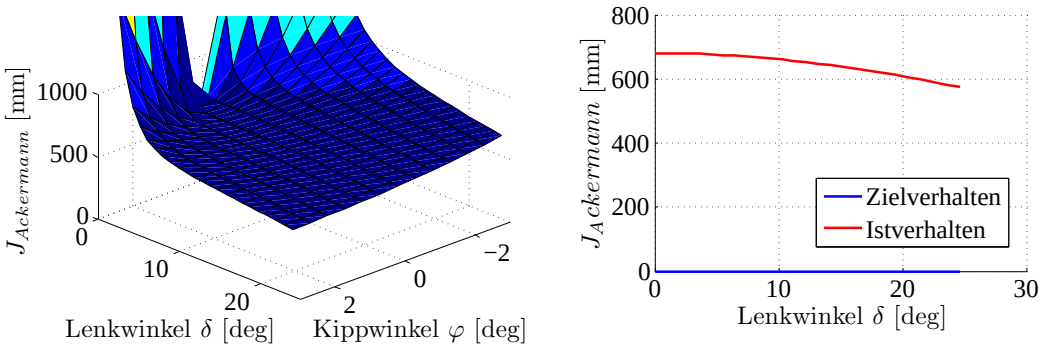
Auffällig ist, dass die Variation der Gelenkpunkte 9 und 12 nur sehr geringe Reaktionen der charakteristischen Parameter hervorrufen. Das lässt sich durch die große Länge der Lenkstangen, (6) und (7) in Abbildung 3.10, erklären. Um vergleichbar große Reaktionen wie andere Gelenkpunkte hervorzurufen, müssten die Positionen dieser Punkte um deutlich mehr als 10 mm verschoben werden. Daher werden nur die Gelenkpunkte 7, 8,10,13,14,15,16 und 17 zur Optimierung des Mehrkörpersystems herangezogen.

3.2.5. Ergebnisse

Eine erste Simulation des MKS-Systems zeigt das ursprüngliche Verhalten der charakteristischen Größen $L_{\text{Ackermann}}$ und $L_{\text{Lenkwinkeldifferenz}}$.

Gelenk	$J_{\text{Ackermann}}$	$J_{\text{Lenkwinkeldifferenz}}$
7	-	+
8	∼	+
9	∼	-
10	+	+
12	-	-
13	+	-
14	∼	+
15	+	-
16	+	-
17	+	-

Tabelle 3.3.: Bewertung der Sensibilitätsanalyse



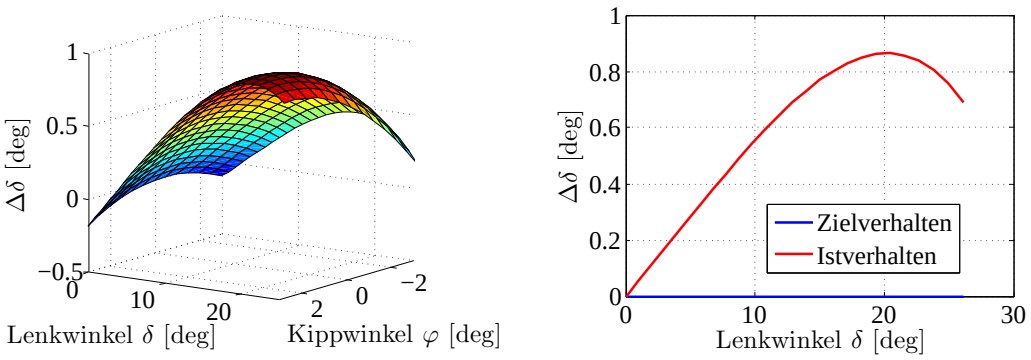
(a) (b)

Abbildung 3.21.: Ursprüngliches Verhalten von $J_{\text{Ackermann}}$

In Abbildung 3.21(a) ist die charakteristische Größe $L_{\text{Ackermann}}$ über dem Lenkwinkel δ und dem Kippwinkel der Vorderachse φ dargestellt. Der Parameter zeigt, bei einem Kippwinkel von $\varphi = 0\,\text{mm}$ relativ gleichbleibende Werte zwischen 580 und 680 mm über dem Lenkwinkel δ, Abbildung 3.21(b). Wird die Kippachse gedreht, steigt $L_{\text{Ackermann}}$ bei kleinen Lenkwinkeln, $\delta < 5\,°$, deutlich über 1000 mm an, was bei den damit verbundenen hohen Bahnradien vernachlässigbar bleibt. Für diese erste Simulation wurde der Größenordnungsfaktor mit $w_{om,1} = 1$ gewählt, vgl. Kostenfunktion 2.1 und Mechanism_Parameter Datei A.4. Der Wert des charakteristischen Parameters dieser Simulation beträgt:

$$J_{\text{Ackermann}} = 2122 \tag{3.9}$$

In Abbildung 3.22(a) ist die Lenkwinkeldifferenz $\Delta\delta$ über dem Lenkwinkel δ und dem Kippwinkel der Vorderachse φ dargestellt. Der Parameter steigt von ca. $0\,°$ bei $\delta = 0\,°$,



(a) (b)

Abbildung 3.22.: Ursprüngliches Verhalten von $J_{\text{Lenkwinkeldifferenz}}$

auf 0.85 ° bei $\delta = 20\,°$ an und fällt danach wieder ab. Über dem Kippwinkel φ ändert sich dieser charakteristische Parameter vergleichsweise geringfügig. Auch hier wurde der Größenordnungsfaktor mit $w_{om,2} = 1$ gewählt. Der Wert des charakteristischen Parameters dieser Simulation beträgt:

$$J_{\text{Lenkwinkeldifferenz}} = 0.569 \tag{3.10}$$

Mit diesen Ergebnissen können die Einstellungen für den Optimierungsprozess definiert werden. Tabelle 3.4 zeigt die Werte der charakteristischen Parameter und die gewählten Werte der Faktoren $w_{om,i}$ und w_i. Für den ersten Versuch der Optimierung werden die charakteristischen Parameter als gleichwertig betrachtet, die Größenordungsfaktoren werden so gewählt, dass sich in der Kostenfunktion, 2.1, dieselbe Größenordnung ergibt.

	$J_{\text{Ackermann}}$	$J_{\text{Lenkwinkeldifferenz}}$
Wert	2122	0.569
$w_{om,i}$	0.1	10^3
w_i	0.5	0.5

Tabelle 3.4.: Parameterwerte und Gewichtungsfaktoren

Die Abbruchkriterien, siehe Abschnitte 2.3.5 und 2.3.1.3 und Mechanism_Parameter Datei A.4, werden folgendermaßen definiert:

Die Ergebnisse dieser ersten Optimierung sind in Abbildung 3.23 dargestellt. Die grüne Linie stellt jeweils das Anfangs-, die rote Linie das optimierte und die blaue Linie das Zielverhalten dar. In jeder Iteration des Opimierungsprozesses wird das aktuelle Verhalten grau dargestellt. Dabei zeigt sich ein gegenläufiges Verhalten in der simultanen Optimierung der beiden charakteristischen Parameter. Die Funktion von $L_{\text{Ackermann}}$ steigt

Abbruchkriterium	Wert
$Max\_Function\_Evals$	2000
$TolFun$	10^{-2}
$TolX$	10^{-3}

Tabelle 3.5.: Abbruchkriterien Shuttle Car

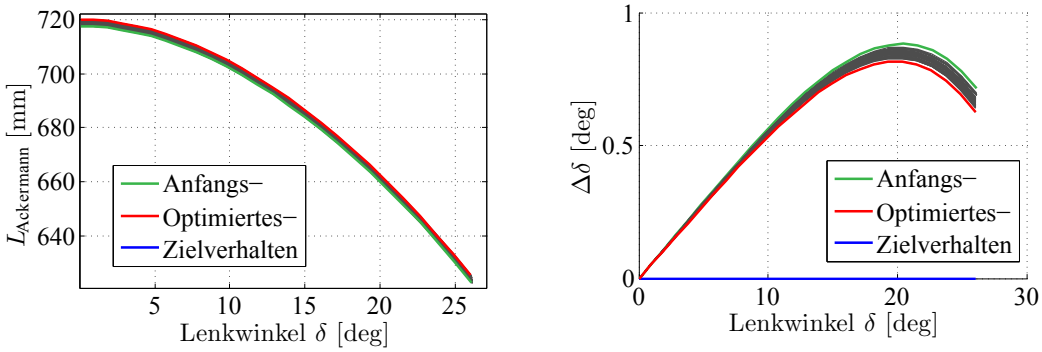


Abbildung 3.23.: Erste, gleichzeitige Optimierung

zu immer höheren Werten, Abbildung 3.23(a), während sich die Lenkwinkeldifferenz $\Delta\delta$ der Zielfunktion nähert, Abbildung 3.23(b). Wird der Gewichtungsfaktor w_1 entsprechend erhöht, lässt sich $L_{\text{Ackermann}}$ der Zielfunktion annähern. Die Lenkwinkeldifferenz steigt dabei allerdings auf Werte von $\Delta\delta = 4.4\,^\circ$ an. Das entspricht einer Erhöhung des Bahnradius um $\Delta R_m = 388\,\text{mm}$.

Mit dieser Vorgehensweise ist es also nicht möglich, beide charakteristischen Parameter geichzeitig zu optimieren. Daher wird die Optimierungs des MKS-Systems in zwei Schritten durchgeführt und, unter Berücksichtigung der Bewertung in Tabelle 3.3, die Gelenkpunkte den charakteristischen Parametern zugeordnet.

Optimierung der Lenkwinkeldifferenz: Hier werden die Gelenkpunkte 7, 8, 10 und 14, Abbildung 3.26 optimiert. Damit wird erreicht, dass die Lenkwinkeldifferenz $\Delta\delta = \delta_1 - \delta_2$ von Vorder- zu Hinterrad während der Lenkbewegung verringert wird. Die Lenkwinkeldifferenz von linken zu rechten Rädern, $\delta_1 - \delta_3$, bzw. $\delta_2 - \delta_4$, wird dadurch nicht direkt, die Ackermann Abweichung $L_{\text{Ackermann}}$ wird dabei nicht wesentlich beeinflusst.

Der Optimierungsprozess für diesen charakteristischen Parameter ist in Abbildung 3.24 dargestellt. Die Lenkwinkeldifferenz $\delta_1 - \delta_2$ wird dabei von 0.8 auf unter $0.1\,^\circ$ verringert.

Optimierung der Ackermann Abweichung: Hier werden die Gelenkpunkte 13, 15, 16 und 17 optimiert. Das sind die Gelenkpunkte der Spurstange und der Spurhebel,

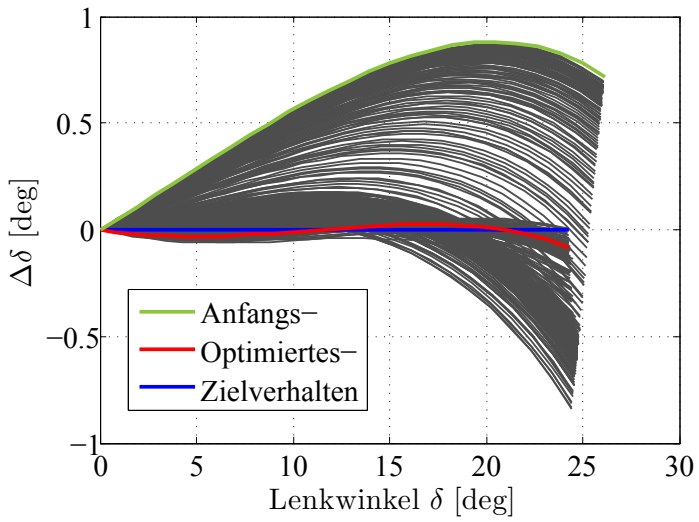


Abbildung 3.24.: Optimierungsprozess Lenkwinkeldifferenz

welche rechte und linke Seite des Lenksystems verbinden. Somit wird in diesem Optimierungsschritt das Verhalten der Lenkwinkeldifferenz $\Delta\delta$ nicht mehr verändert.

Der Optimierungsprozess der Ackermann Abweichung ist in Abbildung 3.25 dargestellt. Die Abweichung $L_{\text{Ackermann}}$ wird dabei von 700 auf unter 100 mm verringert.

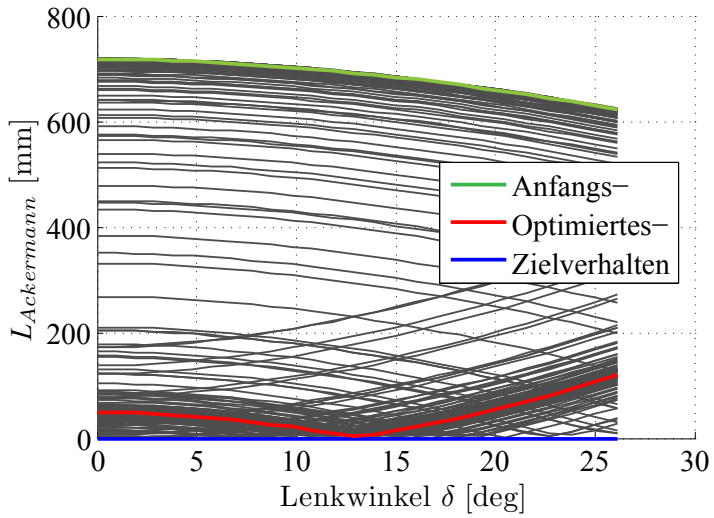


Abbildung 3.25.: Optimierungsprozess Ackermann

Abbildung 3.26 und 3.27 zeigen das Skelett im CAD-Modell. Die optimierten Gelenkpo-

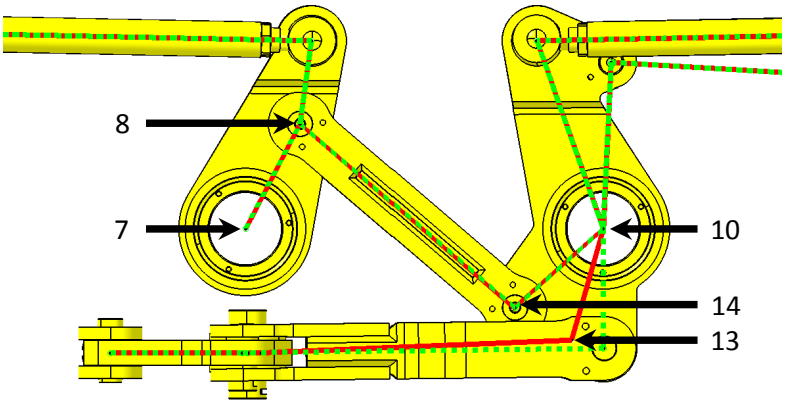


Abbildung 3.26.: Optimierte Gelenkpositionen

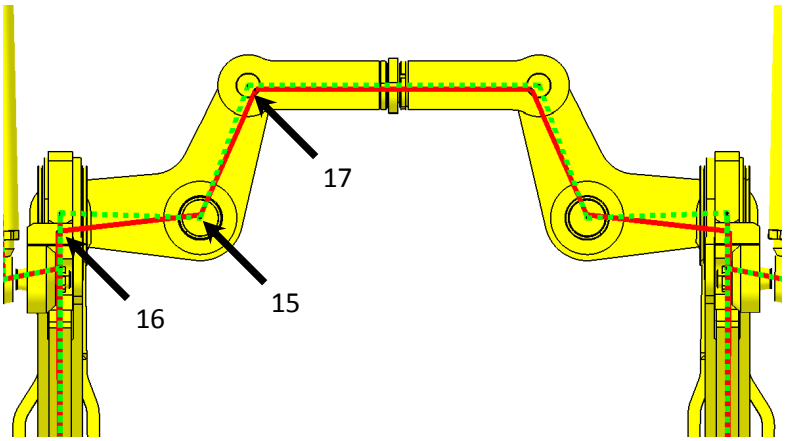


Abbildung 3.27.: Optimierte Gelenkpositionen Spurhebel

sitionen werden durch das rote Skelett und die ursprünglichen Gelenkpositionen durch das grün strichlierte Skelett dargestellt. Dabei sind nur bei den Gelenken 13, 15, 16 und 17 die Verschiebungen deutlich sichtbar. Die folgende Tabelle zeigt in welche Koordinatenrichtungen und um welchen Betrag die optimierten Gelenke verschoben wurden.

Die teilweise geringen Verschiebungen der Gelenkpunkte legen nahe, eine Toleranzanalyse durchzuführen. Dabei wäre festszustellen, ob eine derartige Neupositionierung der

Gelenkpunkt Nr.	Δx	Δy	Δz
7	0.86 mm	0 mm	0.29 mm
8	−0.05 mm	0 mm	0.1 mm
10	0.69 mm	0 mm	0.25 mm
13	−50.33 mm	0 mm	12.9 mm
14	−1.94 mm	0 mm	−0.34 mm
15	−5.15 mm	0.73 mm	0 mm
16	30.16 mm	−3.59 mm	0 mm
17	6.17 mm	12.34 mm	0 mm

Tabelle 3.6.: Verschiebung optimierter Gelenkpunkte

Gelenke sinnvoll ist oder ob nicht besser weniger Gelenke optimiert werden. diese dann aber um höhere Beträge verschoben werden. Eine solche Analyse konnte aus Zeitgründen jedoch nicht durchgeführt werden.

4. Zusammenfassung und Ausblick

Ziel dieser Arbeit war es, eine Methodik zur Optimierung von Mehrkörpersystemen zu entwickeln. Dabei sollte die Methodik auf beliebige Mehrkörpersysteme anwendbar sein.

Im Sinne einer Co-Simulation zwischen Catia V5® und Matlab/Simulink® wurde ein Matlab® basiertes Programm, MuBOS (**M**ulti**B**ody **O**ptimization **S**ystem), entwickelt, das den Konstruktions- und Simulationsprozess in einer frühen Produktentwicklungsphase sinnvoll unterstützt. Dabei werden Geometriedaten und Masseneigenschaften vom CAD-System automatisiert auf ein Mehrkörpersystem (MKS) in Matlab / Simmechanics® übertragen. Dort kann das Modell simuliert und anhand charakteristischer Parameter, wie Sturz- oder Vorspurwinkel, optimiert werden. Im Optimierungsprozess werden, unter Verwendung eines Nelder-Mead Simplex Algorithmus, die Gelenkpositionen des Systems variiert und so nach einem Optimum der charakteristischen Parameter gesucht. Diese können für jede Anwendung frei definiert werden. Um die Integration möglichst vieler Funktionen sicherzustellen, können beliebig viele charakteristische Parameter definiert und optimiert werden. Im Anschluß an die Optimierung werden die Geometriedaten, ebenso automatisiert, wieder in das CAD-Modell übertragen. Damit ist es möglich, auch sehr kurzfristige Änderungen in der Konstruktion rasch mit entsprechenden Berechnungen zu untermauern.

Als Anwendungsbeispiel zur Validitierung von MuBOS wurde eine Schwertlenker Hinterachse eines PKW betrachtet. Dabei wurde die Position eines Querlenkers aufgrund eines eingebauten Radnabenmotors verändert. Ausgehend von dieser neuen Lage wurde das Mehrkörpersystem so optimiert, dass das ursprünliche Verhalten von Sturz und Vorspurwinkel wieder erreicht wurde.

Ein zweites Anwendungsbeispiel zeigt wie flexibel MuBOS auch auf in der Fahrzeugtechnik seltene Sonderfahrzeuge im Untertagbau angewendet werden kann. Die Allradlenkung eines solchen Fahrzeuges wird, um Reifenverschleiß zu minimieren und Kurventauglichkeit sicherzustellen, nach Ackermann ausgelegt. Dabei wird die Ackermann Bedingung sowohl für Fahrt auf ebener Fahrbahn als auch für den räumlichen Fall bei Fahrt mit verdrehter Kippachse formuliert.

Zusammenfassend kann gesagt werden dass mit MuBOS ein Programm zur Verfügung steht, das die konstruktiven Werkzeuge von Catia V5® mit den vielfältigen Berechnungsmöglichkeiten von Matlab / Simulink® verbindet. Somit schlägt es eine Brücke zwischen Konstruktion und Simulation / Berechnung. Mit MuBOS kann rasch und flexibel auf grundsätzliche Fragestellungen im Konstruktions- und Simulationsprozess eingegangen werden.

Die Aufgabenstellungen der beiden Anwendungsbeispiele erforderten nur rein kinematische Vorwärtssimulationen. Mit Matlab / Simmechanics® lassen sich jedoch auch dynamische Vorwärts- und Rückwärtssimulationen durchführen und so Geschwindigkeiten, Beschleunigungen und Kräfte im System berechnen. Weiters existiert, als Schnittstelle zwischen Mehrkörpersystem und CAD Modell, derzeit nur eine Schnittstelle zu Catia V5®. Beide Themen sind Gegenstand weiterführender Projekte.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[BS07] H. H. Braess and U. Seiffert. *Vieweg Handbuch Kraftfahrzeugtechnik*. Vieweg Teubner, 2007.

[Dür11] J. Dürnberger. Fahrdynamische Untersuchung einer elektrischen Einzelradlenkung. Master Thesis, 2011. Technische Universität Graz.

[HE07] B. Heissing and M. Ersoy. *Fahrwerkhandbuch Grundlagen, Fahrdynamik, Komponenten, Systeme, Mechatronik, Perspektiven*. Vieweg+Teubner Verlag, 2007.

[HM11] B. Heißing and E. Metin. *Chassis Handbook: Fundamentals, Driving Dynamics, Components, Mechatronics, Perspectives*. Vieweg+Teubner, Wiesbaden, 2011.

[KR11] Wilfried Krug and Oliver Rose. *Simulation und Optimierung in Produktion und Logistik*. Springer, Berlin Heidelberg, 2011.

[MAT08] MATLAB. *version 7.7.0.471 (R2008b)*. The MathWorks Inc., Natick, Massachusetts, 2008.

[MJ08] A. Moaglio and C. Johnson. *Evolutionary Computation in Combinatorial Optimization*, pages 190–201. Springer Berlin Heidelberg, 2008.

[NM65] J.A. Nelder and R.A. Mead. A simplex method for function minimization. *Computer Journal 7*, pages 308–313, 1965.

[RB00] J. Reimpell and J. Betzler. *Fahrwerktechnik: Grundlagen*. Vogel Buchverlag, 2000.

[Roj12] Andres Eduardo Rojas Rojas. *Passenger Vehicles with In-Wheel Motors*. PhD thesis, Technische Universität Graz, 2012.

[San09] Sandvik Mining and Construction G.m.b.H. Sandvik Shuttle Car TC790 in Action. Broschüre, 2009.

[San12] Sandvik Mining and Construction G.m.b.H. Informationen und Daten im Zuge der Diplomarbeit. Unveröffentlicht, 2012.

[vS99] Reinhold von Schwerin. *Multibody System Simulation*. Springer-Verlag, Berlin Heidelberg New York, 1999.

[Wal06] H. Wallentowitz. *Vertikal- und Querdynamik von Kraftfahrzeugen*. fka - Forschungsgesellschaft Kraftfahrwesen mbH Aachen, Aachen - Germany, 2006.

[Wil10] J. E. Aponte Wilson. Development of a Suspension Kinematics Optimization

System - KOS . Master Thesis, 2010. Technische Universität Graz.

[Woe11] Christoph Woernle. *Mehrkörpersysteme*. Springer-Verlag, Berlin Heidelberg New York, 2011.

[Wol09] Wolfgang Hirschberg and Helmut M. Waser. Kraftfahrzeugtechnik. Vorlesungsskriptum, 2009.

A. Anhang

A.1. Matlab® Codes

Die folgenden Matlab® Codes stellen Beispiele von allen MuBOS-Parameter Dateien dar. Die Dateien sind teilweise unvollständing und dienen lediglich dem Verständnis. So sind z.B. in der Model Parameter Datei, Matlab® Code A.2, nur Daten eines Gelenkpunktes und eines Körpers angegeben.

Matlab Code A.1: Communication Parameter Datei

```
%% Parameter Fields List
Structure_Position = 'Parameters';
Structure_Fields   = {'Communication';
                      };

PreProcessing_FCN  = '';

%% Parameters definition

Communication.Matlab.File_Name = 'Multilink_Results'; % <String> Name of
                                          % the structure's results file;

Communication.CATIA_Wireframe.Name =...    %<String> Name of the workframe
    'MuBOS_Skelett_Controlblade.CATPart';  % CATIA's part which has to be
                                           % saved in the same directory
                                           % as the main product
Communication.CATIA.Work_File.Path =...
    'CAD_TEMPLATES\CAD_Model\Control_Blade'; % <String> Relative path to
                                             % the product file

Communication.CATIA.Work_File.File_Name =...        % <String> Name of the
    'SCHWERTLENKERACHSE_MODELL_DESIGN.CATProduct';  % CATIA V5 product

Communication.CATIA.visual_flag     = 1; % [-] Flag to indicate whether
                                         % the program should be visualized
                                         % or not: ≠ 1 -> Program will NOT
                                         % be visible

Communication.Excel.Work_File.Path = '\MuBOS\Source_Code_MuBOS\VBA_Excel';
                                         % <String> Relative path of the
                                         % EXCEL file containing VBA
                                         % scripts
```

```
33 Communication.Excel.Work_File.File_Name = 'COM_CATIA.xls'; % <String> Name
34                                          % of the EXCEL file
35 Communication.Excel.visual_flag         = 1; % [-] Flag to indicate whether
36                                          % EXCEL should be visualized
37                                          % or not: ≠ 1 -> Program will NOT
38                                          % be visible
```

Matlab Code A.2: Model Parameter Datei

```
1  %% Parameter Fields List
2  Structure_Position = 'Parameters';
3  Structure_Fields   = {'Model';
4                        };
5
6  PreProcessing_FCN  = 'Model_Parameters_PreProcFCN';
7
8  %% Depth of symetric joint relations
9  Model.Symetric_Joint_Relation_Depth             = 1; % <double> Depth of
10                                                % symetric joint relations.
11
12 %% Definitions of Model Parameters
13
14 %% 1) Definition of Parameter Steering_Axis_Front_Direction_x
15 Model.Parameter(1).Name                                = 'Radsturz';
16 Model.Parameter(1).Value                               = 0; % Value of the
17                                                   % parameter in the CAD model
18 Model.Parameter(1).Path_CATIA = strcat('SCHWERTLENKERACHSE_MODELL_',...
19     'DESIGN/Parameters/!Radsturz');               % <String> Catia path of the
20                                                   % parameter
21
22 %% 1) Definition of Joint W_Front
23 Model.Joint(1).Name                                    = 'Radmittelpunkt';
24                                                   % <String> Joint name
25 Model.Joint(1).Type                                    = 'Spherical'; % <String>
26                                                   % Type of simmechanics joint.
27 Model.Joint(1).Flag_Optimization                       = 0; % <int> Flag,
28                                                   % ==1: position can be optimized
29                                                   % ≠1: position is not optimized
30 Model.Joint(1).Optimized_Coordinate_Directions  = [1 1 1]; % <int> Vector,
31                                                   % [x y z] defining wich
32                                                   % coordinate directions can be
33                                                   % used for opimization.
34
35 Model.Joint(1).Position=[4385.48 -755 0]; % [mm] Joint position
36
37 Model.Joint(1).Path_CATIA = strcat('Schwertlenker_MODELL_DESIGN/MuBOS',...
38          '_Skelett_Controlblade/MuBOS_Skelett_Controlblade.1/Hardpoin',...
39          't_Positions/!Radmittelpunkt');  % <String> Path of the joint
40                                           % inside the CAD model.
41
42 Model.Joint(1).Available_Space.Flag             = 0; % Flag to indicate if
43                                           % Point is controlled for
```

```
                                        % Available Space
Model.Joint(1).Available_Space.Space_CATIA_Part = ''; % <String> Instance
                                        % name of the available space
                                        % part in the CATIA product
Model.Joint(1).Available_Space.Space_CATIA_Path = ''; % <String> Path of
                                        % the available space part in the
                                        % CATIA product

Model.Joint(1).Flag_Symetric                = 0; % Flag to indicate if
                                        % joint is symmetric
                                        % ==1: symmetric
                                        % ≠1: not symmetric
Model.Joint(1).Symetric_Joint               = '2'; % <string> Structure
                                        % position of symmetric joint
Model.Joint(1).Symetric_Plane               = 'xy'; % <sring> plane at
                                        % wich joint is mirrored.

Model.Joint(1).Associated_Bodies            = [1 3]; % <int> Structure
                                        % positions of bodies connected
                                        % to this joint.

%% Definition of Bodies

%% 1) Definition of Body 1
Model.Body(1).Name                          = 'Lenker_1'; % <string>
                                                  % Name of body
Model.Body(1).Mass.Value                    = 1;        % Mass of body
Model.Body(1).Mass.Units                    = 'kg';     % Units of mass
Model.Body(1).Inertia.Value                 = '[1 0 0;0 1 0;0 0 1]';
Model.Body(1).Inertia.Units                 = 'kg*m^2';

Model.Body(1).CG.Show_Port                  = 0;
Model.Body(1).CG.Port_Side                  = 'Left';
Model.Body(1).CG.Position                   = [1 1 1];
Model.Body(1).CG.Units                      = 'mm';

Model.Body(1).CS(1).Associated_Joint        = 6; % <int> Structure
                                        % position of joint connected to
                                        % this coordinate system.
                                        % 0 if no joint is connected.
Model.Body(1).CS(1).Show_Port               = 1; % <int> Set port
                                        % visiblility

Model.Body(1).CS(1).Position                = [-9.289 81.728 123.85];
                                        % Coordinate System position

Model.Body(1).CS(2).Associated_Joint        = 3; % <int> Structure
                                        % position of joint connected to
                                        % this coordinate system.
                                        % 0 if no joint is connected.
Model.Body(1).CS(2).Show_Port               = 1; % <int> Set port
                                        % visiblility

```

```
97 Model.Body(1).CS(2).Position                    = [29.698 352.184 117.781];
98                                              % Coordinate System position
```

Matlab Code A.3: Simulation Parameter Datei

```
1  %% Parameter Fields List
2  Structure_Position = 'Parameters';
3  Structure_Fields   = {'Simulation'...
4                        };
5
6  PreProcessing_FCN  = 'Simulation_Parms_PreProcFCN_MuBos';
7
8  %% Parameters definition
9  Simulation.t_transient = 0;          % [s] Time before reaching the steady
10                                      % state when the vehicle is running
11                                      % straight ahead over a flat surface
12 Simulation.Step_sim    = 0.002;      % [s] Simulation step-time
13 Simulation.t_sim       = 1;          % [s] Simulation time
14
15 Simulation.DebugMode   = 1;          % Flag to set debug mode in Catia.
16                                      % Joint positions are set in Catia at
17                                      % each iteration in the optimization
18                                      % process
19
20 Simulation.Shannon_frequency = 200; % [Hz] Desired highest frequency to be
21                                      % taken into account during simulation
22
23 Simulation.Simmech_Model_name        = 'Controlblade'; % <String> Name of
24                                      % the simmechanics model
25 Simulation.Simmech_Model_visibility  = 0; % Flag to visualize the
26                                      % simmechanics machine during
27                                      % simulation. ≠1: machine is not
28                                      % visualized.
29
30 Simulation.Simmech_Analysis_Mode            = 'Forward dynamics'; %
31 Simulation.Simmech_Machine_Environment_Path =...
32                         'Controlblade/Physical_Model/Machine Environment';
33
34 Simulation.Solver      = 'ode4';     % <String> Solver to be used
35
36 %% Definition of movement
37
38 Simulation.Main_Movement.Start = 70; % Start position of movement
39 Simulation.Main_Movement.End   = -70;% Final position of movement
```

Matlab Code A.4: Mechanism Parameter Datei

```
1  %% Parameter Fields List
2  Structure_Position = 'Parameters';
3  Structure_Fields   = {'Mechanism_Parameters'
4                        };
```

```
5
6  PreProcessing_FCN  = 'Optimization_Mechanism_Parameters_PreProcFCN_MuBos';
7
8  %% Parameters definition
9
10 %% Mechanism Parameter 1
11 Parameter(1).Function_name                      = 'Radsturz'; % <String>
12 Parameter(1).Optimization_flag                  = 1; % [int]
13                                 % ==1: Criterion is used for optimization
14                                 % ≠1: Criterion is not used
15 Parameter(1).Weighting_factor.order_of_magnitude = 0.001; % [-] Order of
16                                 % magnitude
17 Parameter(1).Weighting_factor.importance         = 1; % [-]Weighting factor
18
19 Parameter(1).Goal_Behavior.File_Name             = 'Radsturz'; % <string>
20
21 Parameter(1).Plot.plot_function                  = 1; % <Integer> matlab
22                                 % plot-function to be used. 1 = plot,
23                                 % 2 = plot3
24 Parameter(1).Plot.Title                          = 'Radsturz'; % <String>
25                                 % Title of the subplot for this parameter
26 Parameter(1).Plot.x_data = strcat('MuBOS_Structure.Outputs.Motion.',...
27     'Mechanism_Parameters.Parameter(1).Actual_Behavior'); % <String>
28                                 % Path of the data in the MuBOS structure
29 Parameter(1).Plot.x_label                        = 'Radsturz [deg]';
30                                 % <String> x-label for the plot.
31 Parameter(1).Plot.y_data = strcat('MuBOS_Structure.Outputs.Motion.',...
32     'Model_Output.Physical_Model_Signals.W_Point.z_W_Point'); % <String>
33                                 % Path of the data in the MuBOS structure
34 Parameter(1).Plot.y_label                        = 'Radhub [mm]';
35                                 % <String> y-label for the plot.
36 Parameter(1).Plot.z_data                         = ''; % <String> Path of
37                                 % the data in the MuBOS structure
38 Parameter(1).Plot.z_label                        = ''; % <String> Label
39                                 % for the plot.
40 Parameter(1).Plot.disp_plot_flag                 = 1; % Flag to display the
41                                 % criterion during the optimization process
42                                 % ==1: do plot ≠ 1: do not plot
43 Parameter(1).Plot.Dependent_Axis                 = 'x'; % Dependent Axis
44                                 % in the output plot.
```

Matlab Code A.5: Sensitivity Parameter Datei

```
1  %% Parameter Fields List
2  Structure_Position = 'Parameters';
3  Structure_Fields   = {'Sensitivity';
4                        };
5
6  PreProcessing_FCN  = 'Sensitivity_Parameters_PreProcFCN';
7
8  %% Parameters definition
9
```

```
Sensitivity.Variation_Flag = 1; % ==1: Fix Variation of joints
                                % ==0: Variation relative to available space
Sensitivity.Fix_Variation  = [-10:1:10]; % [mm] Range and step size of
                                % joint variation for sensitivity analysis
Sensitivity.Rel_Variation  = 0.1; % [-] Variation relative to dimensions
                                % of available space
```

Matlab Code A.6: Available Space Parameter Datei

```
%% Parameter Fields List
Structure_Position = 'Parameters';
Structure_Fields   = {'Available_Space';...
                      };
PreProcessing_FCN  = '';

%% Parameters definition


%%  Parameters for testing sphere generation
Available_Space.Sphere.Part_Name        = 'Test_Clash_Sphere'; % <String>
                                       % Name of the part to be generated
                                       %in CATIA

Available_Space.Sphere.Function.Matlab  = 'Sphere_Matlab_KOS'; % <String>
                                       % Name of the Matlab-Function to
                                       % execute the process

Available_Space.Sphere.Function.VBA     = 'Sphere'; % <String> Name of the
                                       % VBA-Function to execute the
                                       % process

Available_Space.Sphere.Flag             = 1; % [-] Flag to indicate the
                                      % visibility of the body:
                                      %     == 1 -> Body is visible
                                      %     ~= 1 -> Body is NOT visible

%% Parameters for creating clash detection in CATIA's model
Available_Space.Clash.Clash_Name        = 'Clash_Optimization_Routine';
                                      % <String> Name of the clash to be
                                      % created

Available_Space.Clash.Function.Matlab   = 'Clash_Free_Space_Matlab_KOS';
                                      % <String> Name of the Matlab-
                                      % Function to execute the process.

Available_Space.Clash.Function.VBA.Create_...
    Clash = 'Creating_Clash';         % <String> Name of the VBA-Function
                                      % to execute the process

Available_Space.Clash.Function.VBA.Checking_...
    Clash = 'Checking_Clash_FreeSpace'; % <String> Name of the VBA-Function
                                      % to check if is there a clash.
```

Matlab Code A.7: Optimization Parameter Datei

```
%% Parameter Fields List
Structure_Position = 'Parameters';
Structure_Fields   = {'Optimization'
                      };
PreProcessing_FCN  = 'Optimization_Parms_PreProcFCN_MuBos';

%% Parameters definition
% Optimization
Optimization.Options.Out_Available_Space_Penalty    = 100000;
                                    % [-]
Optimization.Options.Max_Function_Evals             = 50;
                                    % [-] Max evaluations for the
                                    % optimization function
Optimization.Options.TolFun                         = 1e-5;
                                    % [-] Termination tolerance on the
                                    % function value in the optimization
                                    % function
Optimization.Options.TolX                           = 1e-6;
                                    % [-] Termination tolerance on the
                                    % inputs variables of the optimization
                                    % function
Optimization.Options.Display                        = 'iter';
                                    % <String> Level of display -> 'off' |
                                    % 'iter' | 'diagnose' | {'final'}
```

Matlab Code A.8: Output Parameter Datei

```
%% Parameter Fields List
Structure_Position = 'Parameters';
Structure_Fields   = {'Outputs';
                      };

PreProcessing_FCN  = 'Output_Parameters_PreProcFCN';

%% Parameters definition
Outputs.Plots.Flag              = 1; % == 1: Display output plot during
                                     % simulation, optimization and
                                     %sensitivity analysis
                                     % ≠1: Do not display output plot.

%% Plots
Outputs.Plots.Titlesize         = 15; % Size of titles in created figures
Outputs.Plots.Suptitlesize      = 20; % Size of suptitles in figures
Outputs.Plots.Labelsize         = 10; % Size of labels in created figures
Outputs.Plots.Axessize          = 10; % Size of axes in created figures
Outputs.Plots.Linewidth         = 2;  % Width of lines in created figures
Outputs.Plots.Linewidth_thin    = 1;  % Width of thin lines in figures

%% Colors
Outputs.Plots.Colormap          = 'Winter';      % <String> Defines the
                                          % colormap to be used for output
```

```
25 Outputs.Plots.Color.Grey         = [0.3,0.3,0.3]; % Color grey
26 Outputs.Plots.Color.Red          = [1,0,0];       % Color red
27 Outputs.Plots.Color.Blue         = [0,0,1];       % Color blue
```

A.2. Sensibilitätsanalyse

In der Sensibilitätsanalyse wird die Position eines definierten Gelenkpunktes der Mehrkörpersystems in jeder der drei kartesischen Koordinatenrichtungen in einem vorher definierten Bereich variiert. In jeder Position wird einmal die Simulationsfunktion aufgerufen, und die charakteristischen Parameter berechnet. Das Ergebnis dieser Analyse ist die Variation der charakteristischen Parameter in jeder Koordinatenrichtung. Somit kann abgeschätzt werden, welchen Einfluss ein Gelenkpunkt auf einen charakteristischen Parameter hat.

Die folgenden Abbildungen zeigen die Ergebnisse der Sensibilitätsanalyse des Shuttle Car TC100 aus dem Anwendungsbeispiel in Abschnitt 3.2. Jeder ausgewählte Gelenkpunkt wurde in allen drei Koordinatenrichtungen mit einer Schrittweite von 1 mm von -10 bis $+10$ mm variiert. Die Nummerierung der Gelenkpunkte erfolgt gemäß der Position in der MuBOS-Structure, vgl. Matlab® Code A.2.

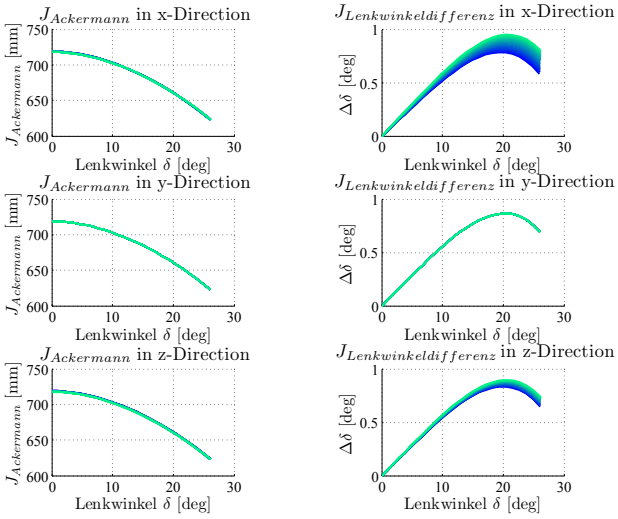


Abbildung A.1.: Sensibilitätsanalyse Gelenk 7

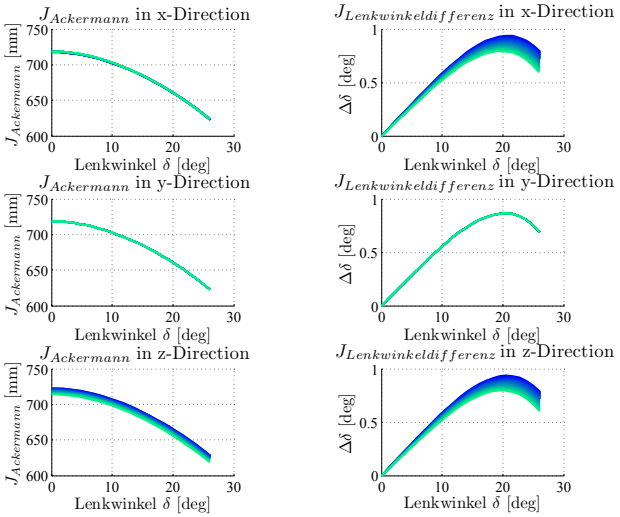


Abbildung A.2.: Sensibilitätsanalyse Gelenk 8

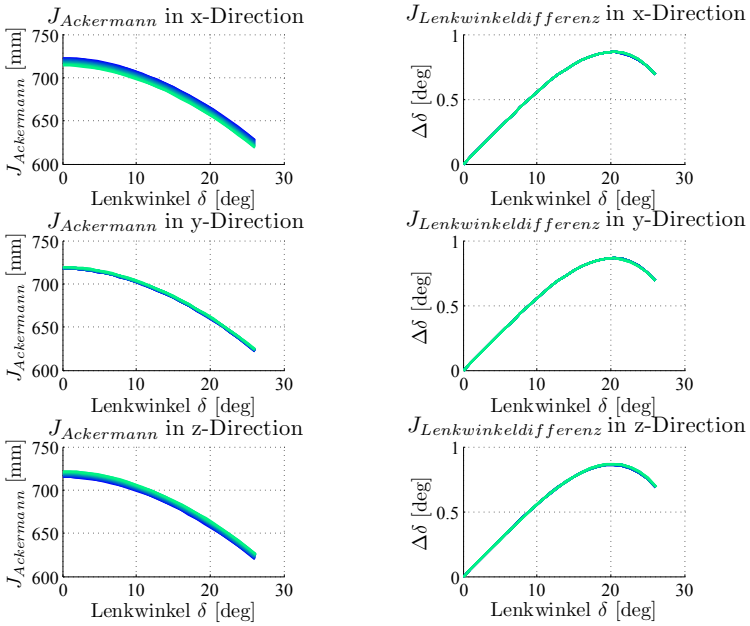


Abbildung A.3.: Sensibilitätsanalyse Gelenk 9

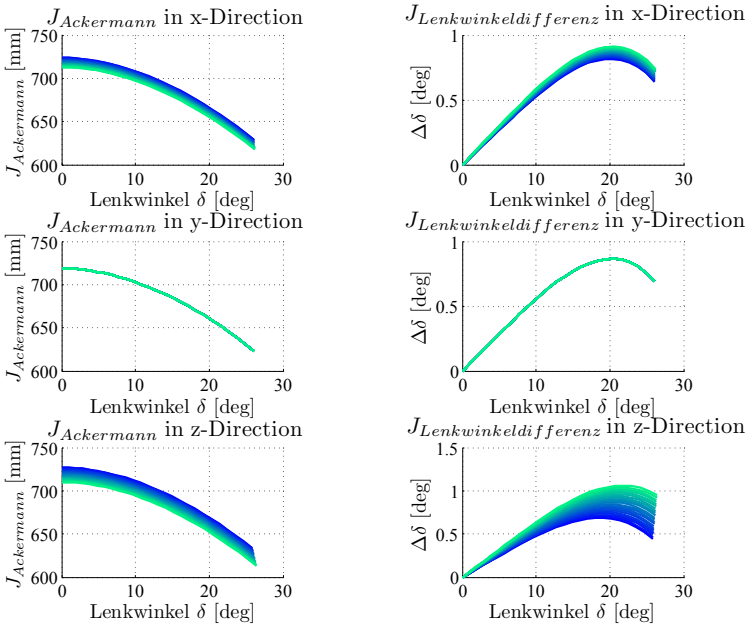


Abbildung A.4.: Sensibilitätsanalyse Gelenk 10

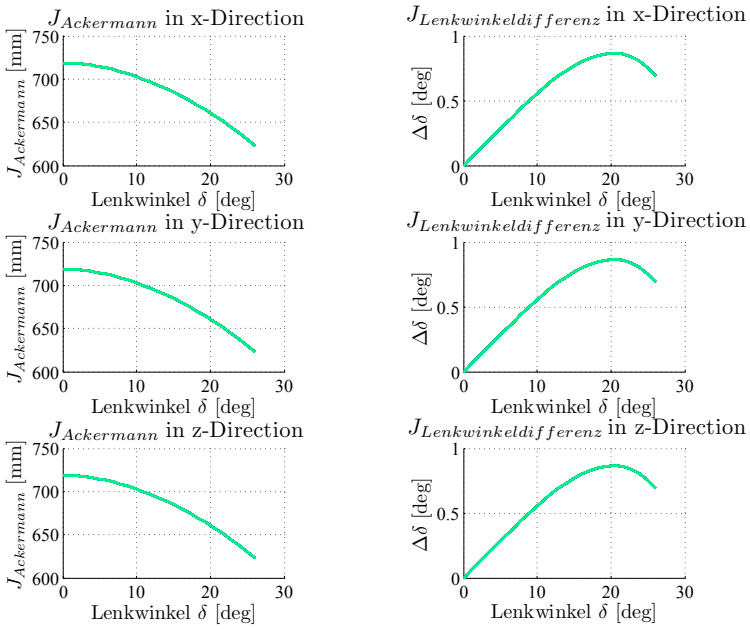


Abbildung A.5.: Sensibilitätsanalyse Gelenk 12

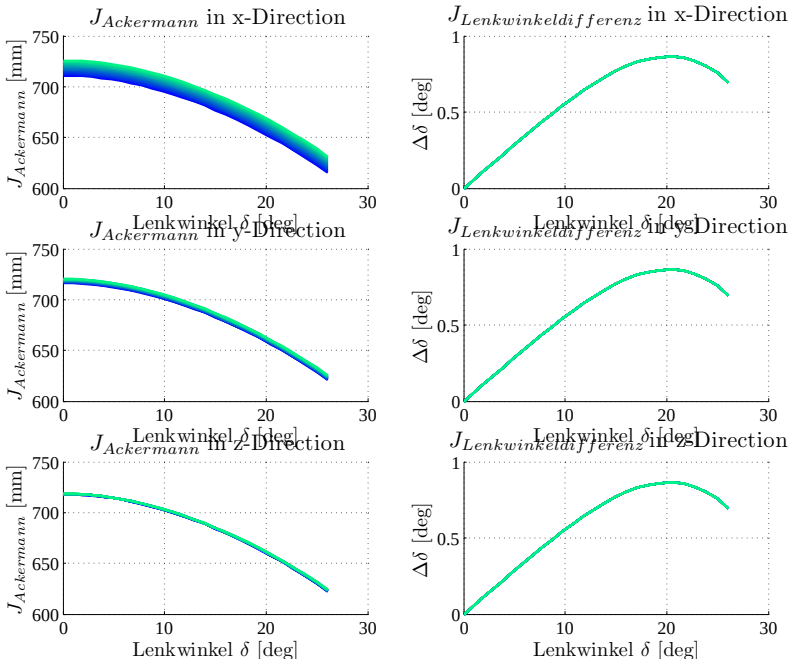


Abbildung A.6.: Sensibilitätsanalyse Gelenk 13

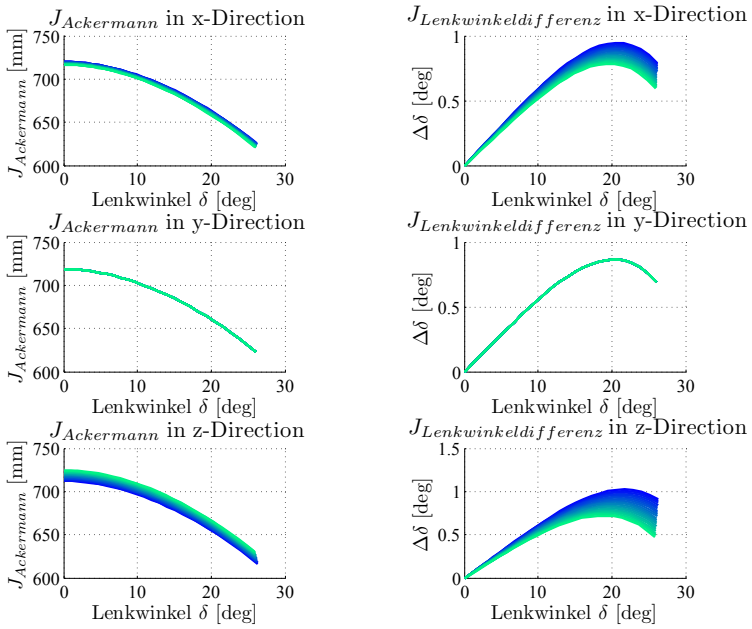


Abbildung A.7.: Sensibilitätsanalyse Gelenk 14

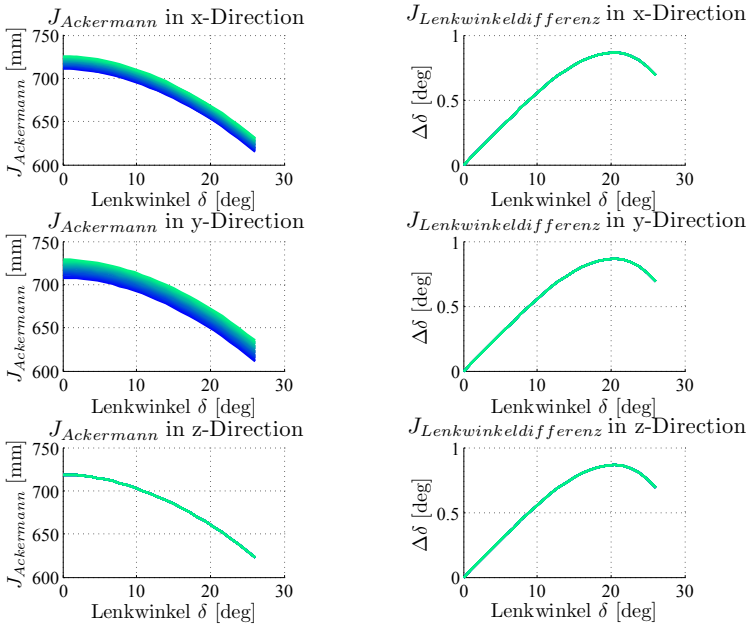


Abbildung A.8.: Sensibilitätsanalyse Gelenk 15

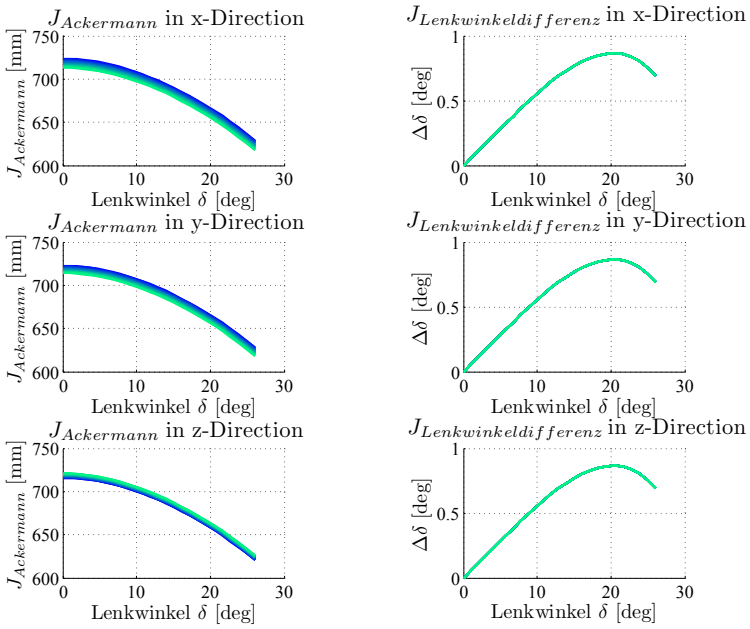


Abbildung A.9.: Sensibilitätsanalyse Gelenk 16

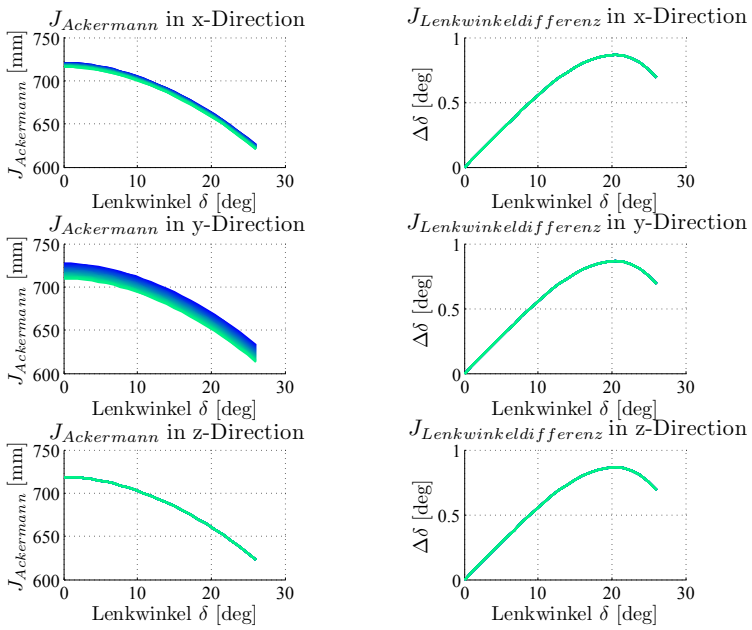


Abbildung A.10.: Sensibilitätsanalyse Gelenk 17