D i p l o m a r b e i t

Entwicklung eines Programms zur automatisierten Simulation und Auswertung von virtuellen Achsanalysen

eingereicht an der

Fakultät Kraftfahrzeugtechnik der

Westsächsischen Hochschule Zwickau

zur Erlangung des akademischen Grades eines

Diplomingenieurs (FH)

vorgelegt von: cand. ing. **M e y e r , M a x** Geb. am: 19.12.1999

Kraftfahrzeugtechnik

Studienschwerpunkt Kraftfahrzeugmechatronik

Erstbetreuer/Zweitbetreuer:	Prof. Dr.-Ing. Uwe Fischer /
	Prof. Dr.-Ing. Jan Schubert
Auftraggeber:	IAMT Engineering GmbH & Co. KG
Betreuer des Auftraggebers:	Dipl.-Math. (techn.) Matthias Seise

[redacted]

Westsächsische Hochschule Zwickau
University of Applied Sciences

Antrag auf Zulassung zum Abschlussprojekt

Diplom ☒ Bachelor ☐ Master ☐

Allgemeine Angaben

Studiengang Kraftfahrzeugtechnik Seminargruppe/Matrikel-Nr. 182235/KFM

Name, Vorname Meyer, Max Anschrift Südstr. 4 07607 Eisenberg

E-Mail-Adresse (nicht FH) Max.Meyer.99@outlook.com

Telefonnummer 015227142918

Zusätzliche Wahlpflicht-Module, die nicht in die Gesamtnote eingehen

---

Angabe noch offener Module (Zulassung mit maximal zwei offenen Modulen gem. jeweils geltender PO)

Weiterführendes Studium an der WHZ geplant Ja ☐ Nein ☒

25.3.22 M. N
Datum Unterschrift Antragsteller/in

Der/Die Antragsteller/in erfüllt die Zulassungsbedingungen der entsprechenden Prüfungsordnung zur Anfertigung der schriftlichen Bachelor/Diplom/Master-Arbeit[1] und hat nachgewiesen, dass er/sie die im Studienplan dafür vorgesehenen Verpflichtungen erfüllt und die festgelegten Prüfungs- und Studienleistungen erbracht hat. [1]... nicht Zutreffendes bitte streichen

25.03.2022 i. A. [redacted]
Datum Name/Unterschrift Prüfungsamt

Thema des Abschlussprojektes inklusive Hauptschwerpunkte

Entwicklung eines Programms zur automatisierten Simulation und Auswertung von virtuellen Achsanalysen

Schwerpunkte:

- Darstellung d. theoretischen Grundlagen zur Achskinematik
- Auswertung typischer kinematischer u. elastokin. Kenngrößen
- Programmierung einer Auswertungsvorlage gem. Anforderungsliste

27.04.22 Prof. Fischer, Uwe
Datum Name/Unterschrift Erstprüfer (Betreuer)

Vorschlag Zweitprüfer: Jan Schubert

27.04.22 M. N
Datum Unterschrift Antragsteller/in

Ausgabetermin 28.04.2022 **Abgabetermin** 18.08.2022

Bestätigung durch den Prüfungsausschuss der Fakultät

28.04.2022
Datum Name/Unterschrift

WESTSÄCHSISCHE HOCHSCHULE ZWICKAU
Fakultät Kraftfahrzeugtechnik
i. A. [redacted]

Postfach 20 10 37
08012 Zwickau

Selbstständigkeitserklärung

Ich versichere, dass ich die Arbeit selbstständig angefertigt und keine anderen als die angegebenen Hilfsmittel benutzt habe. Wörtlich oder sinngemäß aus anderen Quellen übernommene Textstellen, Bilder, Tabellen u. a. sind unter Angabe der Herkunft kenntlich gemacht.

Weiterhin versichere ich, dass diese Arbeit noch keiner anderen Prüfungsbehörde vorgelegt wurde.

Eisenberg, 29.06.2022

Ort, Datum

Max Meyer

Inhalt

I. Verzeichnis der Bilder

II. Verzeichnis der Anlagen

III. Kurzzeichenverzeichnis

Kurzzeichen	Einheit	Bedeutung
ABS	-	Antiblockiersystem
ESP	-	Elektronisches Stabilitätsprogramm
GUI	-	Graphic User Interface
HA	-	Hinterachse
MKS	-	Mehrkörpersimulation
n	mm	Nachlaufstrecke
r_s	mm	Lenkrollradius (scrub)
R_0	mm	Rollzentrum (roll center)
VA	-	Vorderachse
X_{BR}	%	Bremsnickausgleich (anti dive)
γ	°	Sturz (camber)
δ	‘	Spur (toe)
δ_{AM}	°	Ackermannwinkel (ackerman angle)
δ_F	°	Ackermannfehler (ackerman error)
ε_B	°	Bremsabstützwinkel (anti dive angle)
σ	°	Spreizungswinkel (kingpin inclination angle)
τ	°	Nachlaufwinkel (caster angle)

Danksagung

An dieser Stelle möchte ich mich bei all denen bedanken, ohne deren Unterstützung die Erstellung dieser Diplomarbeit in der jetzigen Form nicht möglich gewesen wäre.

Mein Dank gilt an erster Stelle meinen Betreuern bei der IAMT, Herrn Dipl.-Ing Heiko Hennig und Herrn Dipl.-Math. (techn.) Matthias Seise, welche mich während des gesamten Prozesses sehr gut unterstützt und beraten haben.

In gleicher Weise möchte ich mich bei meinen Betreuern an der Westsächsischen Hochschule Zwickau Herrn Prof. Dr.-Ing. Uwe Fischer und Prof. Dr.-Ing. Jan Schubert für Ihre sehr fachliche und kompetente Betreuung bedanken.

Des Weiteren gilt mein Dank allen Kommilitonen des Matrikel Kraftfahrzeugtechnik 182235, die mich während meines Studiums begleitet haben.

Großer Dank gilt auch meinen Eltern und Familie, ohne deren moralische Unterstützung die Bewältigung der Arbeit wesentlich schwieriger gewesen wäre.

1 Einleitung

1.1 Motivation

Bei der virtuellen Entwicklung von mechanischen Baugruppen werden in der Regel mehrere Konzepte und Varianten erstellt. Diese Varianten werden meist sehr aufwendig an vorher ausgewählten Parametern und Bewertungsmatrizen verglichen, damit am Ende die beste Lösung erreicht wird.

Zur richtigen Bewertung dieser Varianten und um aus den berechneten Ergebnissen die korrekten Schlussfolgerungen zu ziehen, ist es wichtig, alle Werte in einer geeigneten Form darzustellen. Viele Berechnungsprogramme bieten interne Auswertungsmöglichkeiten, jedoch meist nur für eine Größe. Um alle Werte vergleichen zu können, müssen die einzelnen Ergebnisse manuell exportiert und extern gemeinsam dokumentiert werden. Dieser Prozess ist nicht nur fehleranfällig, sondern bei der Entwicklung von nicht selten mehr als zehn Varianten auch sehr zeitaufwendig.

Viele große Industrieunternehmen haben sich daher eigene Erweiterungen für das jeweilige Berechnungsprogramm erstellt oder ein eigenes Auswertungstool entwickelt.

1.2 Zielstellung

Ziel der Arbeit ist es, ein Auswertungstool zu entwickeln, welches mithilfe des Mehrkörpersimulationsprogramms Adams virtuelle Achsanalysen auswertet. Dabei sollen die Simulationen anhand von Eingabewerten automatisiert ablaufen und anschließend vordefinierte Fahrwerkskenngrößen mit den Werten aus der Simulation extern berechnet und dargestellt werden.

Zudem soll die Möglichkeit bestehen, bereits vorhandene Werte in Tabellenform einzulesen und mit den simulierten Ergebnissen zu vergleichen.

Ein weiterer Bestandteil der Arbeit ist die Dokumentation der theoretischen Berechnung der vordefinierten Fahrwerkskenngrößen. Diese Größen beziehen sich sowohl auf eine kinematische als auch auf eine elastokinematische Betrachtung.

1.3 Übersicht der Arbeit

Die Arbeit ist in vier Kapitel unterteilt:

- **Kapitel 2** befasst sich mit der theoretischen Berechnung von Fahrwerkskenngrößen, deren Orientierung und Einheit.
- **Kapitel 3** beschreibt den Prozess in Adams und die Umsetzung der Berechnung im Programmskript anhand von Auszügen des Quellcodes.
- **Kapitel 4** fasst die Auswertung der berechneten Ergebnisse zusammen und gibt ein abschließendes Resümee.

2 Theoretische Vorbetrachtung

2.1 Theoretische Berechnung der geforderten Größen

2.1.1 Sturz γ (camber)

Der **Sturz** ist der Winkel zwischen Radmittelebene und einer zur Fahrbahn senkrechten Ebene. [1] S. 25

Bild 1: Rad mit positivem Sturz vgl. [1] S. 25

Der Sturz bewirkt eine mögliche Seitenkraftaufnahme am jeweiligen Rad und hat somit großen Einfluss auf die Querdynamik. Der Sturz ist in der Vorderansicht mathematisch positiv definiert; ein nach außen geneigtes Rad besitzt also einen positiven Sturz, wie in Bild 1 dargestellt. Ein negativer Sturz am kurvenäußeren Rad bewirkt eine sogenannte Sturzseitenkraft, welche zur Fahrzeugmittelebene zeigt und somit die Querführung der jeweiligen Achse verbessert. Um eine verbesserte Seitenkraftübertragung zu erreichen, sollte der Sturz daher negativ sein.

Ein zu großer Sturzwinkel beeinflusst den Reifenverschleiß und den Rollwiderstand negativ. Typische Werte in der Konstruktionslage sind -2° bis 0°.

Im Allgemeinen wird ein Fahrwerk so ausgelegt, dass der Sturz beim Einfedern zunehmend negativ wird, damit eine Beeinträchtigung der Seitenkraft durch positive Sturzwinkel vermieden wird.

2.1.2 Spurwinkel δ (toe)

Der **Spurwinkel** ist der Winkel zwischen den Schnittlinien der Radmittelebene beider Räder mit der Fahrzeugmittelebene ohne Lenkeinschlag. [1] S. 24f.

Bild 2: Achse (Draufsicht) mit Vorspur vgl. [1] S. 24

Wenn die Räder in Fahrtrichtung zueinander zeigen, also der Abstand der Felgenhörner vor den Radmittelpunkten kleiner ist als der Abstand hinten, spricht man von einer Vorspur oder auch positiven Spur wie in Bild 2 dargestellt ist. Ist der Abstand der Felgenhörner vor der Radmittelebene größer als der Abstand dahinter, wird dies als Nachspur oder auch negative Spur bezeichnet.

Typische Werte für die Konstruktionslage:

- Vorderachse bei Hinterradantrieb: 0‘ bis +30’,
- Vorderachse bei Vorderradantrieb: –30’ bis +20’,
- für die Hinterachse max.: –10’ bis +20’.

Die Spur, sowohl positiv als auch negativ, wirkt sich negativ auf den Rollwiderstand und den Reifenverschleiß aus. Um die Fahrstabilität zu gewährleisten, wird eine

geringe statische Vorspur eingestellt, damit die Achse „vorgespannt“ ist. Damit wird erreicht, dass permanent Schlupf am Rad erzeugt wird, wodurch eine eventuelle Seitenkraft sofort übertragen werden kann.

Im Allgemeinen sollte an der Vorderachse beim Bremsen oder in der Kurvenfahrt und somit beim Einfedern eine Nachspur entstehen, um einen stabilen Fahrzustand zu begünstigen. Durch eine negative Vorspur wird ein Untersteuern erzwungen und das Fahrzeug bleibt somit im stabilen Fahrzustand.

2.1.3 Nachlaufwinkel τ (Caster angle) und Nachlaufstrecke n (Caster moment arm)

Der **Nachlaufwinkel** ist der Neigungswinkel der Lenkachse zur Senkrechten auf die Fahrbahn in der x-z-Ebene. Der Nachlaufwinkel ist positiv, wenn die Achse nach hinten geneigt ist. [1] S. 28f.

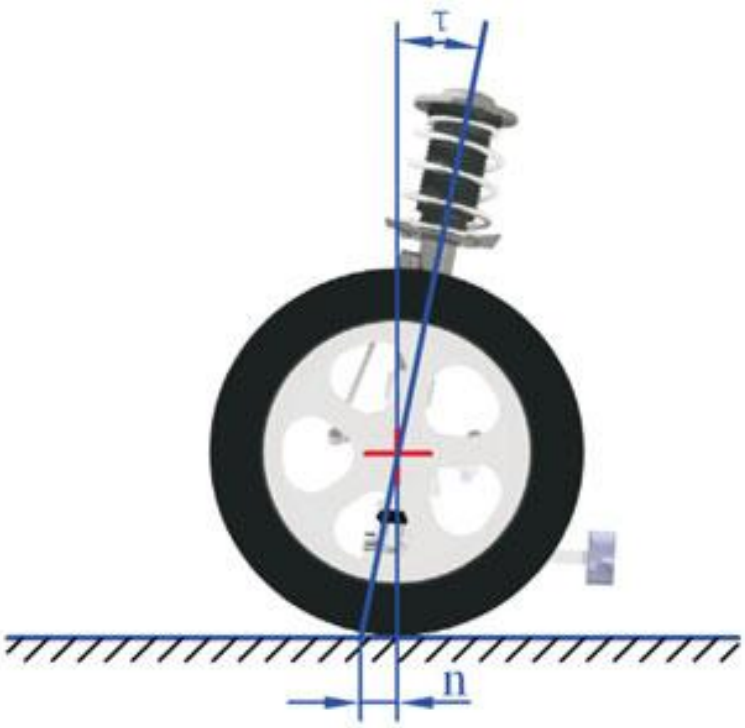


Bild 3: Nachlaufstrecke und Nachlaufwinkel vgl. [1] S. 29

Bild 3 zeigt eine positiven Nachlaufwinkel. (Fahrtrichtung nach links)

Typische Werte in der Konstruktionslage:

- Hinterradantrieb mit Motor vorn: 1° bis 10°,
- Hinterradantrieb mit Motor hinten: 3° bis 15°,
- Vorderradantrieb mit Motor vorn: 1° bis 5°.

Durch den Nachlauf wird der vordere Aufbau beim Lenken angehoben, wodurch ein Lenkrückstellmoment erzeugt wird.

Im Allgemeinen strebt man beim Einfedern eine geringe Änderung des Nachlaufwinkels an, da sich damit auch die Nachlaufstrecke ändert.

Die **Nachlaufstrecke** ist der Abstand auf der Fahrbahn zwischen dem Durchstoßpunkt der Lenkachse und einer Senkrechten durch den Radaufstandspunkt gemessen in Radmittelebene. [1] S. 28ff.

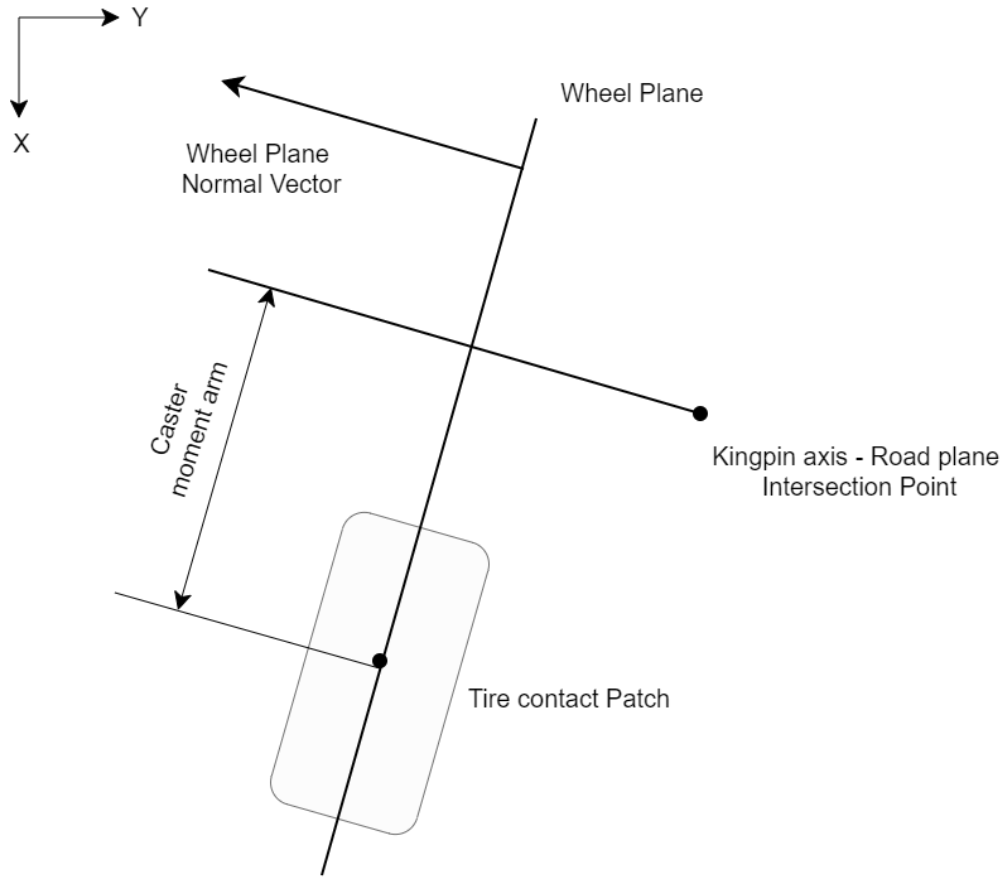


Bild 4: Nachlaufstrecke (Caster moment arm)

Der Nachlauf wird positiv definiert, wenn der Durchstoßpunkt vor dem Radaufstandspunkt liegt. In Bild 4 wird eine positive Nachlaufstrecke (Caster moment arm) dargestellt.

Typische Werte in der Konstruktionslage:

- bei mechanischer Lenkung: 0 mm bis 10 mm,
- bei Servolenkung: 10 mm bis 40 mm.

Die Nachlaufstrecke hat einen großen Einfluss auf die Spurhaltungsstabilität und die Lenkrückstellkraft, da bei einem positiven Nachlauf das Rad hinter der Lenkachse und damit in der Spur bleibt. Besonders beim Bremsen wird die Stabilität dadurch positiv beeinflusst.

2.1.4 Spreizungswinkel σ (Kingpin inclination angle)

Der **Spreizungswinkel** ist der Winkel der Schrägstellung der Lenkachse zu einer Senkrechten zur Fahrbahn in y-z-Ebene. [1] S. 27f.

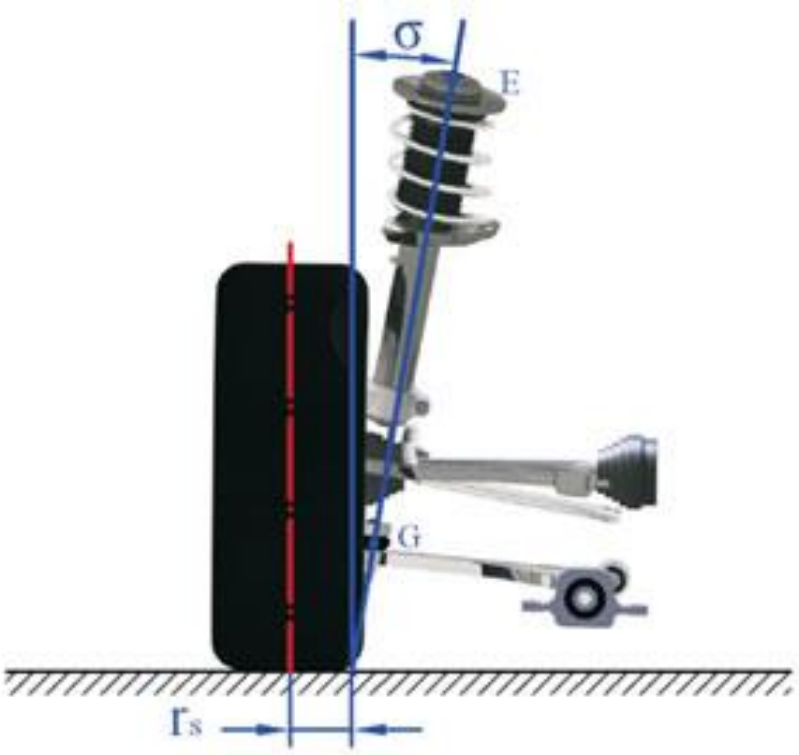


Bild 5: Spreizungswinkel (McPherson) vgl. [1] S. 27

Der Spreizungswinkel ist positiv, wenn die Achse in der Vorderansicht mathematisch negativ gedreht wurde, also nach innen zeigt, wie in Bild 5 dargestellt.

Typische Werte in der Konstruktionslage:

- Hinterradantrieb mit Motor vorn: 5° bis 9°
- Hinterradantrieb mit Motor hinten: 5° bis 13°
- Vorderradantrieb mit Motor vorn: 8° bis 16°

Die Position der Lenkachse sollte nah an der Radmittelebene liegen, um die Hebelarme der am Rad angreifenden Kräfte möglichst klein zu halten. Deswegen wird der untere Gelenkpunkt so weit wie möglich in die Felgenschüssel gelegt. Je nach Aufhängungstyp und verfügbarem Bauraum ist das beim oberen Aufhängungspunkt nur

eingeschränkt möglich, wodurch sich ein erzwungener Spreizungswinkel ergibt. Die Spreizung hat Einfluss auf den Lenkrollradius, welcher unter anderem die beim Lenken auftretenden Rückstellkräfte definiert. Außerdem beeinflusst der Spreizungswinkel die Nachlaufänderung.

Allgemein sollte sich die Spreizung über den Radhub nur wenig ändern, um bei einer ungleichmäßigen Einfederung unerwünschte Lenkmomente, welche aus den Störkrafthebelarmen resultieren, zu verhindern.

2.1.5 Lenkrollradius r_S (scrub radius)

Der **Lenkrollradius** ist der Abstand zwischen Durchstoßpunkt der Lenkachse mit der Fahrbahnebene und der Schnittlinie der Radmittelebene in x-y-Ebene. [1] S. 28

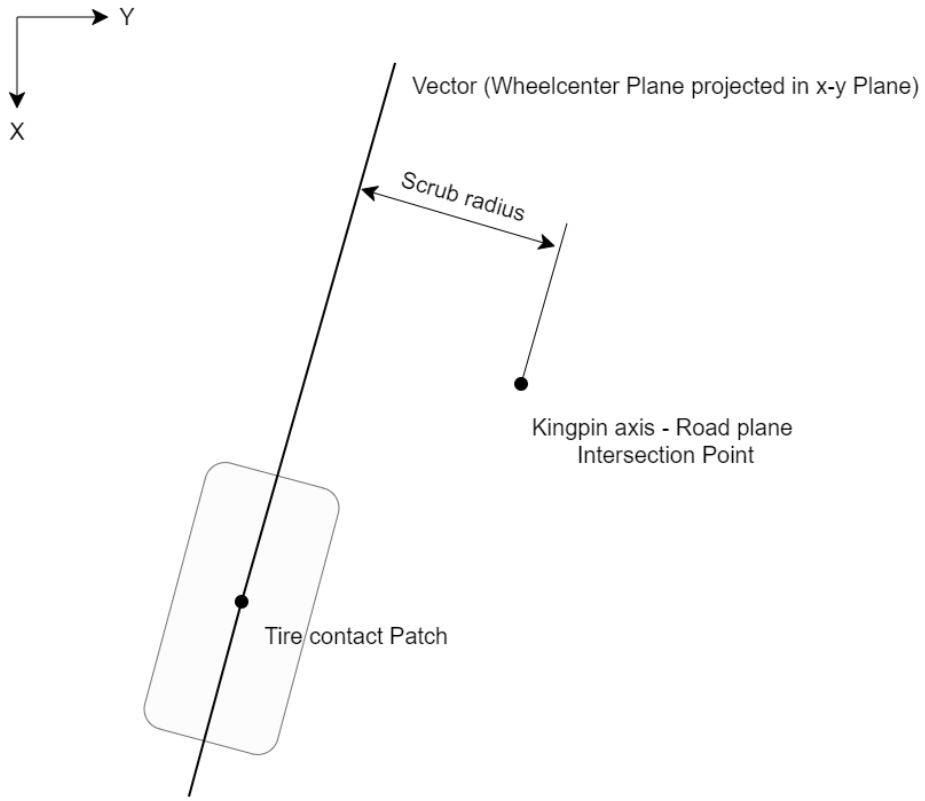


Bild 6: Lenkrollradius (scrub radius)

Der Lenkrollradius ist positiv, wenn der Lenkachsenschnittpunkt in Richtung Fahrzeugmitte von der Radmittelebene ausgehend liegt, wie auch in Bild 6 abgebildet ist. Er ist negativ, wenn er von der Radmittelebene aus nach außen liegt.

Typische Werte in der Konstruktionslage betragen -20 mm bis 80 mm.

Vor der Einführung von elektrischen Regelsystemen (ABS, ESP) in Kraftfahrzeugen wurde der Lenkrollradius bei Fahrzeugen mit Frontantrieb tendenziell negativ ausgelegt, um geringe Antriebseinflüsse in der Lenkung zu spüren. Heute wird der Lenkrollradius generell nahe 0 mm ausgelegt, um den Einfluss der individuellen Radregelung durch Regelsysteme auf das Lenkmoment zu minimieren. Außerdem hat ein Störkrafthebelarm, welcher entsteht, sobald der Lenkrollradius nicht Null ist, negative Einflüsse auf wichtige Fahrversuche, wie zum Beispiel das µ-Splitt-Bremsen. Bei diesem

Versuch werden mit dem Fahrzeug unterschiedliche Oberflächen befahren, zum Beispiel eine Straße, die zur Hälfte vereist ist. Wegen der unterschiedlichen Reibwerte (μ) der Oberfläche, kann es zum Ausbrechen des Hecks kommen.

Vor der Einführung der elektrischen Regelsysteme wurde der Lenkrollradius bewusst so ausgelegt (≠ 0), dass das Fahrzeug beim μ-Splitt-Bremsen auf die Seite mit dem höheren Reibwert lenkt. Diese Auslegung sollte dem Fahrer helfen, das Fahrzeug auf die Fahrbahnseite mit dem höheren Reibwert zu manövrieren, um es sicher zum Stillstand zu bringen.

2.1.6 Rollzentrum R_0 (roll center) und Rollzentrumshöhe (roll center height)

Das **Rollzentrum** ist der Punkt, um den sich die Karosserie gegenüber der Achse beim Wanken dreht.

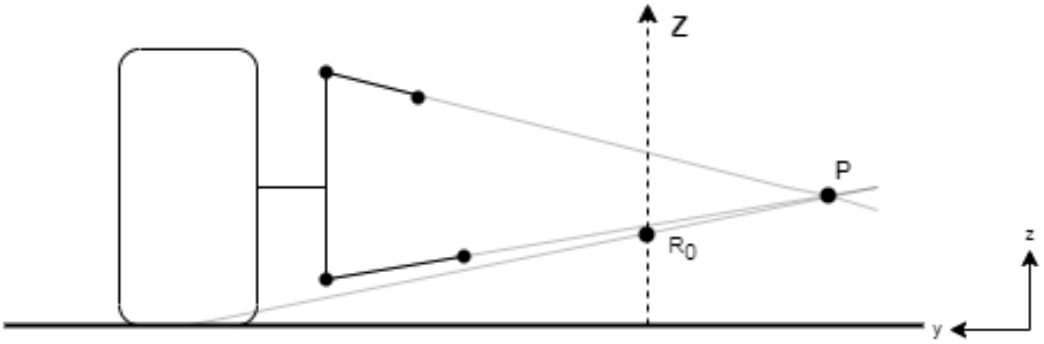


Bild 7: Geometrische Bestimmung des Rollzentrums

Das Rollzentrum ergibt sich aus dem Schnittpunkt der Geraden, welche zwischen Radaufstandspunkt und Querpol P verläuft, mit der Fahrzeugmittelebene. Der Querpol ist der Schnittpunkt aus den Verlängerungen der beiden Querlenker in y-z-Ebene.

Die **Rollzentrumshöhe** ist der Abstand zwischen Rollzentrum und der Fahrbahnebene, wie in Bild 7 dargestellt. Die Rollzentrumshöhe ist positiv, wenn das Rollzentrum über der Fahrbahnebene liegt.

Typische Werte in der Konstruktionslage:

- an der Hinterachse: 80 mm bis 250 mm (höher, wegen eventueller Zuladung)
- an der Vorderachse: 0 mm bis 130 mm

Die Rollzentrumshöhe hat Einfluss auf das Wankverhalten des Fahrzeuges. Bei positiven Werten wankt der Aufbau weniger, da der Hebelarm zum Fahrzeugschwerpunkt klein ist. Eine negative Rollzentrumshöhe bewirkt ein stärkeres Wanken (vgl. [1] S. 25f.).

Im Allgemeinen wird eine positive Rollzentrumshöhe angestrebt, um möglichst kleine Wankwinkel zu erreichen.

2.1.7 Bremsnickausgleich X_{BR} (anti dive) / Bremsabstützwinkel ε_B (anti dive angle)

Der **Bremsnickausgleich** ist der Anteil der Abstützung der beim Bremsen entstehenden Nickmomente durch die Achskinematik der Radaufhängung. [1] S. 71f.

Ein hoher Bremsnickausgleich bedeutet, dass der Fahrzeugaufbau beim Bremsen weniger nickt.

Typische Werte in der Konstruktionslage sind 60% bis 80%.

Analog dazu existiert der Anfahrnickausgleich, bei welchem prinzipiell die gleiche Betrachtung wie beim Bremsnickausgleich gilt. Jedoch wird das Antriebsmoment in der Regel über eine Antriebswelle zum Rad übertragen und stützt sich somit direkt an der Karosserie ab.

Der **Bremsabstützwinkel** ist der Schnittwinkel zwischen der Geraden vom Radaufstandspunkt RP zum Längspol der jeweiligen Achse und der x-Achse. [1] S. 72

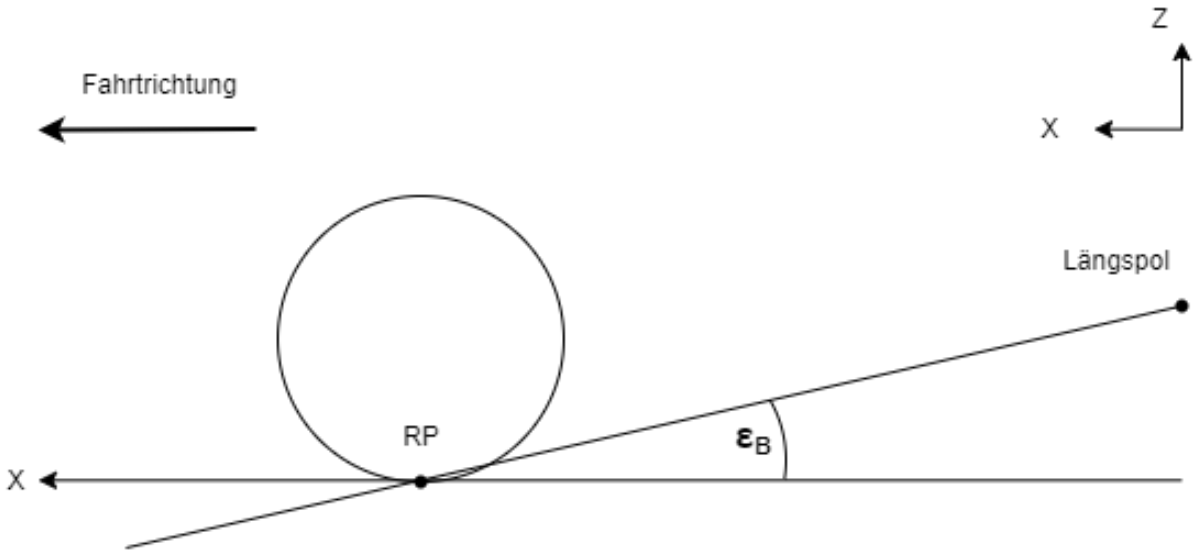


Bild 8: Konstruktion Bremsabstützwinkel

Um den optimalen Bremsabstützwinkel am Gesamtfahrzeug zu bestimmen, wird eine Kräftebilanz um den Radmittelpunkt formuliert. Die Resultierende aus Bremskraft und Radlastdifferenz greift im Radaufstandspunkt (Latsch) an. Die Wirkungslinie der Resultierenden bildet zusammen mit der x-Achse den optimalen Bremsabstützwinkel, wie

in Bild 8 dargestellt. Liegt der tatsächliche Nickpol einer Achse außerhalb dieser Wirkungslinie und es kommt zu dynamischen Radlaständerungen, so verursacht die Resultierende im Reifenlatsch ein Moment um den Fahrzeugschwerpunkt, welches über eine Federkraftänderung (z.B. nicht-lineare Feder, generell höhere Federkonstante) kompensiert werden muss.

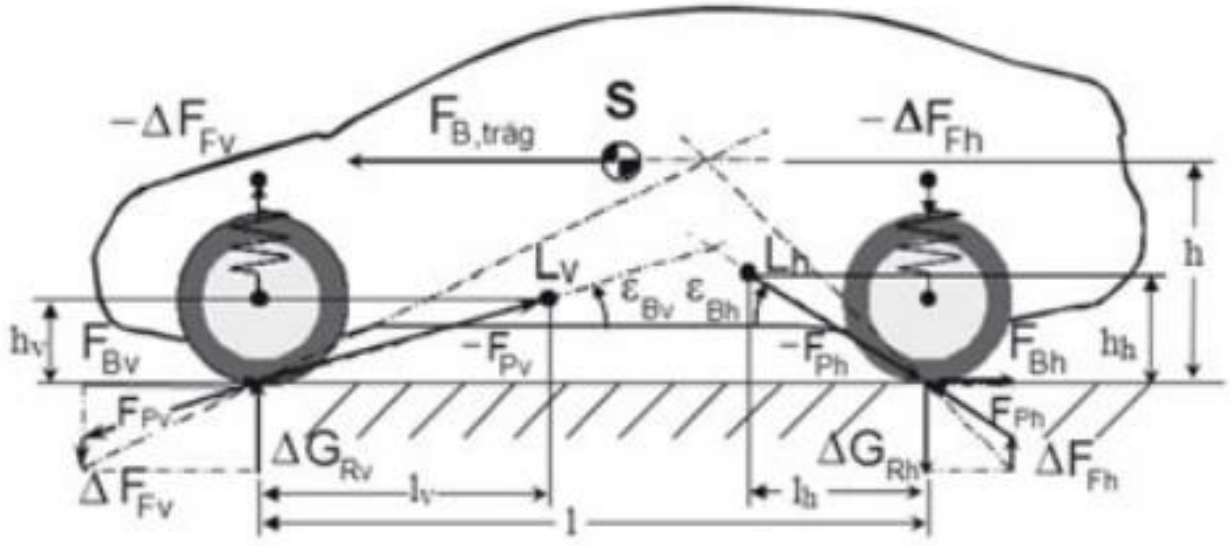


Bild 9: Kräfte und Längen am Fahrzeug beim Bremsen vgl. [1] S. 71

Der optimale Bremsabstützwinkel wird geometrisch wie in Bild 8, sowie über ein Momentengleichgewicht, welches aus Bild 9 resultiert, beschrieben:

$$\tan(\varepsilon_{opt,v}) = \frac{h}{l} \cdot \left(1 + \frac{1}{F_{Bv}/F_{Bh}}\right) \tag{2.01}$$

$$\tan(\varepsilon_{opt,h}) = \frac{h}{l} \cdot \left(1 + F_{Bv}/F_{Bh}\right) \tag{2.02}$$

Eine Momentenbilanz um den tatsächlichen Längspol L einer Achse (Vorderachse), der sich aus der Konstruktion und der kinematischen Lage ergibt, zeigt den Grad des Bremsnickausgleichs X an.

$$X_V = \frac{F_{Bv}}{\Delta G_V} \cdot \frac{h_V}{l_V} = \frac{\tan(\varepsilon_{tats.})}{\tan(\varepsilon_{opt.})} \cdot 100\% \qquad (2.03)$$

Analog erfolgt die Bestimmung für die Hinterachse. Für den Bremsnickausgleich werden also der tatsächliche sowie der optimale Bremsabstützwinkel in Relation gesetzt. Der optimale Bremsabstützwinkel ist dabei eine Größe, welche über die Fahrzeugparameter bestimmt wird, während der tatsächliche Bremsabstützwinkel einen kinematischen Kennwert einer Achse darstellt.

2.1.8 Ackermannwinkel δ_{AM} (ackerman angle) / Ackermannfehler δ_F (ackerman error)

Der **Ackermannwinkel** ist der Vorderradlenkwinkel, um ein frontgelenktes Fahrzeug ohne Seitenkraft und damit ohne Schräglaufwinkel, also bei sehr langsamer Kurvenfahrt, um eine Kurve zu führen. [1] S. 30f.

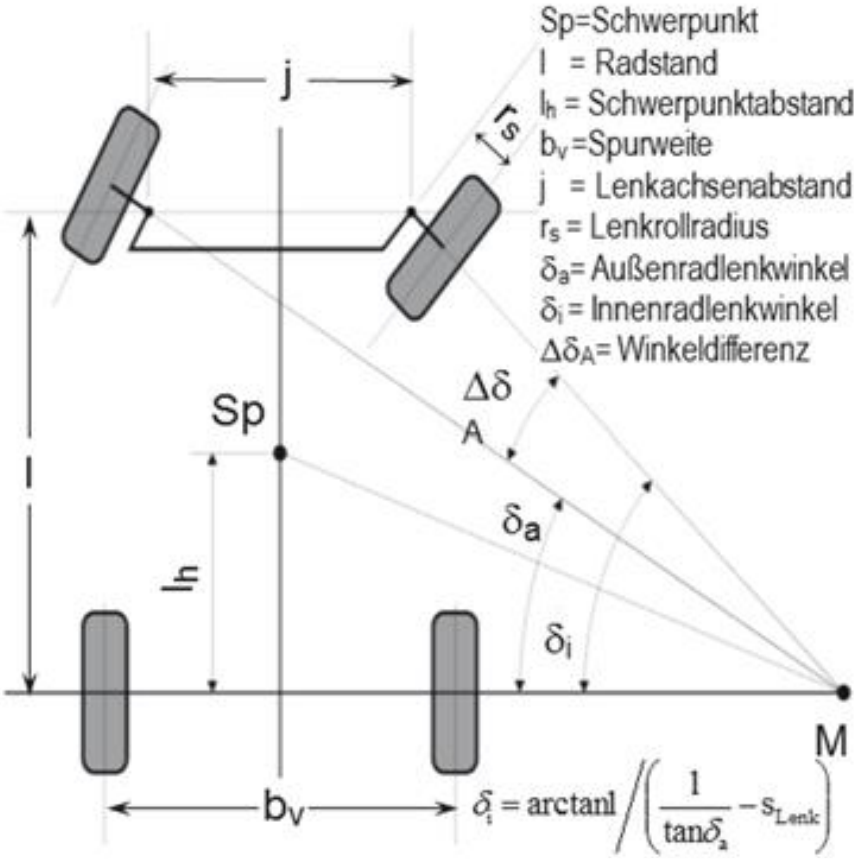


Bild 10: Geometrische Bestimmung Ackermann vgl. [1] S. 30

Da beide Räder einen unterschiedlich großen Kreisbogen abhängig von der Spurweite abfahren müssen, weisen diese auch unterschiedliche Radlenkwinkel auf. Der Lenkwinkel am kurveninneren Rad muss dabei stets größer sein als jener am kurvenäußeren Rad.

Generell werden Lenkungen nie nach dem Ackermann-Gesetz (Ackermann-Gesetz beschreibt die Berechnung für den idealen Differenzwinkel, sodass beide Räder den optimalen Lenkwinkel besitzen) ausgelegt, da dadurch unter anderem der Wendekreis nicht am kleinsten ist. Ohne Abweichung vom idealen Lenkwinkel würde am Rad kein

Reifenschräglauf entstehen und somit keine Seitenkraft übertragen werden. Außerdem müsste der Bauraum des Fußraumes in der Fahrgastzelle zur Seite stark begrenzt werden, da sehr große Einschlagwinkel am kurveninneren Rad entstehen würden.

Deshalb wird bewusst ein **Ackermannfehler** von 10% oder bis zu 3° angestrebt. Der Ackermannfehler beschreibt dabei die Abweichung des realen Radlenkwinkels zum optimalen Radlenkwinkel.

2.1.9 Feder- und Dämpferübersetzung

Die **Feder- bzw. Dämpferübersetzung** ist das Verhältnis zwischen Radhubänderung (Δz_R) und tatsächlicher Längenänderung (Δf) der Feder bzw. des Dämpfers.

$$i = \frac{\Delta f}{\Delta z_R} \tag{2.04}$$

Das Übersetzungsverhältnis i ist in der Regel kleiner als 1 und muss nicht konstant sein, da es von der momentanen Lage der Getriebeglieder abhängig ist (vgl. [1] S. 74). In der Literatur ist diese Definition nicht einheitlich, weswegen es unter Verwendung des Reziprok auch zu Verhältnissen größer als 1 kommen kann. Bei einer heutzutage üblichen Einzelradaufhängung hängt die Übersetzungsänderung nur von der Kinematik der Achse ab.

Bei einer Feder-Dämpfer-Einheit (Federbein) ist die Übersetzung gleich.

2.2 Übersicht der verwendeten Software

Adams - **A**utomatic **D**ynamic **A**nalysis of **M**echanical **S**ystems [2] - ist ein Mehrkörpersimulationsprogamm (MKS-Programm) und eine häufig eingesetzte Software für die Simulation mechanischer Systeme. Es analysiert das Bewegungsverhalten von dreidimensionalen Systemen unter Berücksichtigung aller physischen Interaktionen.

Analysen können statisch, dynamisch, im Zeit- und im Frequenzbereich durchgeführt werden. Deswegen wird Adams in vielen Industriebereichen eingesetzt.

In der Automobilindustrie haben sich Adams und das Modul Adams/Car als Standards durchgesetzt. Dabei werden unter anderem Teilsysteme aus dem Fahrwerks- und Antriebsbereich simuliert sowie vollständige Fahrzeugsysteme.

Adams Car wird in Release Version 2020 (Build: 2020.0.0-CL711253 on Feb 25 2020) verwendet.

Python ist eine objektorientierte Programmiersprache und erschien erstmals 1994 [3]. Laut dem PYPL-Index im Mai 2022 [4] hat Python einen Marktanteil von 27,85% und ist somit weltweit die mit Abstand größte und meistverwendete Programmiersprache.

Python bietet gegenüber anderen Programmiersprachen eine Vielzahl an Vorteilen [5]:

- einfache Syntax
- umfangreiche Standardbibliotheken
- kurzer und einfach zu lesender Code
- gute Erweiterbarkeit dank einer großen Sammlung von Python-Add-on-Paketen
- für alle gängigen Betriebssysteme nutzbar
- frei verfügbar

Python wird in Version 3.10 verwendet.

3 Umsetzung in Adams und Python

3.1 Anforderungsliste

Die Anforderungsliste befindet sich im Anhang 1 und ist in vier Punkte aufgeteilt:

- Die **allgemeinen Anforderungen** beziehen sich auf die zu leistenden Funktionen des Programms, wie zum Beispiel die Anzahl der zu vergleichenden Achskonfigurationen oder die Sprache.
- Die **Kinematik** (Lagerstellen werden als ideal steif betrachtet) listet alle auszuwertenden Größen auf, die jeweilige Einheit und deren Form, entweder tabellarisch in Konstruktionslage oder als graphischer Verlauf.
    - Der graphische Verlauf wird über den Radhub dargestellt (Y-Achse).
- Die **Lenkkinematik** listet alle auszuwertenden Größen auf, die jeweilige Einheit und deren Form, entweder tabellarisch in Konstruktionslage oder als graphischer Verlauf mit Ausnahme vom Spurkreis und Ackermannfehler (Ackermannfehler wird immer maximal angegeben, Spurkreis immer minimal).
    - Der graphische Verlauf wird über den gemittelten Radlenkwinkel dargestellt (X-Achse).
- Die **Elastokinematik** (Lagerelastizitäten werden berücksichtigt) listet alle auszuwertenden Größen auf, die jeweilige Einheit und deren Form, entweder tabellarisch in Konstruktionslage oder als graphischer Verlauf.
    - Der graphische Verlauf wird bei den statischen Lastfällen über die jeweilige Kraft angegeben (Y-Achse) und sonst über den Radhub (Y-Achse).
    - Die Radlast mit/ohne Stabilisator soll in einem Diagramm dargestellt werden.

3.2 Modelle und Simulationen in Adams

Um zu gewährleisten, dass die Achsanalyse universell einsetzbar ist, müssen Grundannahmen in Adams getroffen werden.

Das globale Achskoordinatensystem wird folgendermaßen definiert: Der Koordinatenursprung (in Adams origo) liegt in der Straßenebene und mittig zwischen den beiden Radzentren. Ausgehend von diesem Punkt zeigt X entgegen der Fahrtrichtung, Y zum rechten Rad und Z nach oben. Bild 11 veranschaulicht sowohl die Lage als auch die Orientierung des Koordinatensystems.

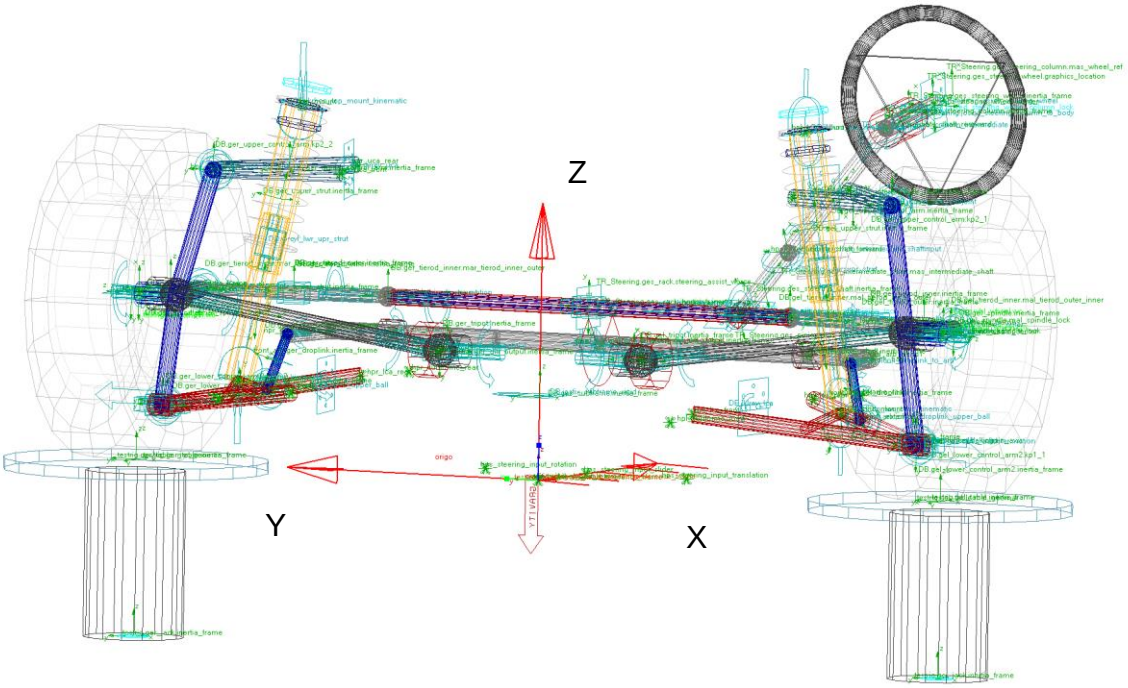


Bild 11: Orientierung des globalen Koordinatensystems (isometrische Frontansicht)

Das Assembly (Modell der Baugruppe) ist aus mehreren Subsystemen (meist kleinere Baugruppen) aufgebaut und sollte aus mindestens der Achse selbst, einem Lenksystem und einem Stabilisator bestehen. Das Assembly wird in Adams mithilfe eines Testrigs (Prüfstand) angesteuert (Radwegsteuerung, Lenkwinkelsteuerung etc.), welcher wichtige Achs- bzw. Fahrzeugparameter (z.B. angenommener Radstand des Gesamtfahrzeugs) enthält. Außerdem muss das Modell bestimmte Marker besitzen. Eine genaue Beschreibung folgt in den Unterpunkten 3.2.2 und 3.2.3.

Das Modell muss sich beim Öffnen in einem ideal steifen Zustand befinden, d.h. alle Subsysteme sind im kinematischen Modus und der Subframe (Befestigungspunkte der Achse an die Karosserie) ist deaktiviert. Andernfalls kommt es zu unerwünschten Elastizitäten, welche das Ergebnis teilweise sehr stark beeinflussen können.

Am Ende jeder Simulation werden fünf unterschiedliche Dateien gespeichert vom Typ .acf, .adm, .wev, .msg und .res. In der Ergebnisdatei (Resultfile mit Dateiendung .res) werden die Ergebnisse aus der Simulation gespeichert.

3.2.1 Erstellung des Befehlsskripts

Um in Adams Simulationen automatisiert ablaufen zu lassen, muss ein sogenanntes Befehlsskript erstellt werden. Dieses wird automatisch beim Start des Programmes ausgeführt, wenn eine Datei Namens „acar.cmd“ im vorher ausgewählten Arbeitsverzeichnis liegt. In diese Datei werden alle benötigten Befehle in Adams-Syntax eingefügt und abgespeichert. Das Skript enthält unter anderem die Informationen, welches Assembly betrachtet wird, wie viele Steps (Berechnungsschritte) die Simulation besitzt, welche Versuche mit den jeweiligen Parametern durchgeführt werden und welche zusätzlichen Requests (Abfragen) gestellt werden.

Das Befehlsskript ist für die geforderte Achsanalyse folgendermaßen aufgebaut:

- Öffnen des Assemblys
- Definieren der Requests (z.B. Bewegung eines Punktes während der Simulation)
- Starten der kinematischen Simulationen:
    - Erster Versuch ist der *opposite Wheeltravel,* bei welchem die Räder gleichzeitig gegeneinander bewegt werden (z.B. linkes Rad -80 mm Radhub, rechtes Rad +80mm Radhub).
    - Zweiter Versuch ist das sog. *Steering*, bei welchem die Achse anhand eines definierten Lenkradwinkels von rechts nach links durchgelenkt wird.
- Umschalten des kinematischen Modus in den elastokinematischen Modus

- Starten der elastokinematischen Simulationen:
    - Dritter Versuch ist der *parallel Wheeltravel,* bei welchem die Räder gleichzeitig miteinander bewegt werden (z.B. von -80 mm Radhub nach +80 mm Radhub).
    - Vierter Versuch ist der *opposite Wheeltravel.*
    - Fünfter und sechster Versuch sind die *statischen Lastfälle,* bei welchen jeweils eine Längs- bzw. Querkraft im Radaufstandspunkt eingeleitet wird.
- Schließen des Assemblys

Bei mehreren Modellen wird dieser Vorgang für jedes Modell wiederholt.

3.2.2 Berechnungsmodelle

Beispielhaft werden für diese Arbeit zwei Modelle gewählt, an denen die Funktionsweise des zu schreibenden Programms dargestellt werden soll.

Das erste Modell ist eine Doppelquerlenkerachse. Ein solcher Achstyp zeichnet sich dadurch aus, dass er zwei Dreieckslenker (dargestellt in rot und hellblau), besitzt welche übereinander angeordnet sind wie in Bild 12 abgebildet. Somit ist eine geometrische Bestimmung der Lenkachse möglich. Diese Bestimmung erfolgt durch die beiden Punkte, an denen das Schwenklager mit dem jeweiligen Dreieckslenker verbunden ist.

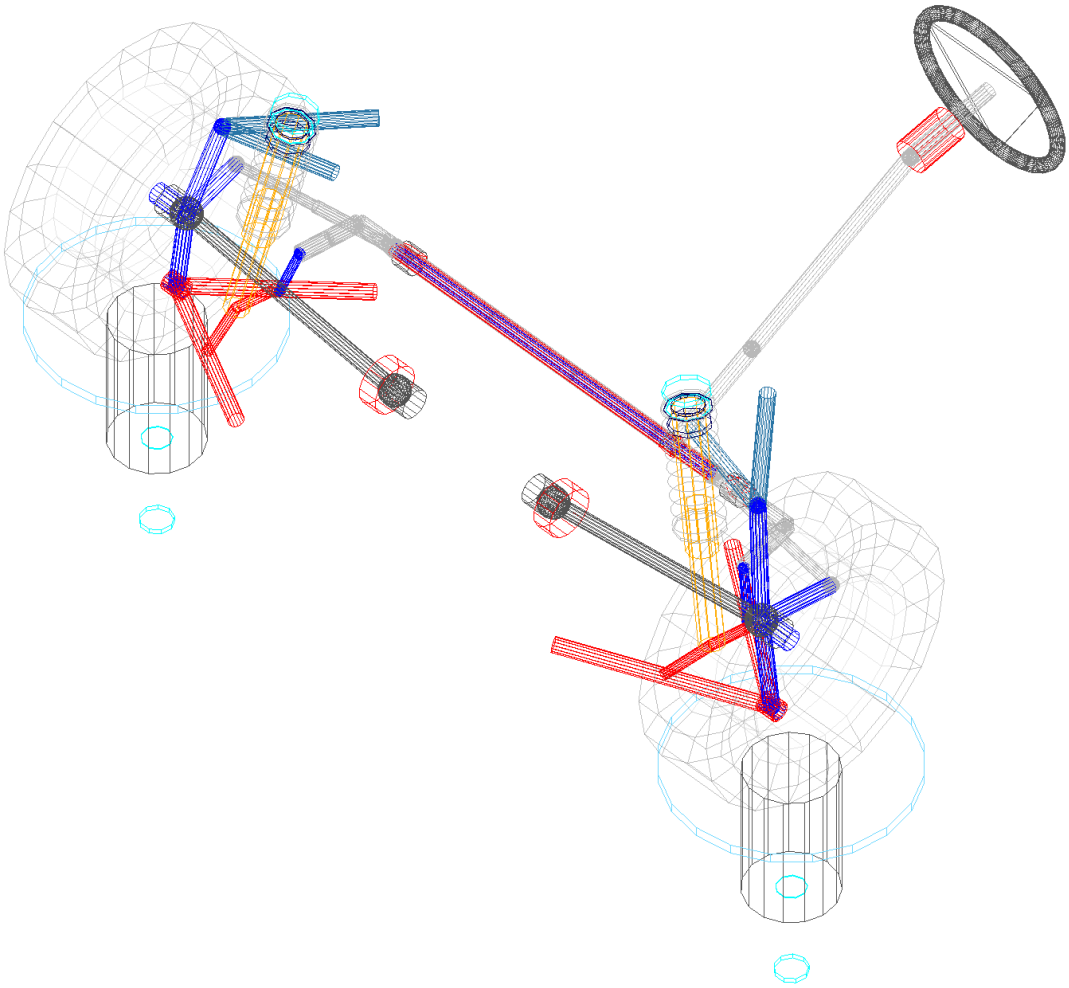


Bild 12: Modell 1 - Doppelquerlenkerachse

Das zweite Modell ist ebenfalls eine Doppelquerlenkerachse, allerdings ist der untere Dreieckslenker (dargestellt in grün) aufgelöst, wie in Bild 13 abgebildet. Das hat zur Folge, dass die Lenkachse nicht mehr geometrisch bestimmbar ist, sondern nur noch virtuell existiert. Der untere Punkt der Lenkachse muss für jeden Berechnungsschritt neu berechnet werden. Diese Berechnung wird im Unterpunkt 3.3.4 erläutert.

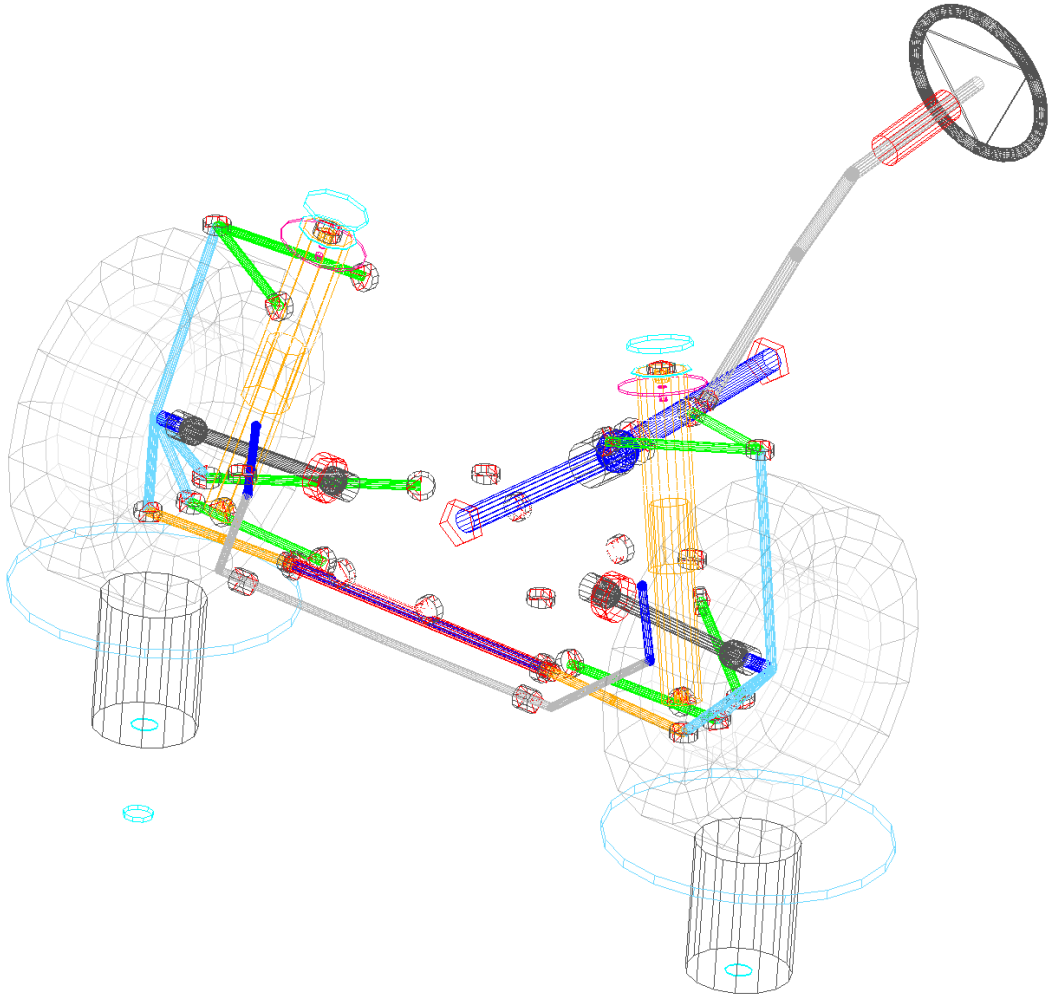


Bild 13: Modell 2 – Doppelquerlenkerachse mit aufgelöstem Dreieckslenker unten

Die Berechnung der Werte erfolgt unter anderem durch die Bewegung bestimmter Punkte an der Achse. Um diese Bewegung aufzuzeichnen, muss in Adams, wie bereits erwähnt, ein Request erstellt werden, welches den Abstand zu origo und damit die Bewegung eines vorher definierten Punktes am Modell (Marker) zurückgibt. Diese Marker werden zuvor in Adams definiert.

3.2.3 Bedeutung der ausgewählten Versuche

In der Achsanalyse werden pro Modell insgesamt sechs Versuche simuliert. Davon zwei im kinematischen und vier im elastokinematischen Modus.

Beim kinematischen *opposite Wheeltravel* wird die Achse nach den vorgegebenen Eingabewerten durchgefedert. Das rechte Rad ist am Anfang maximal eingefedert und das linke Rad maximal ausgefedert. Die Räder bewegen sich während der Simulation gegeneinander (opposite) in die jeweils andere Position. In diesem Versuch werden folgende Größen in Abhängigkeit vom Radhub ausgewertet: Sturz, Spur, Nachlaufstrecke und -winkel, Feder- und Dämpferübersetzung, Rollzentrumshöhe und der Bremsabstützwinkel. Alle auszuwertenden Größen könnten wahlweise auch mit einem *parallel Wheeltravel* bestimmt werden, da immer nur ein Rad betrachtet wird.

Der zweite kinematische Versuch ist das *Steering*. Dabei wird die Achse nach den vorher festgelegten Grenzen durchgelenkt. Der Versuch beginnt mit maximal eingeschlagenen Rädern nach rechts und endet mit maximal eingeschlagenen Rädern nach links. (Lenkeinschlag ist abhängig von der Lenkübersetzung und den Eingabewerten) In diesem Versuch werden folgende Größen in Abhängigkeit vom gemittelten Lenkwinkel am Rad ausgewertet: Ackermannwinkel, Ackermannfehler, Nachlaufstrecke und -winkel, Lenkrollradius, Spreizungswinkel, Spurkreisradius und die Lenkübersetzung.

Die elastokinematischen Versuche lassen sich prinzipiell zu zwei Kategorien zusammenfassen. Sowohl der *parallel* als auch *opposite Wheeltravel* dienen zur Bestimmung der Radlasten, der Vertikalsteifigkeit, der Spur und des Sturzes. Um die Radlast mit und ohne Stabilisator zu bestimmen, wird jeweils der andere Versuch verwendet. Beim *parallel Wheeltravel* hat der Stabilisator keinen Einfluss auf die Radlast über den Radhub, da beide Räder gleichzeitig in die gleiche Richtung eingefedert werden. Im Gegensatz dazu verteilt beim *opposite Wheeltravel* der Stabilisator die Radlast zwischen den Rädern und daher ändert sich die Radlast über dem Radhub.

Die beiden statischen Lastfälle (*static Load)* leiten jeweils eine laterale (Quer-) und eine longitudinale (Längs-) Kraft ein. Der Wert und die Richtung der Kräfte werden ebenfalls über eine Eingabe bestimmt. Die Größen, welche in diesen Versuchen bestimmt werden, sind Spur und Sturz über den Kraftverlauf, sowie die Längs- und Quersteifigkeit.

3.3 Programmierung mittels Python

3.3.1 Dateistruktur und Aufbau des Programms

Das Programm besteht aus insgesamt neun Skripten und ist in zwei Unterprogramme unterteilt.

Unterprogramm 1

Input Values

Der erste Teil ist ein GUI (Graphic User Interface), in dem sowohl die Assemblys mit den dazugehörigen Daten als auch die Versuchsgrößen der Versuche eingetragen werden. Das Hauptfenster ist in Bild 14 zu sehen.

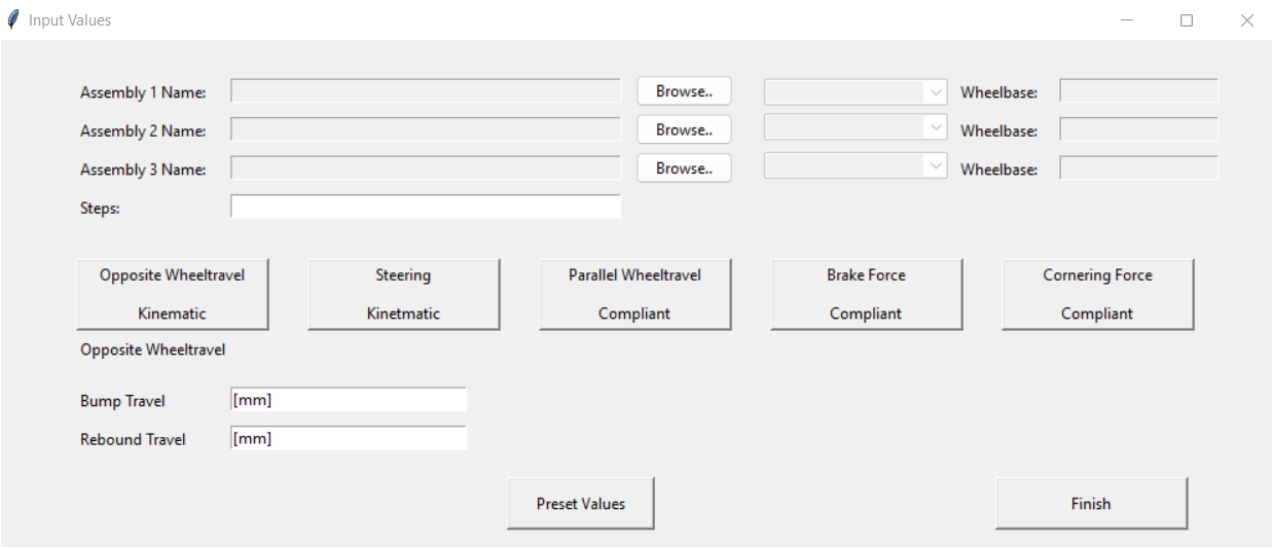


Bild 14: GUI

Sobald ein Assembly ausgewählt wird, aktiviert sich das nebenstehende Dropdown-menü, zur Auswahl des Achstyps. Abhängig von dieser Auswahl, öffnet sich ein weiteres Fenster im Vordergrund, wie auf Bild 15 und 16 dargestellt wird, um die Marker und Hardpoints (charakteristische Punkte einer Achse) einzutragen. Nach erfolgreicher Eingabe und Bestätigung mit der *Save-Schaltfläche* aktiviert sich das Eingabefeld für die Wheelbase (Radstand), welche ebenfalls zur Berechnung benötigt wird.

Falls eine Tabelle eingelesen werden soll, wird die Datei wie ein Assembly ausgewählt und im Dropdownmenü der Punkt Table ausgewählt. In diesem Fall erscheint kein zusätzliches Eingabefenster und die Eingabe für den Radstand bleibt deaktiviert.

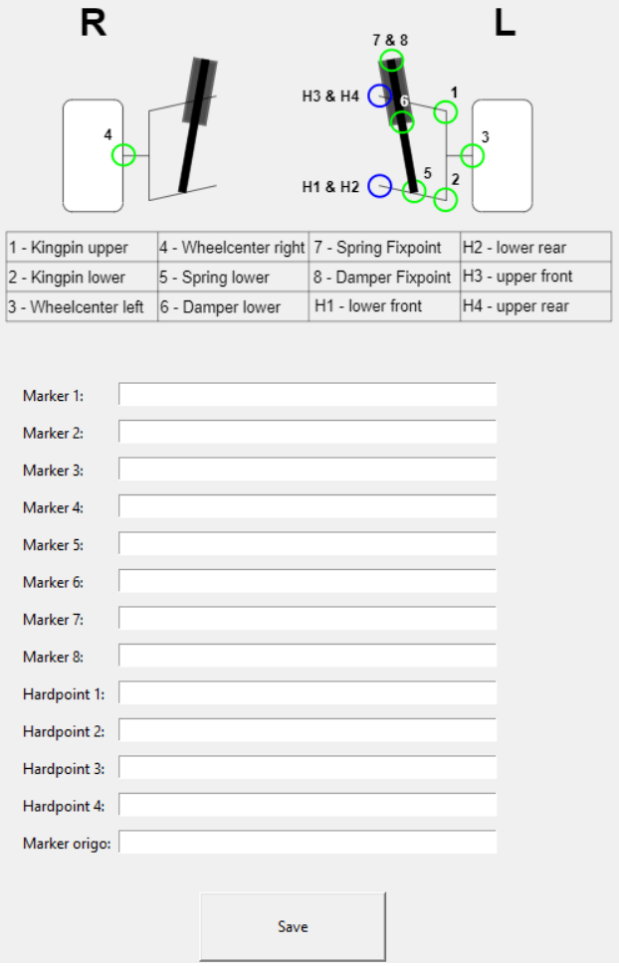


Bild 15: Doppelquerlenker GUI

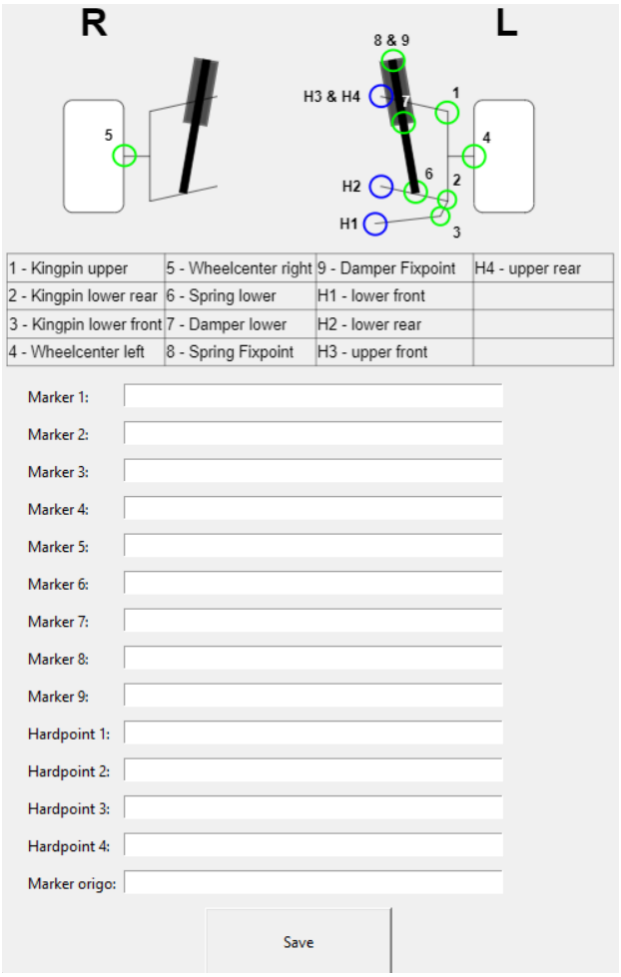


Bild 16: Multi Link GUI

Im unteren Teil des Hauptfensters auf Bild 14 befinden sich die Eingabefelder für die Versuchsgrößen. Diese können einzeln eingegeben oder über die *Schaltfläche Preset* Values mit allgemeinen vordefinierten Werten beschrieben werden.

Nach erfolgter Eingabe der Werte und Betätigung der *Schaltfläche Finish* wird das Hauptfenster geschlossen und die Skripte acar_File und database werden ausgeführt.

acar_File

Das Skript acar_File erzeugt mithilfe der Eingabewerte die Datei acar.cmd mit dem darin befindlichen Befehlsskript, um die Adams Simulation zu automatisieren. Die Syntax für das Befehlsskript existiert bereits und wird durch die Marker ergänzt.

database

Das Skript database wird erstmals im Skript Input Values aufgerufen, nachdem die *Schaltfläche Finish*, welche im Bild 14 dargestellt ist, betätigt wurde. Damit wird eine Datenbank erstellt, in welcher alle Eingabedaten an den zweiten Teil des Programms übergeben werden. Anschließend werden diese Werte in der Datei pythonsqlite.db im Arbeitsverzeichnis des Programms abgespeichert.

Unterprogramm 2

Main

Im Skript Main sind die Hauptfunktionen des zweiten Teils verankert. Es dient hauptsächlich zur Steuerung und beinhaltet den Suchalgorithmus für die von Adams erzeugten Ergebnisdateien (Resultfiles).

config

Der Inhalt des config-Skripts dient hauptsächlich zur Kommunikation der Skripte untereinander. In diesem werden Zwischengrößen gespeichert und teilweise auch berechnet oder auch die Datenbank aufgerufen, um Größen zu importieren und abzuspeichern.

Calculate_DB

Das Skript Calculate_DB dient ausschließlich zur Berechnung einer Doppelquerlenkerachse. Es wird vom Main-Skript importiert und aufgerufen, sobald der Achstyp Doppelquerlenker erkannt wird. In diesem Skript werden alle geforderten Größen berechnet und abgespeichert.

Calculate_mulitLink

Das Skript Calculate_multiLink dient ausschließlich zur Berechnung eines aufgelösten Querlenkers. Es wird vom Main-Skript importiert und aufgerufen, sobald der Achstyp multi Link erkannt wird. In diesem Skript werden alle geforderten Größen berechnet und abgespeichert.

table

Dieses Skript importiert, insofern vorhanden, die Tabelle mit den manuell eingetragenen Werten eines schon vorhandenen und bereits berechneten Modells und speichert diese ab.

output

Das letzte Skript visualisiert die Daten aus der Berechnung. Zunächst werden alle gespeicherten Werte importiert und formatiert. Anschließend werden die Tabellen und Graphen erstellt. Je nach Graph werden die Achsbeschriftungen und Achsabschnitte angepasst. Schlussendlich wird die im Arbeitsverzeichnis vorhandene PowerPoint-Vorlage geöffnet und die Graphen und Tabellen eingefügt.

Fehlerprävention

Im Skript sind bis jetzt nur wenige Funktionen zum Vorbeugen von Eingabefehlern integriert. Ein wesentlicher Bestandteil davon ist, dass das Programm erst beendet werden kann, sobald alle erforderlichen Felder mit Werten beschrieben sind. Sollten in diesen Feldern Werte vom falschen Datentyp (z.B. Buchstaben statt Zahlen) stehen, wird dies vom Programm nicht erkannt und es wird zu einer Fehlermeldung in Adams oder in Python kommen.

Das Erweitern und Verbessern dieser Funktionen ist Gegenstand zukünftiger Arbeiten.

3.3.2 Auswertung der Berechnungsdatei aus Adams

Wie bereits im vorhergehenden Unterpunkt angesprochen, werden die Ergebnisdateien (Resultfiles) aus der Adams-Simulation ausgelesen. Zunächst wird der Aufbau einer solchen Datei erläutert:

```
"Adams Car suspension assembly"
</Comment>
</ModelInfo>
<Units angle="rad" length="mm" mass="kg" time="sec" />
<StepMap name="map_001">
<Entity name="time">
<Component name="TIME" unitsValue="sec" id="1" />
</Entity>
<Entity name="_cv_part_XFORM" entity="_cv_part" entType="Part" objectId="2">
<Component name="X" unitsValue="mm" id="2" />
<Component name="Y" unitsValue="mm" id="3" />
<Component name="Z" unitsValue="mm" id="4" />
<Component name="PSI" unitsValue="rad" id="5" />
<Component name="THETA" unitsValue="rad" id="6" />
<Component name="PHI" unitsValue="rad" id="7" />
<Component name="VX" unitsValue="mm/sec" id="356" />
<Component name="VY" unitsValue="mm/sec" id="357" />
<Component name="VZ" unitsValue="mm/sec" id="358" />
<Component name="WX" unitsValue="rad/sec" id="359" />
<Component name="WY" unitsValue="rad/sec" id="360" />
<Component name="WZ" unitsValue="rad/sec" id="361" />
<Component name="ACCX" unitsValue="mm/sec**2" id="710" />
<Component name="ACCY" unitsValue="mm/sec**2" id="711" />
<Component name="ACCZ" unitsValue="mm/sec**2" id="712" />
<Component name="WDX" unitsValue="rad/sec**2" id="713" />
<Component name="WDY" unitsValue="rad/sec**2" id="714" />
<Component name="WDZ" unitsValue="rad/sec**2" id="715" />
</Entity>
<Entity name="whl_wheel_XFORM" entity="testrig.whl_wheel" entType="Part" objectId="4">
<Component name="X" unitsValue="mm" id="8" />
<Component name="Y" unitsValue="mm" id="9" />
<Component name="Z" unitsValue="mm" id="10" />
<Component name="PSI" unitsValue="rad" id="11" />
<Component name="THETA" unitsValue="rad" id="12" />
<Component name="PHI" unitsValue="rad" id="13" />
<Component name="VX" unitsValue="mm/sec" id="362" />
<Component name="VY" unitsValue="mm/sec" id="363" />
<Component name="VZ" unitsValue="mm/sec" id="364" />
<Component name="WX" unitsValue="rad/sec" id="365" />
<Component name="WY" unitsValue="rad/sec" id="366" />
<Component name="WZ" unitsValue="rad/sec" id="367" />
<Component name="ACCX" unitsValue="mm/sec**2" id="716" />
<Component name="ACCY" unitsValue="mm/sec**2" id="717" />
<Component name="ACCZ" unitsValue="mm/sec**2" id="718" />
<Component name="WDX" unitsValue="rad/sec**2" id="719" />
<Component name="WDY" unitsValue="rad/sec**2" id="720" />
<Component name="WDZ" unitsValue="rad/sec**2" id="721" />
```

Bild 17: Auszug aus einem Resultfile von Adams

Am Anfang der Datei werden alle Größen, die Adams auswertet, mit Name, Einheit und einer internen ID dokumentiert. Diese Auflistung befolgt keine ersichtliche Reihenfolge, wie in Bild 17 erkennbar ist. Nach dieser Auflistung folgen verschiedene Blöcke, welche ausschließlich aus Zahlen bestehen. Jeder Zahlenblock enthält alle Werte für einen Step (Berechnungsschritt) sortiert nach den IDs.

Ein kurzes Beispiel: Es sei der Wert für cv_Part_XFORM in z-Richtung gesucht, zu sehen auf Bild 17. Dieser Wert besitzt die ID-Nummer 4. In jedem Zahlenblock ist der vierte Eintrag der Wert für den gesuchten Wert. In diesem Fall, wie im Bild 18 dargestellt, im ersten Berechnungsschritt 0.

```
<Step type="quasiStatic">
1
0 0 0 0 0 0
-1.75979566977013713 -799.59092400310873927 -79.47499999999988063 -0.00551518498880035 1.58928163714628945
0.01089816372169194
-6.4165314587598381 795.52862292562019775 80.52499999999920988 -3.13469414239171851 1.53981594648740527
-3.1181720527544412
-0.09200000000003919 -805.99500000000000455 -464.90851980598586124 0 0 0
-0.09200000000003919 805.99500000000000455 -304.78178798502750624 0 0 0
-0.092 -805.99500000000000455 -784.90851980598586124 0 0 0
-0.092 805.99500000000000455 -624.78178798502744939 0 0 0
0 -3.21624529935327342E-14 -1.22464679914735322E-14 0 0 0
0 -3.21624529935327342E-14 7.49944519773628735E-12 0 0 0
0 -3.21624529935319264E-14 -0.05278820539726618 0 0 0
0 -3.21624529935335483E-14 7.47495226175335465E-12 0 0 0
18.51899999999999835 -371.30000000000001137 6.87399999999999967 0 1.57079632679489656 0
```

Bild 18: Auszug aus einem Zahlenblock

Im Skript werden zunächst alle Resultdateien (Dateien, die mit .res enden) eingelesen. Je nach Name werden diese der jeweiligen Simulation zugeordnet. Alle auszulesenden Größen haben im Befehlsskript einen eineindeutigen Namen erhalten, sodass im Resultfile nach diesen Werten gesucht, die interne ID zugeordnet und anschließend abgespeichert werden kann. Mithilfe dieser ID kann jeder Wert aus den Zahlenblöcken ausgelesen werden.

3.3.3 Berechnung der geforderten Größen mittels Python

Lenkachse

Die Position und Richtung der Lenkachse ist eine wichtige Größe jeder Achse. Aus ihr werden Größen wie der Spreizungswinkel, der Nachlaufwinkel oder der Lenkrollradius abgeleitet. Die Bestimmung der Lenkachse hängt stark vom Achstyp ab. Bei einer Doppelquerlenkerachse oder einer McPherson wird sie durch den äußeren Punkt des oberen Querlenkers bzw. des Domlagers (McPherson) und dem äußeren Punkt des unteren Querlenkers bestimmt. Beide Achstypen besitzen eine geometrische Lenkachse.

Bei einer Achse mit einem oder mehreren aufgelösten Querlenkern spricht man von einer virtuellen Lenkachse, da die Bestimmung nicht geometrisch darstellbar ist (s. 3.2.2). In diesem Fall wird eine Methode zur Berechnung benötigt. Liegen die Lenker in einer Ebene, können diese verlängert und somit der Schnittpunkt für die Lenkachse bestimmt werden. Ist dies, wie bei Modell 2 nicht der Fall, muss dieser Punkt approximativ ermittelt werden.

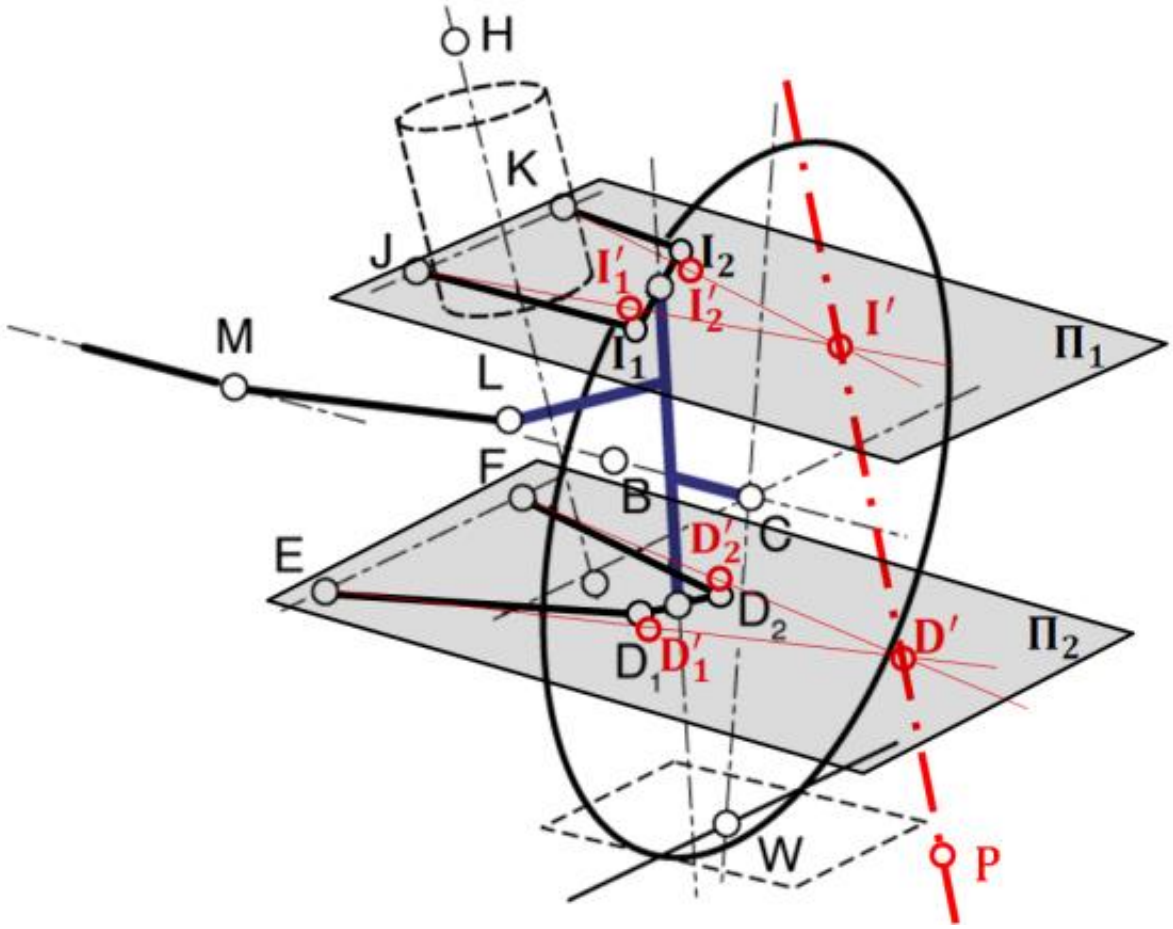


Bild 19: Ermittlung der Lenkachse (Bsp. Fünflenkerachse) vgl. [6]

Im Bild 19 ist eine Fünflenkerachse, bei welcher die einzelnen Lenker nicht in einer Ebene liegen, abgebildet. Die Methode zur Ermittlung der Lenkachse lautet wie folgt:

„I_1 und I_2 werden miteinander verbunden, diese Verbindungsstrecke wird halbiert und genau in der Mitte ein Hilfspunkt definiert. Drei Punkte legen eine Ebene eindeutig fest. Mit diesem Hilfspunkt und mit den Punkten J und K wird eine Hilfsebene Π_1 konstruiert. Auf diese Hilfsebene werden die originalen Lenkerpunkte I_1 und I_2 rechtwinklig projiziert, so erhält man die Punkte I'_1 und I'_2. Nun hat man erreicht, dass die Punkte I'_1 und I'_2 mit *J* und *K* stets auf einer Ebene, nämlich genau Π_1 liegen. Jetzt können die Geraden $(\overline{J\,I'_1},\ \overline{K\,I'_2})$ verlängert werden, um den Schnittpunkt I′ zu erhalten. Analog ergibt sich Punkt D′ zu der virtuellen Spreizachse der Fünflenkerachse.“ [6] S. 61

Dieses Verfahren gilt als Annäherung zur Bestimmung der Spreizachse (Lenkachse) einer Fünflenkerachse. Diese Methode wurde für das zweite Modell angepasst und in das Python-Skript entsprechend implementiert. Da nur ein Dreieckslenker bestimmt werden musst, bleibt das Ergebnis weiterhin eine Annäherung, ist aber präziser als bei der Fünflenkerachse.

Im Folgenden wird erläutert, wie die Größen laut Anforderungsliste berechnet werden und im Skript implementiert sind.

Sturz γ

Um den Sturz zu berechnen, benötigt man eine Projektion der Radmittelebene in die y-z-Ebene. Da Adams die Radmittelebene nicht direkt als einzelnen Wert ausgibt und die Projektion noch ein zusätzlicher Schritt wäre, ist diese Methode ungeeignet.

```
# Berechnung Sturz
camber.append(np.arctan(-(WCy_l[i] - TCPy_l[i]) / (WCz_l[i] - TCPz_l[i])))
camber_deg.append(180 / np.pi * camber[i])  # Umrechnung in Gradmass
```

Bild 20: Berechnung des Sturzes in Python

$$\gamma = \arctan\left(\frac{-(Radmitte\ links[y] - Radaufstandspunkt\ links\ [y])}{Radmitte\ links[z] - Radaufstandspunkt\ links\ [z]}\right) \quad (3.01)$$

Im Skript wird der Sturz über den Radmittelpunkt und den Radaufstandspunkt gebildet wie in Bild 20 dargestellt. Zwischen diesen beiden Punkten wird eine Gerade in y-z-Ebene gelegt, von welcher zunächst nur der Anstieg bestimmt wird, wie in Bild 21 zu sehen. Anschließend wird vom negativen Anstieg (siehe Definition Sturz) der Arcus-Tangens (Schnittwinkel mit Straßenebene projiziert in y-z-Ebene) gebildet. Dieses Ergebnis entspricht dem Sturz wie in Formel 3.01 beschrieben.

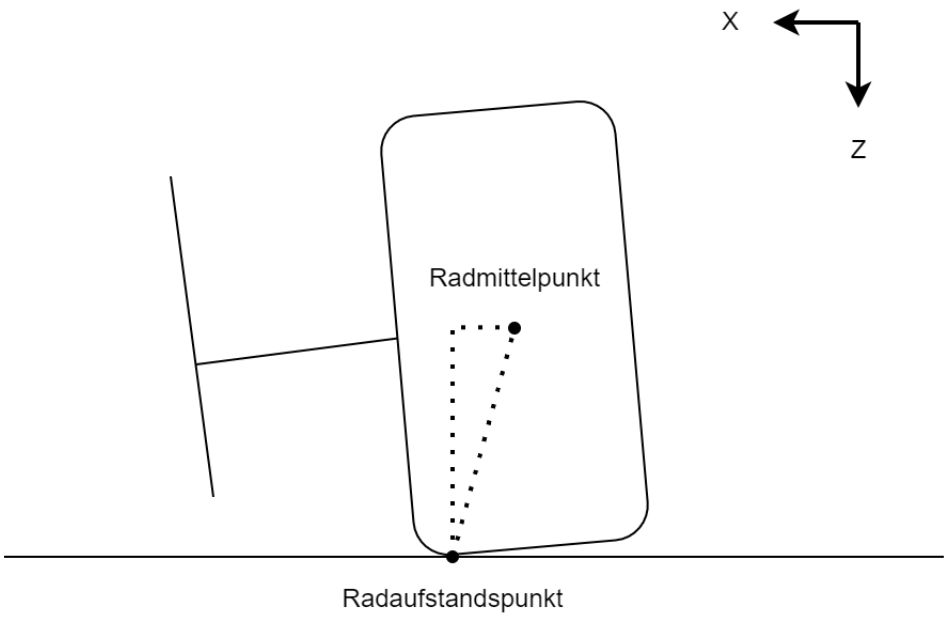


Bild 21: Ermittlung des Sturzes

Spurwinkel δ

Der Spurwinkel ist die einzige Größe, die nicht im Skript berechnet wird, sondern über die Rotation des Markers Wheelcenter direkt aus dem Adams-Resultaten übernommen werden kann, wie in Bild 22 dargestellt.

```
# Berechnung Spur
toe.append(-WP_xrot_l[i])
toe_deg.append((180 / np.pi * toe[i]) * 60)  # Umrechnung in Gradmass und in Gradminuten
```

Bild 22: Berechnung der Spur in Python

Hier ist allerdings folgendes zu beachten:

Nach der globalen Orientierung des Koordinatensystems entspricht die Spur der Rotation um die y-Achse. Jedoch dreht Adams in einer Routine den Marker so, dass die Rotation um die x-Achse betrachtet werden muss. Im Skript wird diese Besonderheit mitberücksichtigt. Bild 23 zeigt die eigentliche Orientierung des Markers in Adams.

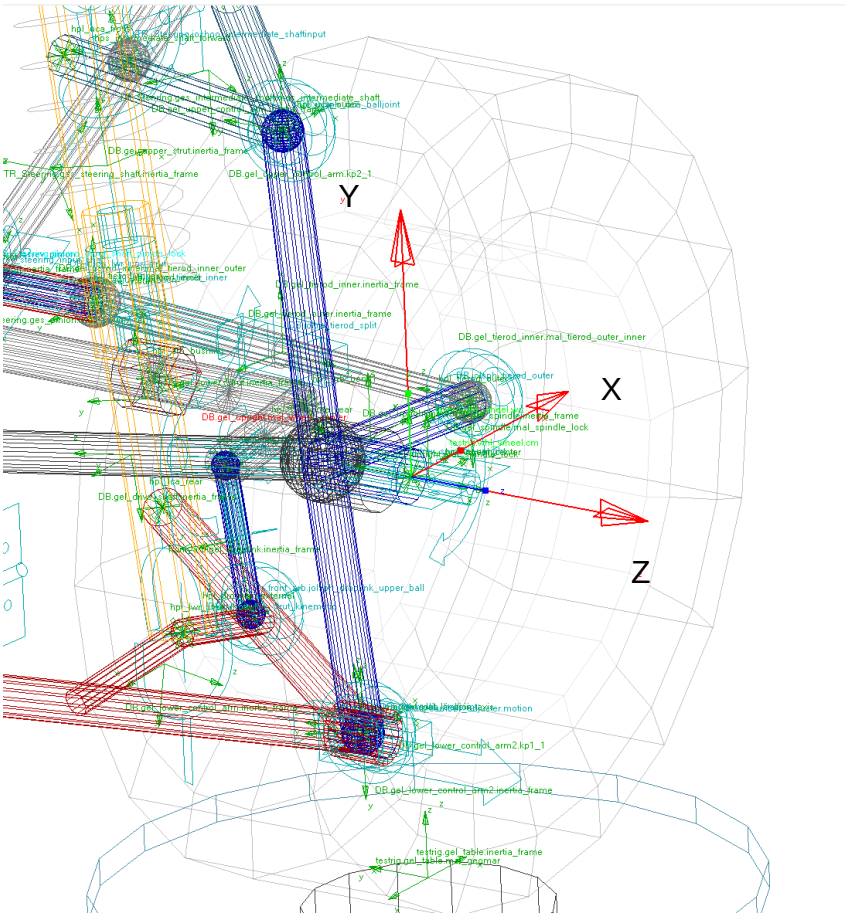


Bild 23: Orientierung Marker Wheelcenter (isometrische Frontansicht)

Nachlaufwinkel τ und Nachlaufstrecke n

Der Nachlaufwinkel ergibt sich aus der projizierten Lenkachse in x-z-Ebene und deren Schnittwinkel mit der Straßenebene. Diese Methode ähnelt jener zur Bestimmung des Sturzes, jedoch mit einer anderen Winkelfunktion wie in Formel 3.02 beschrieben.

```
# Schnittwinkel zwischen Lenkachse und Straßenebene
caster_angle.append(np.arccos(norm(np.dot(k, np.array([s[i][0], 0, s[i][2]])))
                              / (norm(k) * norm(np.array([s[i][0], 0, s[i][2]])))))
# Umrechnung in Gradmass
caster_angle_deg.append(180 / np.pi * caster_angle[i])
```

Bild 24: Berechnung des Nachlaufwinkels in Python

$$\tau = \arccos\left(\frac{|Straßennormalenvektor \cdot Lenkachse_{proj.}|}{|Straßennormalenvektor| \cdot |Lenkachse_{proj.}|}\right) \tag{3.02}$$

Im Skript, welches in Bild 24 dargestellt ist, wird die Lenkachse als s bezeichnet und k ist der Normalenvektor der Straßenebene.

Mit dem Nachlaufwinkel, dem Nachlaufversatz (Abstand der Lenkachse zum Radmittelpunkt) und dem jeweiligen Radhalbmesser kann die Nachlaufstrecke bestimmt werden. Jedoch wird der Nachlaufversatz nicht berechnet und müsste deswegen zusätzliche implementiert werden. Deswegen wird die Nachlaufstrecke mit bereits bekannten Größen berechnet, wie in Formel 3.03 beschrieben wird.

```
if KIP[i][0] + (-KIP[i][1] / (1 / np.tan(-WP_xrot_l[i]))) <= TCPx_l[i] + (
        -TCPy_l[i] / (1 / np.tan(-WP_xrot_l[i]))):
    # Bedingung zum Erkennen des Vorzeichens, da sonst keine negativen Werte entstehen können
    caster_arm.append(
        norm(np.cross(KIP[i] - np.array([TCPx_l[i], TCPy_l[i], TCPz_l[i]]),
                      np.array([1, 1 / np.tan(-WP_xrot_l[i]), 0])))
        / norm(np.array([1, 1 / np.tan(-WP_xrot_l[i]), 0])))
else:
    caster_arm.append(
        -norm(np.cross(KIP[i] - np.array([TCPx_l[i], TCPy_l[i], TCPz_l[i]]),
                       np.array([1, 1 / np.tan(-WP_xrot_l[i]), 0])))
        / norm(np.array([1, 1 / np.tan(-WP_xrot_l[i]), 0])))
# Abstand in x-Richtung in Radmittelebene zwischen Radaufstandspunkt und Lenkachse für Durchstoßpunkt
```

Bild 25: Berechnung der Nachlaufstrecke in Python

$$n = \frac{|(Durchstoßpunkt - Radaufstandspunkt) \times Radmittelebene|}{|Radmittelebene|} \tag{3.03}$$

Die Formel 3.03 ist eine Abstandsformel. Sie beschreibt den Abstand des Durchstoßpunktes und des Radaufstandspunktes entlang des Vektors der Radmittelebene. Auf Bild 26 wird der Sachverhalt nochmals veranschaulicht.

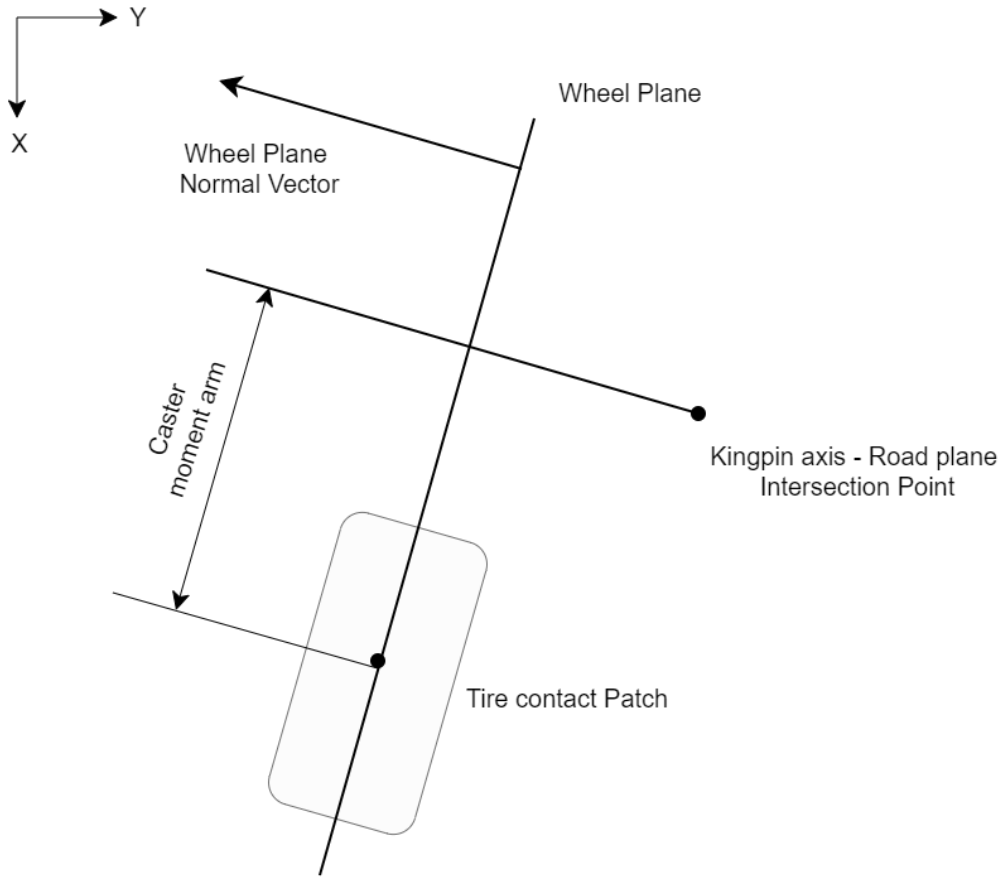


Bild 26: Skizze zur Nachlaufstrecke

Im Skript, welches auf Bild 25 abgebildet ist, steht vor der eigentlichen Berechnung eine Bedingung. Diese wird benötigt, um das Vorzeichen des Nachlaufs zu erkennen, da ein Abstand immer positiv ist. Diese Bedingung projiziert beide Punkte auf die Y-Achse in Richtung der Radmittelebene. Anschließend wird verglichen, welcher projizierte Punkt näher am Koordinatenursprung liegt. Somit wird laut theoretischer Definition das Vorzeichen vergeben.

Die Bezeichnung KIP im Skript entspricht dem Kingpin Intersection Point (Durchstoßpunkt der Lenkachse), WP_xrot der Rotation der Radmittelebene um x und TCPx_l dem Tire Contact Patch (Radaufstandspunkt), in diesem Fall der x-Wert auf der linken Seite.

Spreizungswinkel σ

Um den Spreizungswinkel zu berechnen, befolgt man die gleichen Schritte wie beim Nachlaufwinkel mit dem Unterschied, dass die Lenkachse in die y-z-Ebene projiziert wird.

```
# Schnittwinkel zwischen Lenkachse und Straßenebene
kingpin_inclination_angle.append(np.arccos(norm(np.dot(k, np.array([0, s[i][1], s[i][2]])))
                                           / (norm(k) * norm(np.array([0, s[i][1], s[i][2]])))))
# Umrechnung in Gradmaß
kingpin_inclination_anlge_deg.append(180 / np.pi * kingpin_inclination_angle[i])
```

Bild 27: Berechnung des Spreizungswinkels in Python

$$\sigma = \arccos\left(\frac{|Straßennormalenvektor \cdot Lenkachse_{proj.}|}{|Straßennormalenvektor| \cdot |Lenkachse_{proj.}|}\right) \quad (3.04)$$

Im Skript, welches auf Bild 27 abgebildet wird, wird die Lenkachse als s bezeichnet und k ist der Normalenvektor der Straßenebene. Die Berechnung erfolgt laut Formel 3.04, welche ebenso im Skript implementiert ist.

Lenkrollradius r_S

Der Lenkrollradius bildet das Äquivalent zur Nachlaufstrecke ähnlich wie der Spreizungswinkel zum Nachlaufwinkel.

```
# Lenkrollradius
# Bedingung zum Bestimmen des Vorzeichens, da ein Abstand berechnet wird
if KIP[i][1] + KIP[i][0] * (np.tan(-WP_xrot_l[i])) >= TCPy_l[i] + TCPx_l[i] * (np.tan(-WP_xrot_l[i])):
    scrub.append(
        norm(
            np.cross(KIP[i] - np.array([TCPx_l[i], TCPy_l[i], TCPz_l[i]]),
                     np.array([1, -np.tan(-WP_xrot_l[i]), 0])))
        / norm(np.array([1, -np.tan(-WP_xrot_l[i]), 0])))
else:
    scrub.append(
        -(norm(
            np.cross(KIP[i] - np.array([TCPx_l[i], TCPy_l[i], TCPz_l[i]]),
                     np.array([1, -np.tan(-WP_xrot_l[i]), 0])))
         / norm(np.array([1, -np.tan(-WP_xrot_l[i]), 0]))))
```

Bild 28: Berechnung Lenkrollradius in Python

$$r_S = \frac{|(Durchstoßpunkt - Radaufstandspunkt) \times Normalenvektor\ \ Radmittelebene|}{|Normalenvektor\ \ Radmittelebene|} \qquad (3.05)$$

Wie im Skript auf Bild 28 oder auch in der Formel 3.05 abgebildet, ist der Lenkrollradius der Abstand zwischen Durchstoßpunkt der Lenkachse und Radaufstandspunkt entlang des Normalenvektors der Radmittelebene. Auch hier wird eine Bedingung zum Erkennen des Vorzeichens benötigt. Die Definition von positiv und negativ wurde bereits im Kapitel 2 erläutert. In Bild 29 wird die Bestimmung des Lenkrollradius‘ nochmals veranschaulicht.

Die Bezeichnung KIP im Skript entspricht dem Kingpin Intersection Point (Durchstoßpunkt der Lenkachse), WP_xrot der Rotation der Radmittelebene um x und TCPx_l dem Tire Contact Patch (Radaufstandspunkt), in diesem Fall der x-Wert auf der linken Seite.

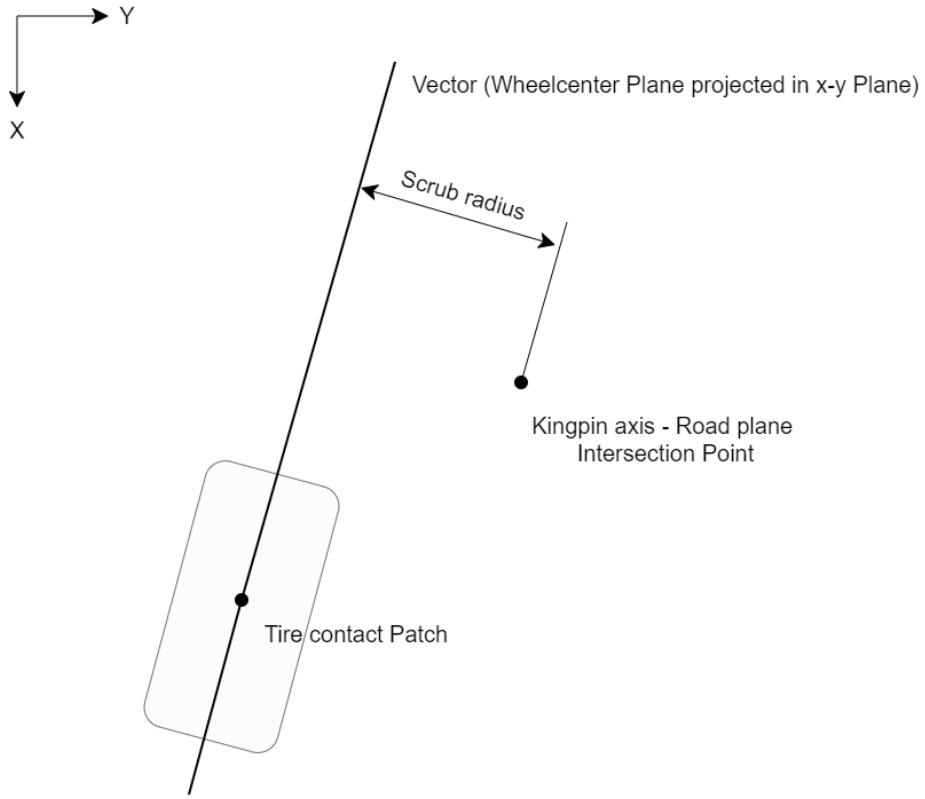


Bild 29: Skizze zum Lenkrollradius

Rollzentrum R_0 und Rollzentrumshöhe

Die geometrische Bestimmung der Rollzentrumshöhe wurde bereits im Kapitel 2 erläutert. Da diese Methode eine Gerade als Vereinfachung für den jeweiligen Querlenker voraussetzt, ist sie für eine allgemeine Berechnung ungeeignet und wird vorwiegend zur theoretischen Erklärung des Rollzentrums verwendet.

Im Programmskript ist folgende Berechnungsmethode implementiert:

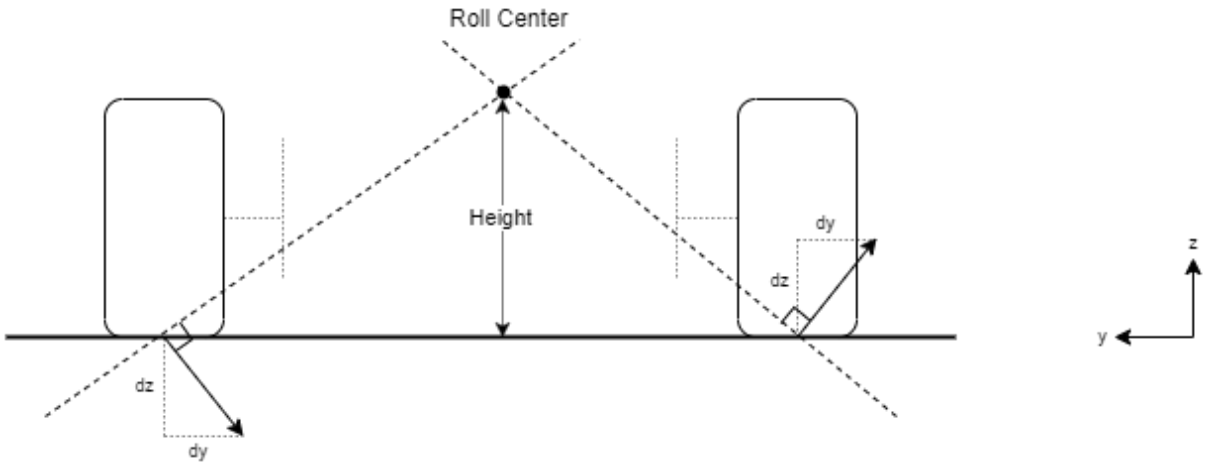


Bild 30: Skizze zum Rollzentrum

Jene Gerade, welche vom Radaufstandspunkt zum Querpol verläuft, ist orthogonal zum Geschwindigkeitsvektor des Radaufstandspunktes. Dieser Geschwindigkeitsvektor kann auf die Bewegung des Radaufstandspunktes in y-z-Ebene reduziert werden, da lediglich die Richtung und nicht der Betrag von Relevanz ist. Im Bild 30 wird der Sachverhalt abgebildet.

```
# Bilden von linearen Funktionen in der y-z-Ebene
m1.append(-((TCPy_l[i] - TCPy_l[i - 1]) / (TCPz_l[i] - TCPz_l[i - 1])))
n1.append(TCPz_l[i] - m1[i] * TCPy_l[i])

# Berechnung der z-Koordinate des Schnittpunktes der Geraden
z1.append(n1[i] - TCPz_l[i])
rollcenter_height.append(z1[i])
```

Bild 31: Berechnung der Rollzentrumshöhe in Python

Im Skript, welches auf Bild 31 abgebildet ist, wird zunächst der Anstieg der Normalen des Geschwindigkeitsvektors ermittelt:

$$m_{Tangente} = \frac{\Delta z_{Radaufstandspunkt}}{\Delta y_{Radaufstandspunkt}} \; und \; m_1 = \frac{-1}{m_2} \tag{3.06}$$

ergibt:

$$m_{Normale} = \frac{-\Delta y_{Radaufstandspunkt}}{\Delta z_{Radaufstandspunkt}} \tag{3.07}$$

Die Radaufstandspunkte werden mit TCP (Tire Contact Point), der jeweiligen Koordinate und der Fahrzeugseite bezeichnet. Anschließend wird das absolute Glied berechnet:

$$f(y) = z = m_{Normale} \cdot y + n_{Normale} \tag{3.08}$$

Nach n umgestellt:

$$n_{Normale} = z - m_{Normale} \cdot y \tag{3.09}$$

Die Rollzentrumshöhe ist der Abstand in z-Richtung zwischen dem Schnittpunkt der Normalen mit der Ordinate und dem Radaufstandspunkt.

Somit:

$$R_0 = n_{Normale} - z_{Radaufstandspunkt} \tag{3.10}$$

Im Bild 30 ist zu sehen, dass nicht der Schnittpunkt mit der z-Achse verwendet wird, sondern der Schnittpunkt mit der Geraden auf der anderen Fahrzeugseite. Diese Methode kann man verwenden, wenn beide Räder gleichzeitig um den gleichen Betrag und Richtung bewegt werden. Der Schnittpunkt der Gerade liegt in diesem Fall auf der z-Achse und entspricht somit dem absoluten Glied.

Diese Berechnungsmethode bietet den Vorteil, dass sie vollkommen achstypunabhängig ist, einen geringen Berechnungsaufwand besitzt und wenige Eingabewerte erfordert. Es gibt keine erwähnenswerten Nachteile.

Bremsabstützwinkel ε_B

Der Bremsabstützwinkel ist der Winkel der projizierten Gerade zwischen Längspol und Radaufstandspunkt und der x-Achse, wie in Bild 33 dargestellt (siehe Kapitel 2.1.7).

```
# Berechnen Bremsabstützwinkel
r, p = symbols("r p")
equation1 = Eq((xu[i] + r * (float(Hardpoint1[0]) - float(Hardpoint2[0]))),
               (xo[i] + p * (float(Hardpoint3[0]) - float(Hardpoint4[0]))))
equation2 = Eq((zu[i] + r * (float(Hardpoint1[2]) - float(Hardpoint2[2]))),
               (zo[i] + p * (float(Hardpoint3[2]) - float(Hardpoint4[2]))))

solution = solve((equation1, equation2), (r, p))
SPad.append(np.array([(xu[i] + solution[r] * (Hardpoint1[0] - Hardpoint2[0])),
                      (zu[i] + solution[r] * (Hardpoint1[2] - Hardpoint2[2]))]))

m11.append(np.tan(((TCPz_l[i] - float(SPad[i][1])) / (TCPx_l[i] - float(SPad[i][0])))))
anti_dive_angle.append(180 / np.pi * m11[i])
```

Bild 32: Berechnung des Bremsabstützwinkels in Python

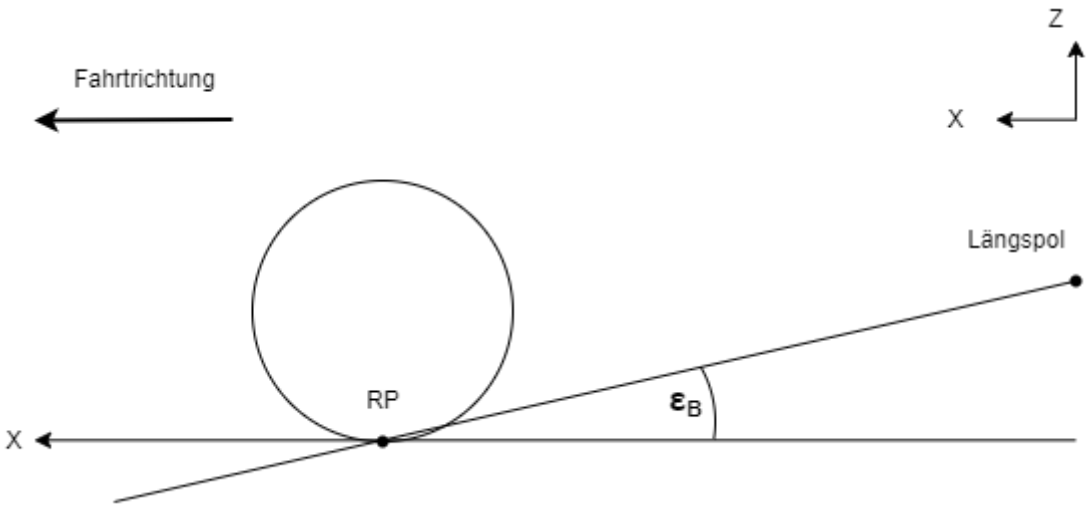


Bild 33: Skizze zum Bremsabstützwinkel

Die Umsetzung in Python erfolgt zunächst durch die Bestimmung des Längspols.

Zur Berechnung werden jeweils die drei Punkte des Dreieckslenkers benötigt. Die im Skript bezeichnete Variable Hardpoint gibt jeweils einen Befestigungspunkt des Dreiecklenkers an der Karosserie an. Die Variablen x bzw. z (Index u steht für unten und o für oben) geben den äußeren Punkt des Dreiecklenkers an, welcher auch zur Bestimmung der Lenkachse erforderlich ist.

Zuerst wird der Anstieg in x-z-Ebene zwischen den jeweiligen Befestigungspunkten der Querlenker berechnet. Anschließend wird eine Gerade in x-z-Ebene mit diesem Anstieg gebildet, welche durch den äußeren Punkt des Querlenkers verläuft. Der Schnittpunkt dieser Geraden ist der Längspol. Auf Bild 34 wird die Bestimmung veranschaulicht. Im Skript auf Bild 32 wird dieser als SPad bezeichnet.

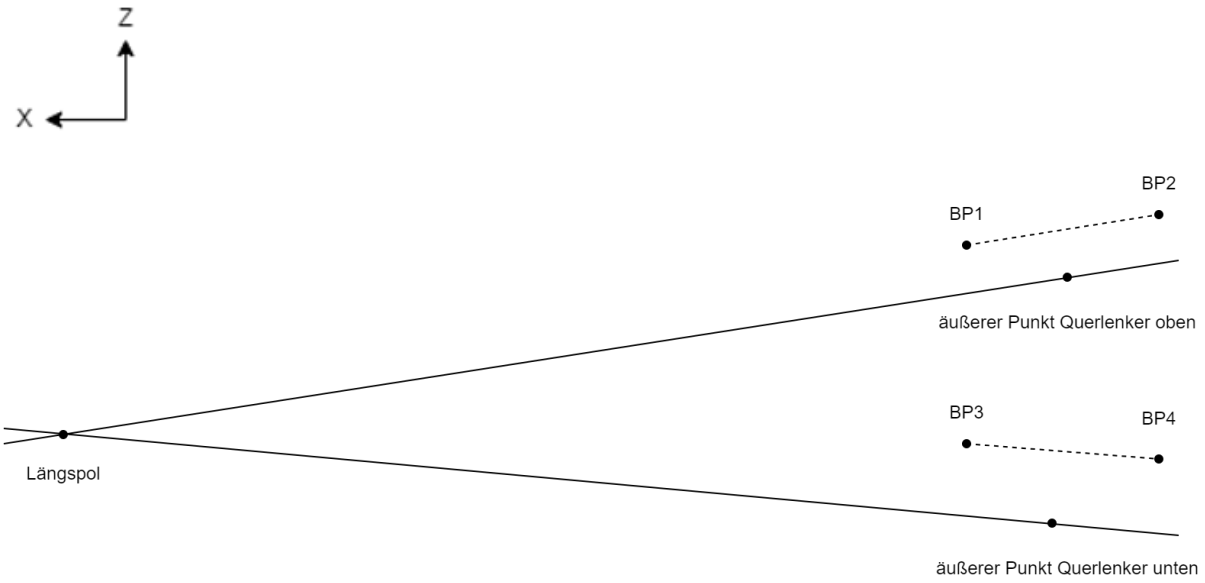


Bild 34: Skizze zum Längspol

Abschließend wird der Schnittwinkel der Geraden zwischen dem Längspol und dem Radaufstandspunkt berechnet.

Ackermannwinkel δ_{AM} und Ackermannfehler δ_F

Um den Ackermannwinkel zu bestimmen, werden zunächst einige Zwischengrößen benötigt. Der Bahnradius, welcher in Formel 3.11 berechnet wird, gibt an, wie groß der Radius des gedachten mittleren Rades ist (wie bei Einspurmodell, Fahrzeug wird auf ein Rad in der Fahrzeugmittelebene reduziert).

$$Bahnradius = Radstand + \frac{1}{tan(Radlenkwinkel)} + \left(\frac{|RP\ links| + |RP\ rechts|}{2}\right) \tag{3.11}$$

Der Ackermannwinkel berechnet sich laut Formel 3.12. Dieser Winkel gibt den optimalen Lenkeinschlag für das gedachte mittlere Rad an. Aus diesem Wert können die Lenkwinkel für das kurveninnere und kurvenäußere Rad abgeleitet werden.

$$\delta_A = \cot\left(\frac{Radstand}{Bahnradius}\right) \tag{3.12}$$

Der Ackermannfehler kann nicht auf das gedachte Rad bezogen werden, da er dort null wäre. Deswegen wird im Folgenden das rechte Rad betrachtet. Um diesen Fehler zu berechnen, wird der ideale Lenkwinkel benötigt, welcher in Formel 3.13 berechnet wird.

$$Lenkwinkel_{ideal} = arctan\left(\frac{Radstand}{Bahnradius - RP_y}\right) \tag{3.13}$$

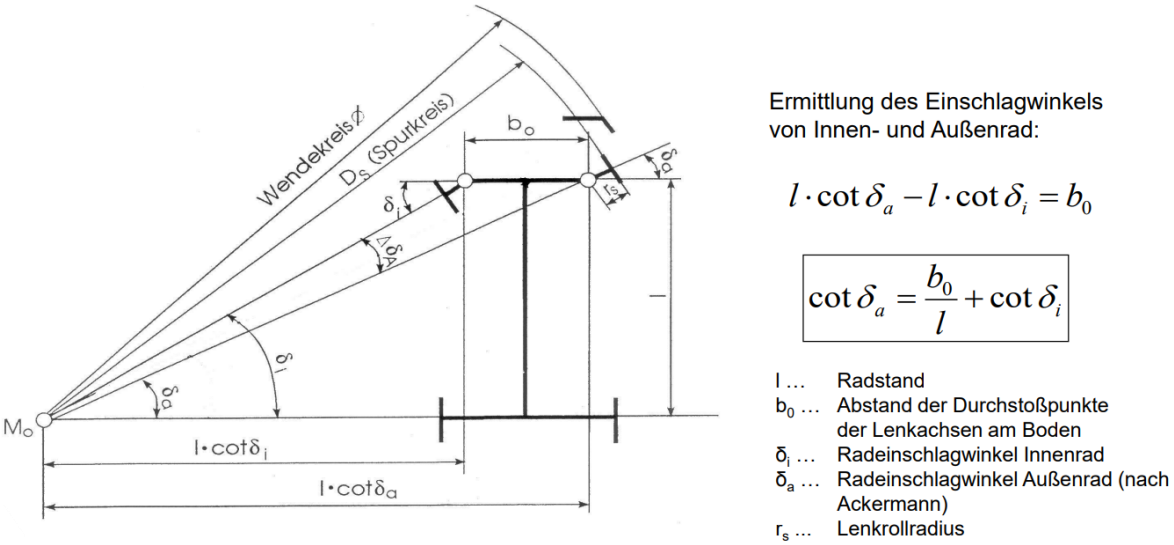
Der Ackermannfehler ist die Differenz vom Lenkwinkel zum idealen Lenkwinkel, wie in Formel 3.14. Diese Differenz gilt nur für das jeweils kurveninnere Rad, da das kurvenäußere den idealen Lenkwinkel besitzt. Wie bereits im Kapitel 2 erläutert, nimmt man diese Differenz in Kauf, da diese sowohl den Kurvenradius als auch den benötigten Bauraum begünstigt.

$$\delta_F = Lenkwinkel - Lenkwinkel_{ideal} \tag{3.14}$$

```
turn_radius_r.append(wheelbase *
                     (1 / np.tan(-WP_xrot_r[i])) + ((norm(TCPy_l[i]) + norm(TCPy_r[i])) / 2))
turn_radius_l.append(wheelbase *
                     (1 / np.tan(-WP_xrot_l[i])) - ((norm(TCPy_l[i]) + norm(TCPy_r[i])) / 2))

if -WP_xrot_l[i] <= 0:
    ackerman_angle.append(np.arctan(wheelbase / turn_radius_l[i]))
    # Ackerman fuer Lenken nach links
else:
    ackerman_angle.append(np.arctan(wheelbase / turn_radius_r[i]))
    # Ackerman fuer Lenken nach rechts
ackerman_angle_deg.append(180 / np.pi * ackerman_angle[i])

# Berechnung der benötigten Größen für Ackermannfehler
steer_angle_left = - WP_xrot_l[i]
steer_angle_right = - WP_xrot_r[i]

ideal_steer_angle_left = np.arctan(wheelbase / (turn_radius_r[i] - TCPy_l[i]))
ideal_steer_angle_right = np.arctan(wheelbase / (turn_radius_r[i] - TCPy_r[i]))

if WP_xrot_r[i] <= 0:
    ackerman_error.append(180 / np.pi * (steer_angle_left - ideal_steer_angle_left))
else:
    ackerman_error.append(180 / np.pi * (steer_angle_right - ideal_steer_angle_right))
```

Bild 35: Berechnung Ackermann in Python

Im Bild 35 ist das Python Skript abgebildet. In diesem werden die Größen Bahnradius (turn_radius) und der Lenkwinkel$_{ideal}$ (ideal_steer_angle) jeweils für links und rechts berechnet. Links und rechts bezieht sich auf den Lenkeinschlag der Räder und nicht auf das Rad.



Ermittlung des Einschlagwinkels von Innen- und Außenrad:

$$l \cdot \cot \delta_a - l \cdot \cot \delta_i = b_0$$

$$\cot \delta_a = \frac{b_0}{l} + \cot \delta_i$$

l … Radstand
b_0 … Abstand der Durchstoßpunkte der Lenkachsen am Boden
δ_i … Radeinschlagwinkel Innenrad
δ_a … Radeinschlagwinkel Außenrad (nach Ackermann)
r_s … Lenkrollradius

Bild 36: Skizze zum Ackermann

Eine weitere geforderte Größe ist der Spurkreisradius, welcher in Formel 3.15 berechnet wird. Wie in Bild 36 dargestellt, ist der Spurkreisradius ausgehend vom kurvenäußeren Rad. Bild 37 veranschaulicht die Implementierung im Skript. In der Auswertung wird jedoch nur der kleinste Spurkreis betrachtet (optimal bei größtem Lenkeinschlag).

```
outside_turn_radius.append(2 * np.sqrt(np.square(turn_radius_r[i] + ((norm(TCPy_l[i]) + norm(TCPy_r[i]))
                                                                     / 2)) + np.square(wheelbase)))
```

Bild 37: Berechnung Spurkreisradius in Python

$$Spurkreis = 2 * \sqrt{\left(Bahnradius + \left(\frac{|RP\ links| + |RP\ rechts|}{2}\right)\right)^2 + Radstand^2} \qquad (3.15)$$

Feder- und Dämpferübersetzung

Für die Feder- und Dämpferübersetzung wurde ein zusätzliches Request im Befehlsskript definiert. Dieses gibt die Verschiebung des oberen und des unteren Punktes an der Feder- bzw. dem Dämpfer an. Im Skript wird, wie auf Bild 38 abgebildet wird, lediglich die Änderung pro Berechnungsschritt und damit die Übersetzung berechnet.

```
spring_ratio.append((slt[i - 1] - slt[i]) / (WCz_l[i] - WCz_l[i - 1]))
damper_ratio.append((dlt[i - 1] - dlt[i]) / (WCz_l[i] - WCz_l[i - 1]))
```

Bild 38: Berechnung der Feder- und Dämpferübersetzung in Python

Lenkübersetzung

Die kinematische Lenkübersetzung berechnet sich laut Formel 3.16.

$$i_S = \frac{Lenkradwinkel}{Radlenkwinkel\ gemittelt} \tag{3.16}$$

Die Berechnung im Programm erfolgt wie im Skript auf Bild 39 dargestellt. Bei einem Lenkwinkel von 0° wird ein Wert interpoliert, da dieser theoretisch null betragen würde. In der Realität ist dieser Wert nicht null, da dies bedeuten würde, dass es um die Nulllage keine Lenkübersetzung geben würde.

```
# Berechnung Lenkübersetzung
if i == 0:
    steer_ratio.append('')
else:
    c = (config.Steerlwr + i * ((config.Steerupr - config.Steerlwr) / config.Steps))
    steer_ratio.append(c / ((((-WP_xrot_l[i]) + (-WP_xrot_r[i])) / 2) * 180 / np.pi))
    if steer_ratio[i] == 0:
        steer_ratio[i] = steer_ratio[i - 1] + (steer_ratio[i - 1] - steer_ratio[i - 2])
```

Bild 39: Berechnung Lenkübersetzung in Python

3.4 Auswertung und Diskussion der Ergebnisse

In Anlage 2 und 3 befinden sich die PowerPoint-Präsentationen von Modell 1 und Modell 2 inkl. der Ergebnisse.

In beiden Präsentationen befinden sich zwei Auswertungsdarstellungen. Die erste Darstellung trägt den Namen des Assemblys, welches in das Programm eingelesen wurde und spiegelt die berechneten Werte aus dem Programm wider. Die zweite Darstellung verwendet den Namen Table und repräsentiert die Werte, welche Adams berechnet hat. Diese Werte wurden aus Adams exportiert und in die entsprechende Tabellenvorlage im Excel-Format eingetragen.

Die Ergebnisse aus Adams wurden zur Validierung des Programms verwendet.

Die Feder-Dämpfer-Übersetzung wird in Adams nicht berechnet und somit auch nicht ausgegeben. Da die Werte unverändert von Adams nach Python übertragen wurden, entfällt der Vergleich in beiden Präsentationen und es wird nur der Wert aus dem Programm angegeben.

Alle Werte beziehen sich auf das Beispielmodell, welches keiner realen Achse entspricht. Unrealistische Werte oder Verläufe sind somit nicht als Fehlfunktion des Programms zu bewerten.

Modell 1

Wie in Anlage 2 zu sehen, gibt es in allen Tabellen und Graphen des Modell 1 (Doppelquerlenkerachse) kaum Abweichung zwischen den berechneten Ergebnissen vom Python-Skript und denen von Adams. In Bild 40 und 41 wird ein Auszug der erstellten Präsentation dargestellt. Bild 40 veranschaulicht den Verlauf des Sturzes über den Radhub. Verglichen werden dabei die Werte, welches das Programm berechnet hat (blaue Kurve) und die Werte, welche aus Adams exportiert wurden (orange Kurve, liegt exakt auf der blauen Kurve). In Bild 41 wird die Tabelle der Vorderachskinematik dargestellt. Wie auch im Diagramm werden hier ebenfalls die beiden Werte verglichen. Alle berechneten Werte stimmen exakt überein.

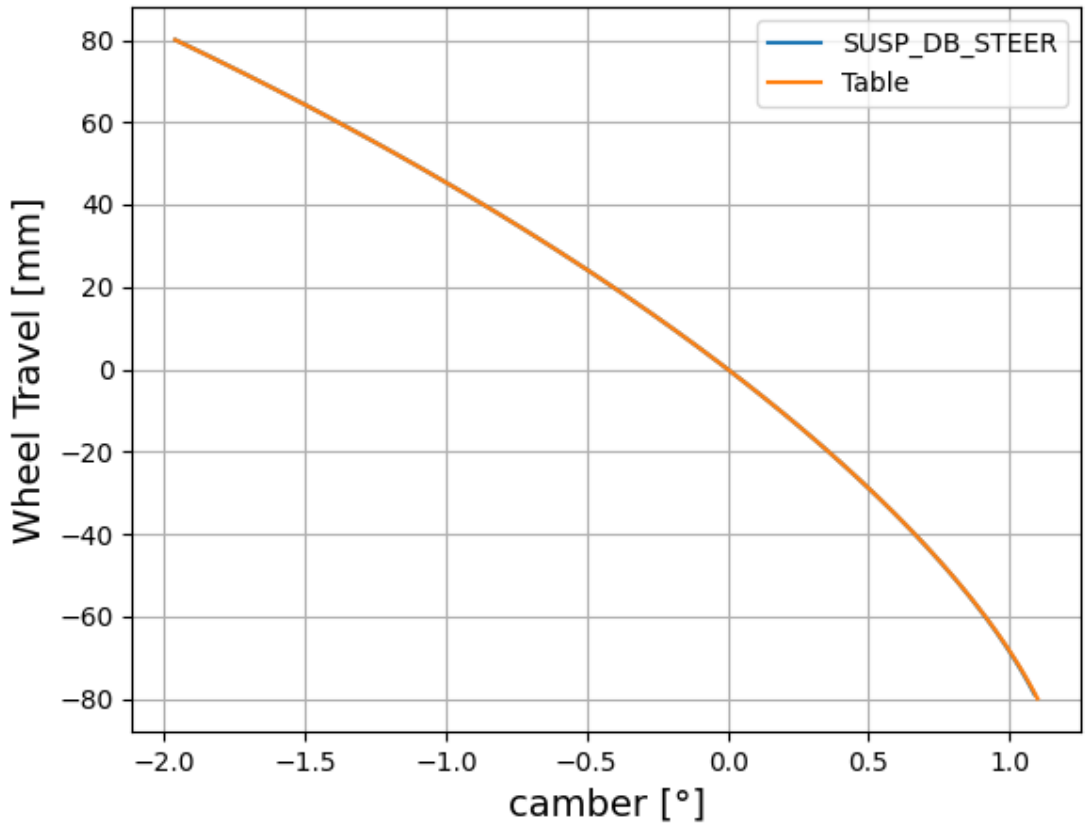


Bild 40: Graph aus Anlage 2

Designation	Unit	SUSP_DB_STEER	Table
Toe	[']	0	0
Camber	[°]	-0	0
Caster angle	[°]	5.38	5.38
Caster arm	[mm]	9.41	9.41
Spring ratio	[-]	0.516	0.516
Damper ratio	[-]	0.516	0.516
Rollcenter height	[mm]	155	154
Anti dive angle	[°]	0.43	0.43

Bild 41: Tabelle aus Anlage 2

Modell 2

Die Werte und Graphen des zweiten Modells (Doppelquerlenkerachse mit aufgelöstem Dreieckslenker unten) sind in Anlage 3 dargestellt. Auch bei diesem Modell gibt es kaum Abweichungen, außer bei jenen Größen, die die Lenkachse zur Berechnung benötigen. Dies betrifft den Lenkrollradius (scrub), welcher im Bild 42 über den gemittelten Lenkwinkel dargestellt wird, den Nachlaufwinkel und damit auch die Nachlaufstrecke (caster angle und caster moment arm) und den Spreizungswinkel (kingpin inclination angle). Im Bild 43 werden diese Werte tabellarisch dargestellt. Die Differenz ergibt sich, da Adams eine andere Methode zur Berechnung der Lenkachse verwendet. Der Lösungsweg ist nicht ausführlich genug, um eine Bewertung über deren Genauigkeit abzugeben. Eine Rücksprache mit dem Adams-Support hat ergeben, dass es durchaus zu Abweichungen in der Berechnung mit der von Adams verwendeten Methode kommen kann. Zum jetzigen Zeitpunkt wird dieses Problem untersucht.

Die in dieser Arbeit verwendete und im Unterpunkt 3.3.3 erläuterte Methode wurde im CAD-Programm Catia für mehrere Berechnungsschritte nachkonstruiert und stimmte exakt mit den vom Programm berechneten Ergebnissen überein.

Alle anderen Abweichungen liegen im tolerierbaren Bereich.

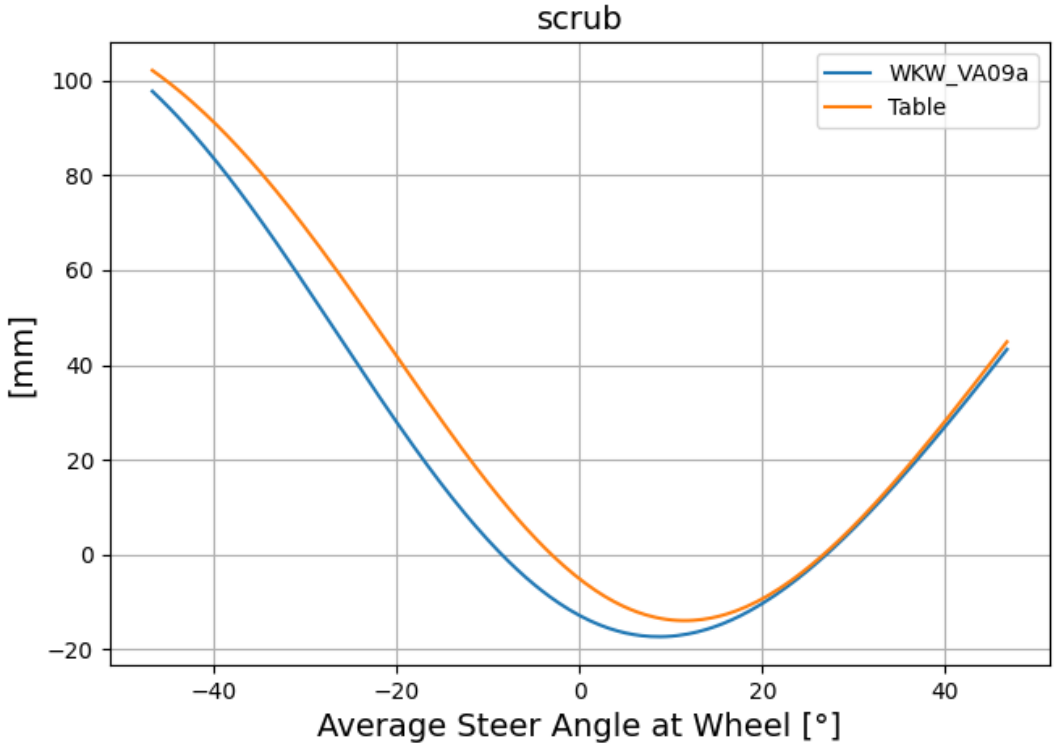


Bild 42: Graph aus Anlage 3

Designation	Unit	WKW_VA09a	Table
Ackermann error	[°]	8.02	7.94
Caster arm	[mm]	33.97	34.39
Caster angle	[°]	4.63	4.75
Scrub	[mm]	-12.26	-5.17
Steer ratio	[-]	15.23	15.23
Kingpin inclination angle	[°]	5.49	5.23
Outside turn diameter	[mm]	10123	10103

Bild 43: Tabelle aus Anlage 3

4 Zusammenfassung

Gegenstand dieser Arbeit war es ein Programm zu entwickeln, welches virtuelle Achssimulationen im MKS-Programm Adams automatisiert und anschließend anhand der Simulationsdaten extern achsspezifische Größen berechnet und darstellt.

Durch dieses Tool werden Entwicklungsprozesse vereinfacht und verbessert, indem ein einfacher Vergleich mehrerer Varianten ermöglicht und fehlerfrei visualisiert wird.

Anhand einer Doppelquerlenkerachse und einer Doppelquerlenkerachse mit aufgelösten unteren Dreieckslenker sollte eine vollständige Achsanalyse gemäß Anforderungsliste durchgeführt werden. Diese beiden Vorderachsen sind gängige Achstypen und decken ein breites Spektrum an weiteren Varianten ab.

Mithilfe einer GUI werden alle benötigten Daten an das Programm übermittelt. Dabei besteht die Möglichkeit extern Daten aus einer Tabelle zu importieren. Das Programm erstellt anhand der Eingabe ein Befehlsskript, welches in Adams eingelesen und umgesetzt wird. Die erzeugten Daten der Simulation in Adams werden wiederum eingelesen und zur Berechnung verwendet. Abschließend werden die Ergebnisse graphisch und tabellarisch in einer PowerPoint-Präsentation ausgegeben.

Um die geforderten Größen gemäß Anforderungsliste korrekt zu berechnen, wurde zunächst eine Vorbetrachtung der theoretischen Berechnung durchgeführt. In dieser sind sowohl die Berechnungsmethoden als auch die Bedeutung der jeweiligen Größe mit allgemeinen Werten und Definition des Vorzeichens aufgelistet.

Anschließend wird beschrieben, wie das entwickelte Programm aufgebaut ist. Dabei wird das prinzipielle Vorgehen in Python und Adams erläutert und die Umsetzung in Python dokumentiert.

Das abschließende Fazit stellt die berechneten Ergebnisse aus dem Programm mit denen aus Adams für die jeweiligen Modelle gegenüber. Dabei wurde als Vergleichsmodell jeweils die Werte aus Adams in eine Exceltabelle eingetragen und anschließend mit den extern berechneten Werten verglichen. Dieser Vergleich dient unter anderem zur Plausibilisierung der errechneten Werte.

Alle berechneten Ergebnisse wurden anhand der Adams-internen Auswertung überprüft und validiert.

Bei dieser Überprüfung fielen zwei Punkte auf:

1. Aufgrund einer Adams-internen Routine wird der Marker am Wheelcenter während der Simulation so gedreht, dass die Spur nicht mehr der ursprünglichen Rotation um die jeweilige Achse entspricht. Zum Zeitpunkt der Erstellung der Arbeit wird dieses Problem vom Adams-Support überprüft.
2. Bei der Berechnung der virtuellen Lenkachse am Bespiel vom Modell 2 mit aufgelöstem Dreieckslenker unten verwendet Adams eine andere Berechnungsmethode, sodass es zu Abweichungen zu den extern berechneten Werten kommt. Eine Aussage über die Genauigkeit der in Adams verwendeten Methode ist aufgrund fehlender Dokumentation nicht möglich. Auch hier überprüft der Adams-Support das Problem.

Das im Zuge der Diplomarbeit entwickelte Programm zur automatischen Auswertung von Achsanalysen ist ein wichtiger Teil der allgemeinen Achsauslegung, da Varianten effizient und fehlerfrei bewertet werden können. Alle Punkte der Anforderungsliste wurden zur vollen Zufriedenheit erfüllt.

Literaturverzeichnis

[1] Heißig Bernd, Ersoy Metin, Gies Stefan (Hrsg.) (2013), Fahrwerkhandbuch, 4. Aufl., München, Lemförde, Wolfsburg

[2] Hexagon: Adams

https://www.mscsoftware.com/de/product/adams Stand: 07.06.2022

[3] Retesco: Was ist Python?

https://www.retresco.de/ressourcen/lexikon/lexikoneintrag/python Stand: 07.06.2022

[4] Statista: Die beliebtesten Programmiersprachen weltweit laut PYPL-Index im Mai 2022

https://de.statista.com/statistik/daten/studie/678732/umfrage/beliebteste-programmiersprachen-weltweit-laut-pypl-index/ Stand: 07.06.2022

[5] Dipl.-Ing. (FH) Stefan Luber / Nico Litzel: Was ist Python?

https://www.bigdata-insider.de/was-ist-python-a-730480/#:~:text=Bei%20Python%20handelt%20es%20sich,Monty%20Python's%20Flying%20Circus%E2%80%9C%20ab Stand: 07.06.2022

[6] Diplomarbeit von David Ferenc Böhm: Integrierte Darstellungs- und Simulationsmethoden in der konzeptionellen Fahrwerksentwicklung

https://diglib.tugraz.at/download.php?id=576a81829ad9f&location=browse Stand: 20.06.2022

Internetquellen

https://www.delftstack.com/de/howto/python/how-to-find-files-with-certain-extension-only-in-python/

Stand: 08.03.2022

https://www.delftstack.com/de/howto/python/python-find-string-in-file/#:~:text=Ausgabe%3A%20Copy%20True-,Verwenden%20der%20Methode%20find%20zum%20Suchen%20einer%20Zeichenkette%20in%20einer,wird%20die%20gew%C3%BCnschte%20Zeichenkette%20%C3%BCbergeben

Stand: 08.03.2022

https://realpython.com/python-strings/

Stand: 08.03.2022

https://lanbugs.de/howtos/python-co/python-snippet-in-einer-datei-suchen-und-zeilennummern-zurueckgeben/#

Stand: 08.03.2022

https://www.delftstack.com/de/howto/python/python-find-string-in-file/#verwenden-sie-die-methode-file-readlines-um-eine-zeichenkette-in-einer-datei-in-python-zu-finden

Stand: 08.03.2022

https://www.delftstack.com/de/howto/python/position-of-character-in-string/#verwenden-sie-die-funktion-rfind-um-die-position-eines-zeichens-in-einer-zeichenkette-zu-ermitteln

Stand: 08.03.2022

https://edley.de/insights/python-strings/

Stand: 08.03.2022

https://www.adamsmith.haus/python/answers/how-to-catch-and-print-exception-messages-in-python#:~:text=Use%20except%20Exception%20as%20to,object%2C%20which%20can%20be%20printed.

Stand: 08.03.2022

https://www.w3schools.com/python/python_try_except.asp

Stand: 08.03.2022

https://www.geeksforgeeks.org/python-convert-a-list-into-a-tuple/

Stand: 09.03.2022

https://thispointer.com/python-how-to-check-if-an-item-exists-in-list-search-by-value-or-condition/

Stand: 09.03.2022

https://python-pptx.readthedocs.io/en/latest/api/enum/XlChartType.html

Stand: 09.03.2022

https://stackoverflow.com/questions/845058/how-to-get-line-count-of-a-large-file-cheaply-in-python

Stand: 09.03.2022

https://stackoverflow.com/questions/16373887/how-to-set-the-text-value-content-of-an-entry-widget-using-a-button-in-tkinter

Stand: 10.03.2022

https://stackoverflow.com/questions/10588317/python-function-global-variables
Stand: 10.03.2022

https://www.delftstack.com/de/howto/python/write-string-to-a-file-in-python/

Stand: 11.03.2022

https://www.delftstack.com/de/howto/python/how-to-delete-a-file-and directory/#:~:text=Python%203%20l%C3%B6scht.,L%C3%B6schen%20einer%20Datei%20in%20Python,Erlaubnis%20hat%2C%20sie%20zu%20l%C3%B6schen.
Stand: 11.03.2022

https://stackoverflow.com/questions/4488570/how-do-i-write-a-tab-in-python

Stand: 11.03.2022

https://www.delftstack.com/de/howto/python/append-one-string-to-another-in-python/
Stand: 11.03.2022

https://edley.de/insights/python-string-format/#:~:text=Daf%C3%BCr%20musst%20du%20die%20Zeichenkette,es%2C%20alle%20verschiedene%20Datentypen%20auszugeben.

Stand: 11.03.2022

https://www.python-forum.de/viewtopic.php?t=43615

Stand: 15.03.2022

https://docs.python.org/3/library/dialog.html

Stand: 15.03.2022

https://codefather.tech/blog/tuple-to-string-python/#:~:text=The%20simplest%20method%20to%20convert,by%20using%20the%20map%20function.

Stand: 15.03.2022

https://www.python-kurs.eu/tkinter_entry_widgets.php

Stand: 15.03.2022

https://www.pythontutorial.net/tkinter/tkinter-button/

Stand: 15.03.2022

https://python-forum.io/thread-26729.html

Stand: 15.03.2022

https://careerkarma.com/blog/python-valueerror-invalid-literal-for-int-with-base-10/#:~:text=The%20Python%20ValueError%3A%20invalid%20literal,a%20string%20to%20an%20integer.

Stand: 17.03.2022

https://numpy.org/doc/stable/reference/generated/numpy.arcsin.html

Stand: 17.03.2022

https://www.python-forum.de/viewtopic.php?t=23683

Stand: 17.03.2022

https://www.python-kurs.eu/matrix_arithmetik.php

Stand: 17.03.2022

https://www.delftstack.com/de/howto/python/python-convert-list-of-strings-to-ints/
Stand: 17.03.2022

https://matplotlib.org/stable/gallery/index.html

Stand: 17.03.2022

https://www.python-kurs.eu/matplotlib.php

Stand: 17.03.2022

https://studyflix.de/mathematik/abstand-gerade-gerade-2007

Stand: 24.03.2022

https://www.programiz.com/python-programming/methods/list/sort

Stand: 13.04.2022

https://www.delftstack.com/de/howto/matplotlib/set-number-of-plot-ticks/

Stand: 13.04.2022

https://www.codestudyblog.com/sf2002b/0207215049.html

Stand: 20.04.2022

https://www.delftstack.com/de/howto/python/python-truncate-float-python/

Stand: 20.04.2022

https://www.delftstack.com/de/howto/python/python-run-another-python-script/

Stand: 25.04.2022

https://www.geeksforgeeks.org/file-explorer-in-python-using-tkinter/

Stand: 26.04.2022

https://www.delftstack.com/de/tutorial/tkinter-tutorial/tkinter-combobox/

Stand: 26.04.2022

https://de.acervolima.com/offnen-sie-ein-neues-fenster-mit-einer-schaltflache-in-python-tkinter/

Stand: 26.04.2022

https://www.inf-schule.de/software/gui/entwicklung_tkinter/bilder

Stand: 26.04.2022

https://www.delftstack.com/de/howto/python-tkinter/how-to-set-text-of-tkinter-entry-widget-by-using-a-button/

Stand: 26.04.2022

https://www.delftstack.com/de/howto/python/pythonequationsolver/#:~:text=offiziellen%20Dokumentation%20hier.,L%C3%B6sen%20Sie%20algebraische%20Gleichungen%20in%20einer%20Variablen%20mit%20der%20Methode,die%20algebraische%20Gleichungen%20l%C3%B6sen%20kann.

Stand: 05.05.2022

https://www.sqlitetutorial.net/sqlite-python/creating-database/

Stand: 06.05.2022

https://www.delftstack.com/de/howto/python/how-to-delete-a-file-and-directory/#l%25C3%25B6schen-sie-ein-verzeichnis-in-python

Stand: 24.05.2022

http://pymotw.com/2/zipfile/

Stand: 24.05.2022

https://www.python-forum.de/viewtopic.php?t=7312

Stand: 30.05.2022

Anlage 1

Anforderung	Datum	Verantwortlicher
Vergleich von bis zu 3 Achskonfigurationen	24.03.2022	M.Meyer
Eingabemaske für Assemblies und Parameter	24.03.2022	M.Meyer
Datei für Übergabewerte aus Eingabemaske	20.04.2022	M.Meyer
Automatische Erstellung eines Befehlsskripts für ADAMS um notwendige Versuche durchzuführen	24.03.2022	M.Meyer
Sprache des Programms: Englisch	27.04.2022	M.Meyer
Graphische Auswertung der Ergebnisse im geeigneten Format	24.03.2022	M.Meyer
Dokumentation des Programms	24.03.2022	M.Meyer
Kinematik:		
Vorspur in Gradminuten (Verlauf und tabellarische Darstellung in K0)	24.03.2022	M.Meyer
Sturz in Grad (Verlauf und tabellarische Darstellung in K0)	24.03.2022	M.Meyer
Rollzentrumshöhe in Millimetern (Verlauf und tabellarische Darstellung in K0)	24.03.2022	M.Meyer
Feder- und Dämpferübersetzung (Verlauf und tabellarische Darstellung in K0)	24.03.2022	M.Meyer
Nachlaufstrecke in Millimetern (tabellarische Darstellung in K0)	24.03.2022	M.Meyer
Nachlaufwinkel in Grad (tabellarische Darstellung in K0)	24.03.2022	M.Meyer
Verlauf über Radhub in Millimetern (Y-Achse)	24.03.2022	M.Meyer

Lenkkinematik:		
Lenkrollradius in Millimetern (Verlauf und tabellarische Darstellung in K0)	24.03.2022	M.Meyer
Nachlaufstrecke in Millimetern (Verlauf und tabellarische Darstellung in K0)	24.03.2022	M.Meyer
Nachlaufwinkel in Grad (Verlauf und tabellarische Darstellung in K0)	24.03.2022	M.Meyer
Ackermannwinkel in Grad (Verlauf)	24.03.2022	M.Meyer
Lenkübersetzung (Verlauf)	24.03.2022	M.Meyer
Spreizungswinkel in Grad (tabellarische Darstellung in K0)	24.03.2022	M.Meyer
Ackermannfehler in Grad (tabellarische Darstellung vom max. Wert)	24.03.2022	M.Meyer
Spurkreisradius in Millimetern (tabellarische Darstellung vom min. Wert)	24.03.2022	M.Meyer
Verlauf über Radlenkwinkel Grad (X-Achse)	24.03.2022	M.Meyer
Elastokinematik:		
Vorspur in Gradminuten ohne/mit Einfluss von jeweiliger Längs- bzw. Querkraft (Verlauf und tabellarische Darstellung in K0)	24.03.2022	M.Meyer
Sturz in Grad ohne/mit Einfluss von jeweiliger Längs- bzw. Querkraft (Verlauf und tabellarische Darstellung in K0)	24.03.2022	M.Meyer
Radaufstandskraft in Newton (tabellarsiche Darstellung in K0)	24.03.2022	M.Meyer
Radfederrate in Newton/Millimeterquadrat (tabellarische Darstellung in K0)	24.03.2022	M.Meyer
Längssteifigkeit in Newton/Millimeterquadrat (tabellarische Darstellung in K0)	24.03.2022	M.Meyer
Quersteifigkeit in Newton/Millimeterquadrat (tabellarische Darstellung in K0)	24.03.2022	M.Meyer
Radlastkurve mit/ohne Stabilisator (Verlauf)	24.03.2022	M.Meyer
Verlauf über Radhub in Millimetern (Y-Achse)	24.03.2022	M.Meyer

Anlage 2

Suspension Analysis

Project Report

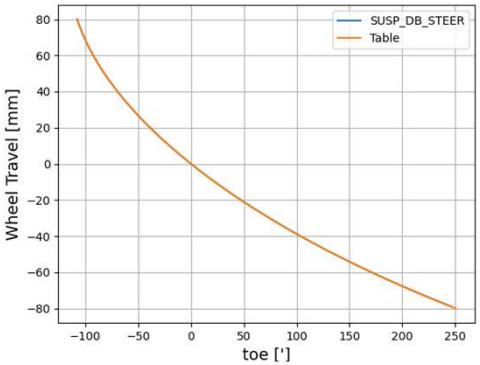
Suspension Analysis

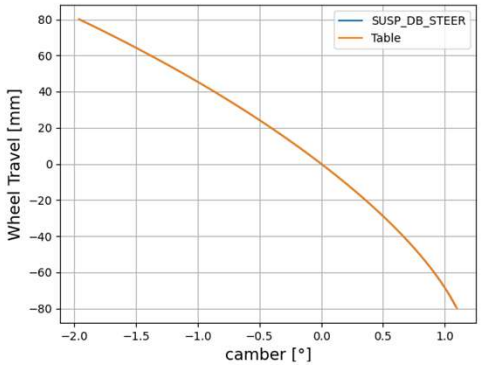
Overview VA Kinematic

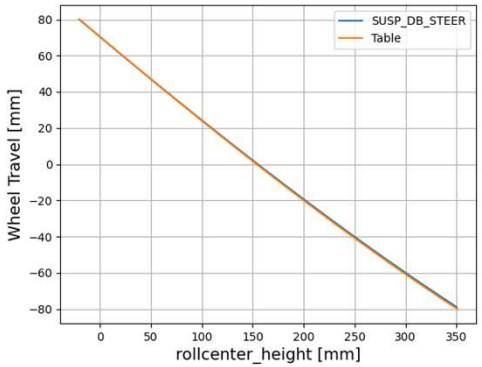
Designation	Unit	SUSP_DB_STEER	Table
Toe	[']	0	0
Camber	[°]	-0	0
Caster angle	[°]	5.38	5.38
Caster arm	[mm]	9.41	9.41
Spring ratio	[-]	0.516	0.516
Damper ratio	[-]	0.516	0.516
Rollcenter height	[mm]	155	154
Anti dive angle	[°]	0.43	0.43

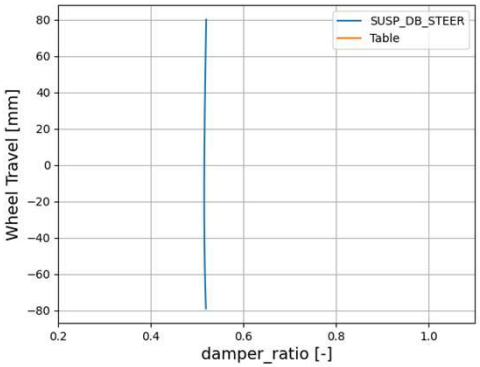
Suspension Analysis

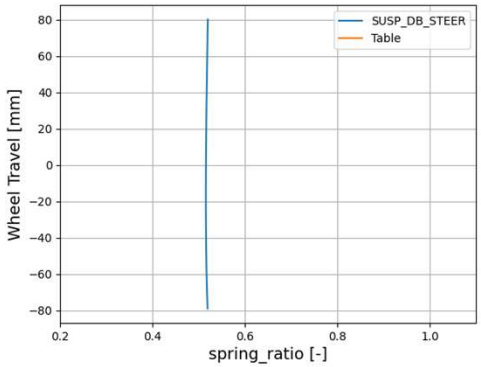
Overview VA Kinematic











Editor: Max Meyer (mea)

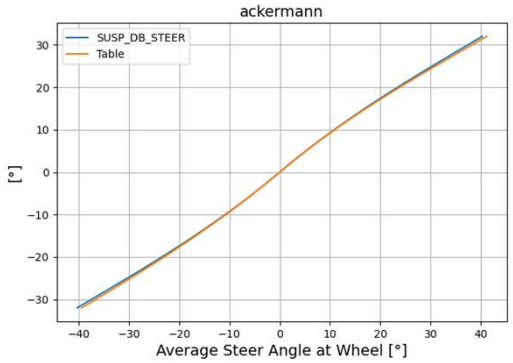
Suspension Analysis

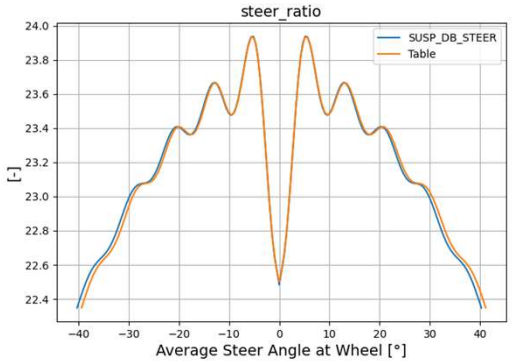
Overview VA Kinematic Steering

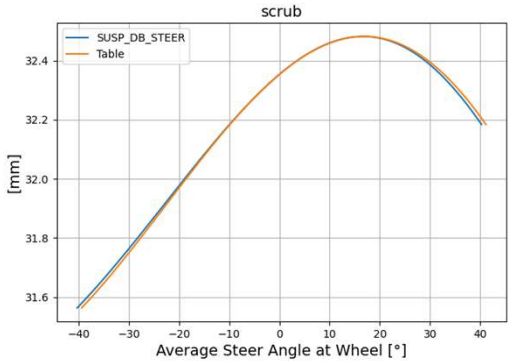
Designation	Unit	SUSP_DB_STEER	Table
Ackermann error	[°]	14.45	14.44
Caster arm	[mm]	9.88	10.15
Caster angle	[°]	5.36	5.38
Scrub	[mm]	32.35	32.35
Steer ratio	[-]	22.57	22.59
Kingpin inclination angle	[°]	10.01	10.01
Outside turn diameter	[mm]	8901	8900

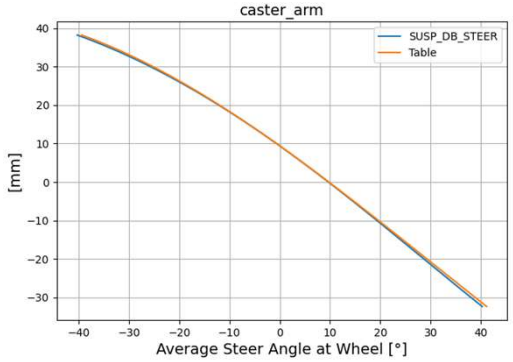
Suspension Analysis

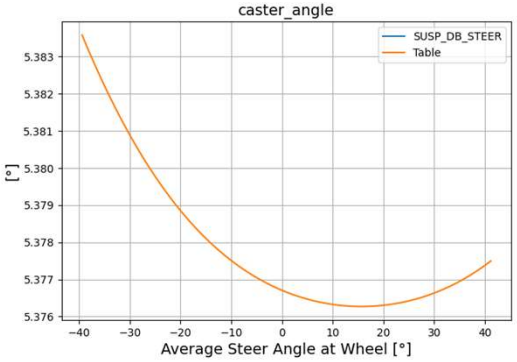
Overview VA Kinematic Steering











Editor: Max Meyer (mea)

Suspension Analysis

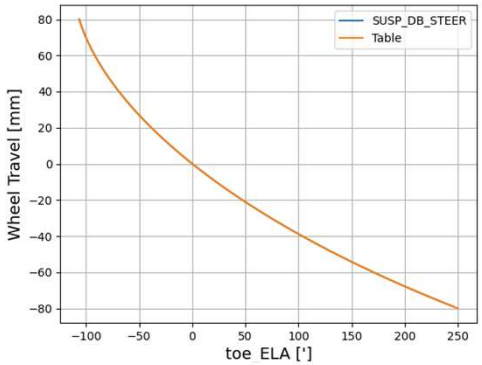
IAMT ENGINEERING

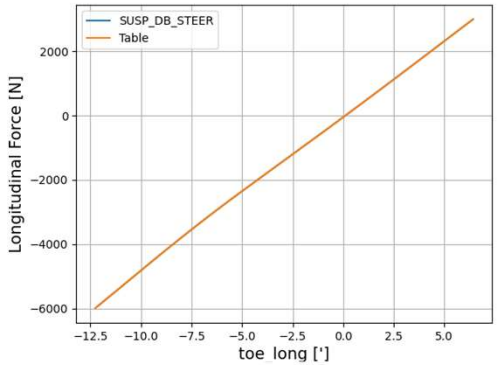
Overview VA Elastokinematic

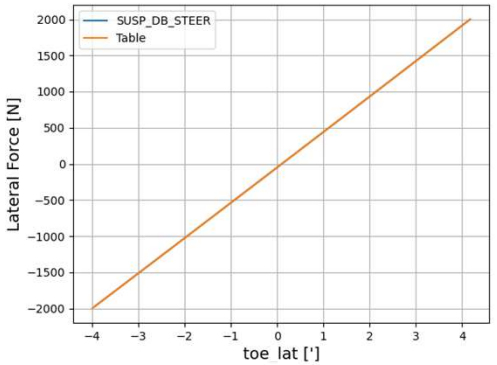
Designation	Unit	SUSP_DB_STEER	Table
Camber	[°]	-0	0
Lateral stiffness at CP	[N/mm]	42520	42516
Longitudinal stiffness at CP	[N/mm]	3321	3321
Toe	[']	0	0
Vertical Stiffness at CP	[N/mm]	26	26
Wheel load	[N]	1650	1650

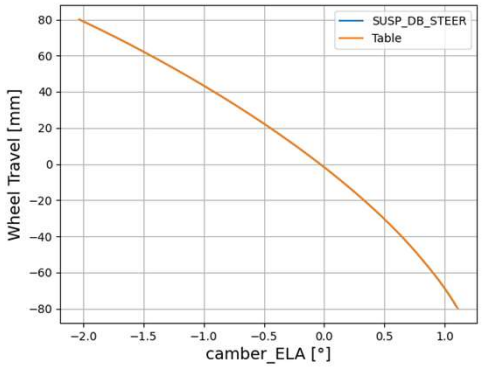
Suspension Analysis

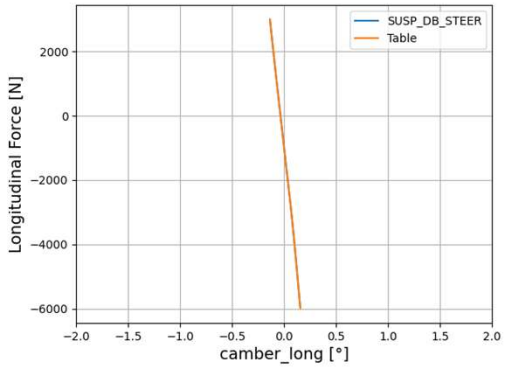
Overview VA Elastokinematic

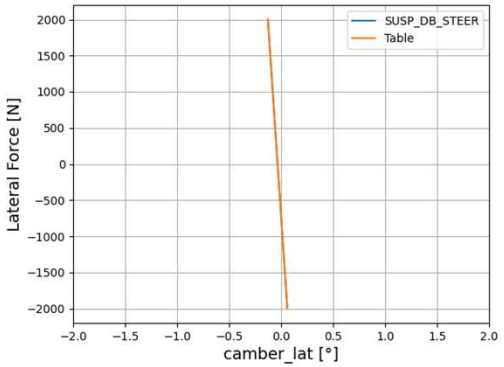








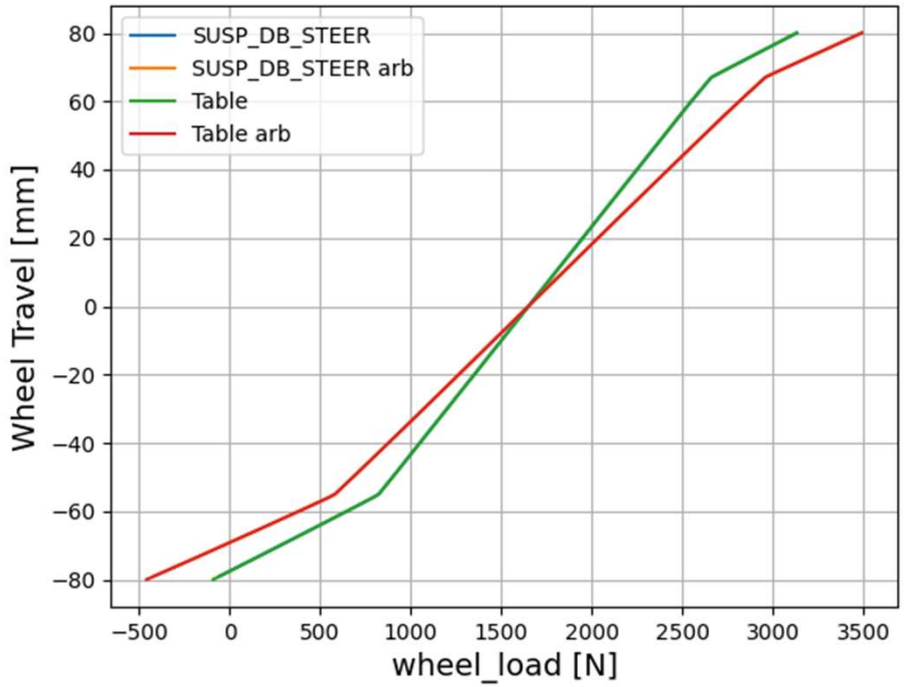




Editor: Max Meyer (mea)

Suspension Analysis

Overview VA Elastokinematic



Editor: Max Meyer (mea)

Suspension Analysis

Anlage 3

Suspension Analysis

Project Report

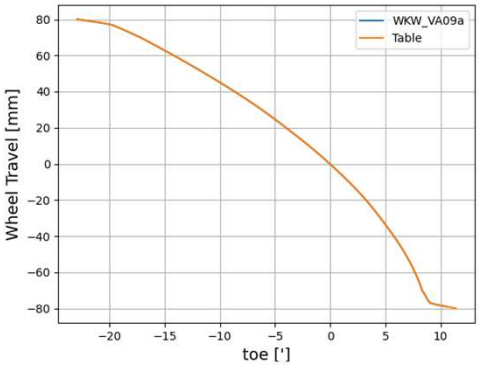
Suspension Analysis

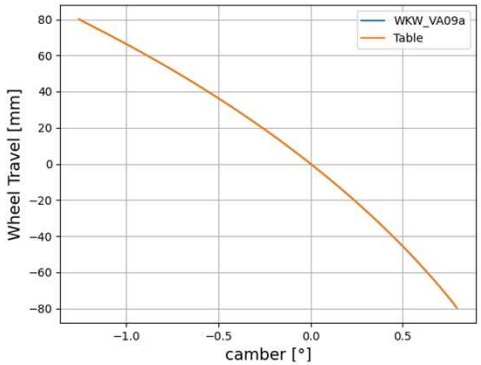
Overview VA Kinematic

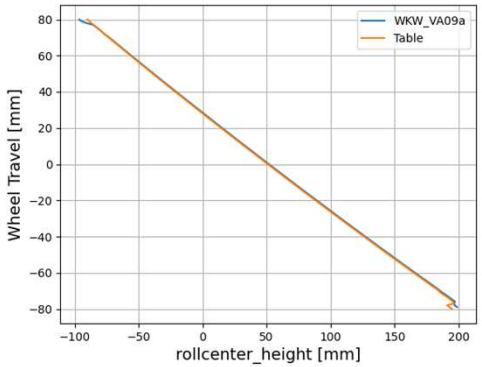
Designation	Unit	WKW_VA09a	Table
Toe	[']	-0	0
Camber	[°]	0	0
Caster angle	[°]	4.53	4.75
Caster arm	[mm]	32.11	34.39
Spring ratio	[-]	0.72	0.719
Damper ratio	[-]	0.72	0.719
Rollcenter height	[mm]	52	51
Anti dive angle	[°]	10.78	10.78

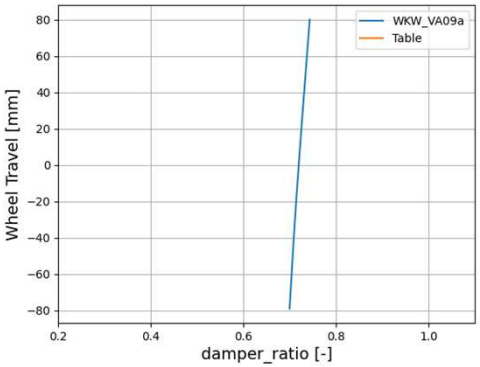
Suspension Analysis

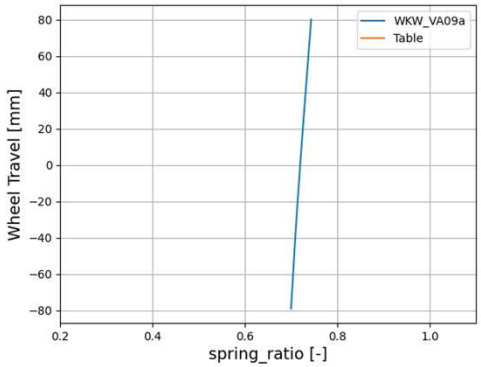
Overview VA Kinematic











Editor: Max Meyer (mea)

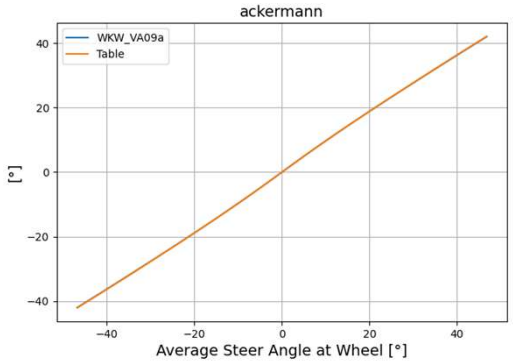
Suspension Analysis

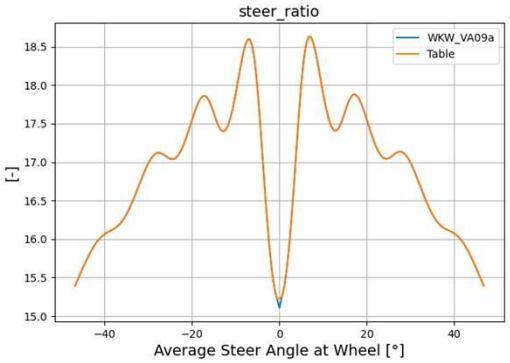
Overview VA Kinematic Steering

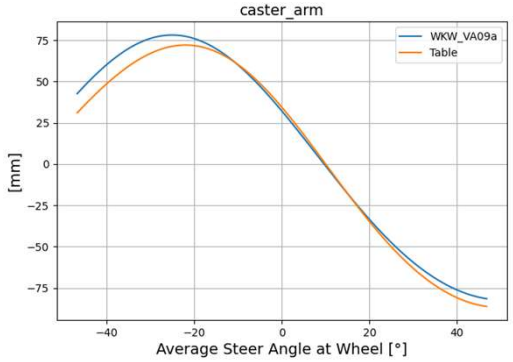
Designation	Unit	WKW_VA09a	Table
Ackermann error	[°]	8.02	7.94
Caster arm	[mm]	33.97	34.39
Caster angle	[°]	4.63	4.75
Scrub	[mm]	-12.26	-5.17
Steer ratio	[-]	15.23	15.23
Kingpin inclination angle	[°]	5.49	5.23
Outside turn diameter	[mm]	10123	10103

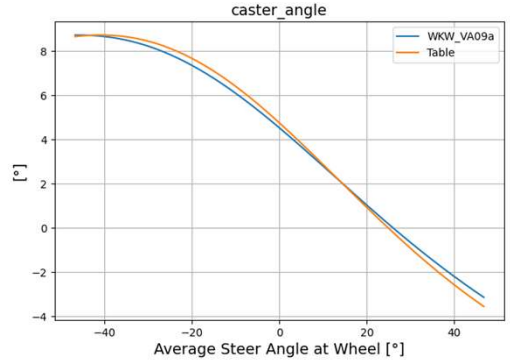
Suspension Analysis

Overview VA Kinematic Steering









Editor: Max Meyer (mea)

Suspension Analysis

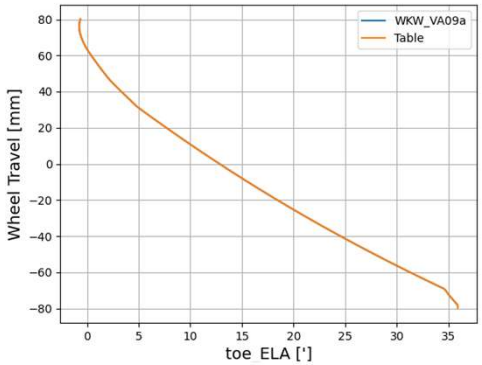
IAMT ENGINEERING

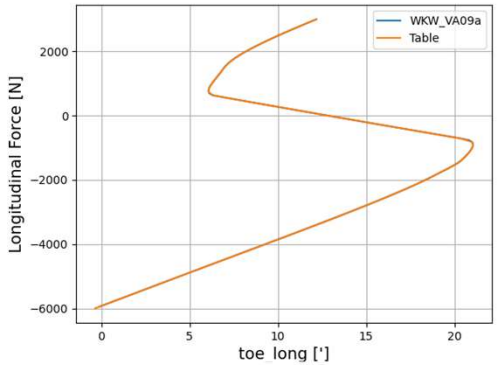
Overview VA Elastokinematic

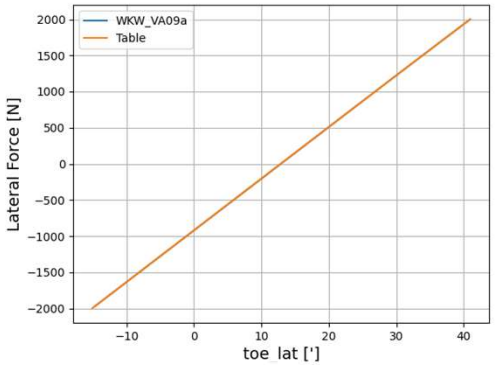
Designation	Unit	WKW_VA09a	Table
Camber	[°]	-0	-0.08
Lateral stiffness	[N/mm]	5316	5316
Longitudinal stiffness	[N/mm]	665	665
Toe	[']	13	13
Vertical stiffness	[N/mm]	108	108
Wheel load	[N]	5889	5889

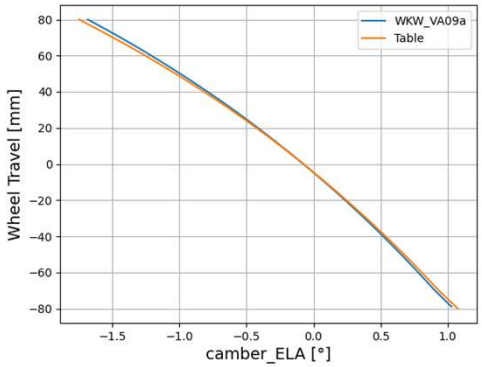
Suspension Analysis

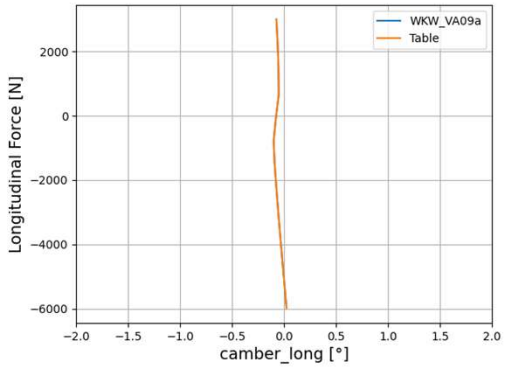
Overview VA Elastokinematic

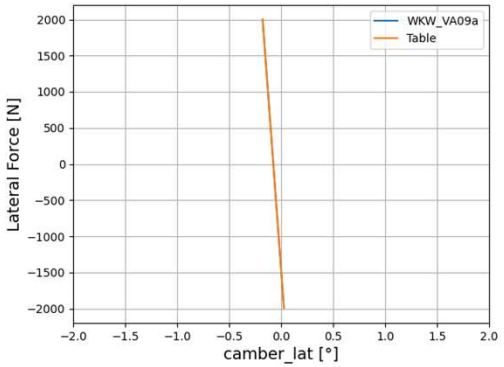








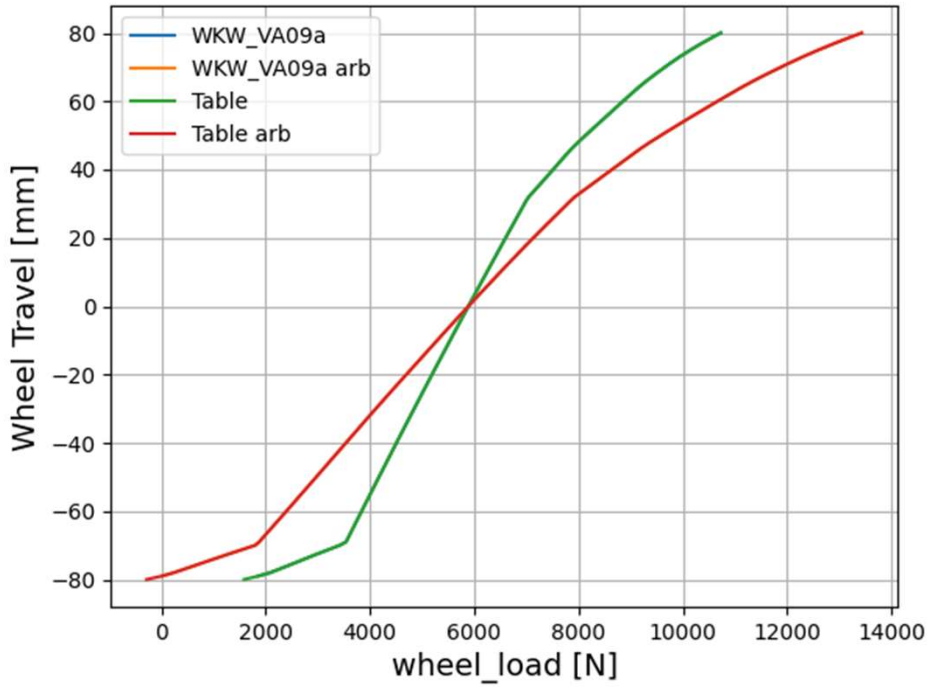




Editor: Max Meyer (mea)

Suspension Analysis

Overview VA Elastokinematic



Editor: Max Meyer (mea)

Suspension Analysis