Development of the software for a Formula Student vehicle suspension optimal hardpoints positions determination

Mraz, Bruno

Master's thesis / Diplomski rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:414305

UNIVERSITY OF ZAGREB
FACULTY OF MECHANICAL ENGINEERING AND NAVAL ARCHITECTURE

MASTER THESIS

Bruno Mraz

Zagreb, 2021.

UNIVERSITY OF ZAGREB
FACULTY OF MECHANICAL ENGINEERING AND NAVAL ARCHITECTURE

MASTER THESIS

Mentor: Student:

Izv. prof. dr. sc. Pero Prebeg Bruno Mraz

Zagreb, 2021.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se mentoru prof. Peri Prebegu na pomoći pri izradi diplomskog rada. Također se zahvaljujem svojoj obitelji (maloj grupi velikih ljudi) što su mi bili podrška te svima ostalima koji su mi pomogli kada je trebalo.

Bruno Mraz (Konrad)

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
Procesno-energetski, konstrukcijski, inženjersko modeliranje i računalne simulacije i brodostrojarski

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 21 - 6 / 1**	
Ur.broj: **15 - 1703 - 21 -**	

DIPLOMSKI ZADATAK

Student: **Bruno Mraz** JMBAG: 0035203993

Naslov rada na hrvatskom jeziku: **Izrada softvera za određivanje optimalnih pozicija točaka ovjesa formula student bolida**

Naslov rada na engleskom jeziku: **Development of the software for a formula student vehicle suspension optimal hardpoints positions determination**

Description of the task:

Design of a formula student vehicle suspension starts with determining wheel kinematics. Usually, it is done using specialized programs, however that is still a long and iterative process often subject to changes due to chassis and suspension dependency. For that reason, there is a need to automate the process of finding suspension hardpoints that satisfy all of suspension kinematic requests. The task of this thesis is to develop a software suitable for determining optimal suspension hardpoints positions, while satisfying all geometric and kinematic constraints.

The task is composed of the following:

- Studying literature on modelling formula student suspension.
- Development of a mathematical model of a quarter suspension model that will define suspension hardpoint positions and kinematic characteristics during wheel movement.
- Development of a graphical user interface which enables change of suspension parameters and presentation of calculated characteristics of suspension kinematics.
- Enabling visualization of suspension for verifying the position of wishbones, tie rod, upright and wheel.
- Formulation of single-objective optimisation problem for determining the formula student vehicle suspension hardpoints that must satisfy all geometric and kinematic constraints.
- Implementation of suspension optimization on the example of a formula student vehicle using developed mathematical model.

The thesis should list the used references and any assistance received.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
30. rujna 2021.	2. prosinca 2021.	13. – 17. prosinca 2021.

Zadatak zadao:

Izv. prof.dr.sc. Pero Prebeg

Predsjednik Povjerenstva:

Prof. dr. sc. Tanja Jurčević Lulić

CONTENTS

List of Figures

List of Tables

List of Notations

Notation	Unit	Description
A, B, C, D	1	Plane parameters for determining wheel vertical movement in global coordinate system
a, b, c, d	1	Plane parameters for determining wheel vertical movement in local coordinate system, ground plane vectors, intersection line parameters
a_{IC}, b_{IC}	1	Intersection line parameters for calculating roll centre height and anti-features
$a_{LCAintrs}$, $b_{LCAintrs}$	1	Lower control arm intersection line parameters for calculating roll centre height
$a_{UCAintrs}$, $b_{UCAintrs}$	1	Upper control arm intersection line parameters for calculating roll centre height
CP, CP_x, CP_y, CP_z	mm	Contact patch point and coordinates
h	mm	Centre of gravity height
IC, IC_x, IC_y, IC_z	mm	Intersection point and coordinates for calculating roll centre height and anti-features
$INCREMENT$	1	Number of increments between upmost or downmost and reference wheel position
l	mm	wheelbase
LCA_1, LCA_{1x}, LCA_{1y}, LCA_{1z}	mm	Lower control arm first point and its coordinates
LCA_{12}, LCA_{12x}, LCA_{12y}, LCA_{12z}	mm	Point on lower control arm rotation axis and its coordinates
LCA_2, LCA_{2x}, LCA_{2y}, LCA_{2z}	mm	Lower control arm second point and its coordinates
LCA_3, LCA_{3x}, LCA_{3y}, LCA_{3z}	mm	Lower control arm third point and its coordinates
$LCA_{3locLCA}$, $x_{LCA3locLCA}$, $y_{LCA3locLCA}$, $z_{LCA3locLCA}$	mm	Lower control arm third point and its coordinates in lower control arm local coordinate system
$LCA_{3locUCA}$, $x_{LCA3locUCA}$, $y_{LCA3locUCA}$, $z_{LCA3locUCA}$	mm	Lower control arm third point and coordinates in upper control arm local coordinate system

$LCA3UCA3_{pr}$	mm	Line connecting lower and upper control arm outer points projected to front plane
LCA_{intdir}	mm	Slope of lower control arm intersection line for determining anti features
LCA_{intPt}	mm	Point on lower control arm intersection line for determining anti features
n	1	Number of members in objective function
$PRECISION$	1	Precision parameter for calculations in code
PT, PT_x, PT_y, PT_z	mm	Arbitrary point and its coordinates
RC, RC_x, RC_y, RC_z	mm	Roll centre point and its coordinates
r_{CA}	mm	Distance between lower and upper control arm third points
r_{LCA}	mm	Radius of lower control arm
R_{LCA}	1	Rotation matrix from lower control arm local coordinate system to global
r_{ST}	mm	Steering arm length
R_{TR}	1	Rotation matrix from tie rod local coordinate system to global
r_{TR}	mm	Tie rod length
r_{UCA}	mm	Radius of upper control arm
R_{UCA}	1	Rotation matrix from upper control arm local coordinate system to global
s	1	Significance value
SPN, $SPNx$, $SPNy$, $SPNz$	mm	Wheel spindle point and its coordinates
SPN_{locTR}	mm	Wheel spindle point in tie rod local coordinate system
SPN_{pr}, SPN_{prx}, SPN_{pry}, SPN_{prz}	mm	SPN projection to ground plane and its coordinates
t	mm	Constant determining tie rod outer point position on line connecting control arms outer points
t_{LCA}	mm	Offset vector between lower control arm local coordinate system and global coordinate system origin

TR_1, TR_{1x}, TR_{1y}, TR_{1z}	mm	Tie rod inner point and its coordinates
TR_{1locTR}, $x_{TR1locTR}$, $y_{TR1locTR}$, $z_{TR1locTR}$	mm	TR1 point and coordinates in TR local coordinate system
TR_2, TR_{2x}, TR_{2y}, TR_{2z}	mm	Tie rod outer point and its coordinates
TR_{2locTR}, $x_{TR2locTR}$, $y_{TR2locTR}$, $z_{TR2locTR}$	mm	TR2 point and coordinates in TR local coordinate system
TR_{pr2}, TR_{pr2x}, TR_{pr2y}, TR_{pr2z}	mm	TR2 projection to line connecting LCA3 and UCA3
t_{TR}	mm	offset vector between TR local coordinate system and global coordinate system origin
t_{UCA}	mm	Offset vector between UCA local coordinate system and global coordinate system origin
UCA_1, UCA_{1x}, UCA_{1y}, UCA_{1z}	mm	Upper control arm first point and its coordinates
UCA_{12}, UCA_{12x}, LCA_{12y}, UCA_{12z}	mm	Point on upper control arm rotation axis and its coordinates
UCA_2, UCA_{2x}, UCA_{2y}, UCA_{2z}	mm	Upper control arm second point and its coordinates
UCA_3, UCA_{3x}, UCA_{3y}, UCA_{3z}	mm	Upper control arm third point and its coordinates
$UCA_{3locUCA}$, $x_{UCA3locUCA}$, $y_{UCA3locUCA}$, $z_{UCA3locUCA}$	mm	Upper control arm third point and its coordinates in upper control arm local coordinate system
UCA_{intdir}	mm	Slope of upper control arm intersection line for determining anti features
UCA_{intPt}	mm	Point on upper control intersection line for determining anti features
v_0	mm	Arbitrary vector for calculating contact patch
w	1	Weight factor
WCN, WCN_x, WCN_y, WCN_z	mm	Wheel centre point and its coordinates
WCN_{locTR}	mm	Wheel centre point in tie rod local coordinate system and its coordinates
WCN_{pr}, WCN_{prx}, WCN_{pry}, WCN_{prz}	mm	WCN projection to ground plane and its coordinates
$WRADIUS$	mm	Wheel radius

$WSTEER$	mm	Steering rack movement
$WVERT$	mm	Vertical movement of lower control arm third point
x_{LCA}, x_{LCAx}, x_{LCAy}, x_{LCAz}	mm	Vector and coordinates describing x axis of lower control arm local coordinate system
x_{UCA}, x_{UCAx}, x_{UCAy}, x_{UCAz}	mm	Vector and coordinates describing x axis of upper control arm local coordinate system
x_{UCTR}, x_{TRx}, x_{TRy}, x_{TRz}	mm	Vector and coordinates describing x axis of tie rod local coordinate system
y_{IC}, z_{IC}	mm	Intersection line coordinates for calculating roll centre height and anti features
y_{LCA}, y_{LCAx}, y_{LCAy}, y_{LCAz}	mm	Vector and coordinates describing y axis of lower control arm local coordinate system
$y_{LCAintersection}$, $z_{LCAintersection}$	mm	Lower control arm intersection coordinates for calculating roll centre height
y_{UCA}, y_{UCAx}, y_{UCAy}, y_{UCAz}	mm	Vector and coordinates describing y axis of upper control arm local coordinate system
$y_{UCAintersection}$, $z_{UCAintersection}$	mm	Upper control arm intersection coordinates for calculating roll centre height
y_{UCTR}, y_{TRx}, y_{TRy}, y_{TRz}	mm	Vector and coordinates describing y axis of tie rod local coordinate system
z_{LCA}, z_{LCAx}, z_{LCAy}, z_{LCAz}	mm	Vector and coordinates describing z axis of lower control arm local coordinate system
$z_{LCA3max}$	mm	Maximum z value of lower control arm third point
$z_{LCA3min}$	mm	Minimum z value of lower control arm third point
z_{TR}, z_{TRx}, z_{TRy}, z_{TRz}	mm	Vector and coordinates describing z axis of tie rod local coordinate system
z_{UCA}, z_{UCAx}, z_{UCAy}, z_{UCAz}	mm	Vector and coordinates describing z axis of upper control arm local coordinate system
$\theta_{inboard}$	rad	Angle for calculating anti features in case of inboard components
$\theta_{outboard}$	rad	Angle for calculating anti features in case of outboard components

SAŽETAK

Cilj ovog diplomskog rada je bio napraviti aplikaciju koja automatizira proces traženja željene kinematike ovjesa dvostrukih ramena uz zadovoljenje ograničenja na karakteristike ovjesa za potrebe Formula Student timova, a naročito FSB Racing Team-a. To bi značilo da studenti mogu provesti manje vremena na zadatke koji ne donose nova znanja i više na konstruiranje.

Pretragom interneta mogu se pronaći komercijalni i besplatni kalkulatori ovjesa sa ili bez otvorenog koda. Pritom komercijalni koriste 3D model opisa ovjesa dok su za besplatne kalkulatore korištene ili 2D aproksimacije ili 3D modeli koji se i sami rješavaju uz pomoć optimizacijskog algoritma. Međutim, niti jedni niti drugi ne prikazuju matematičke modele kako hoda tako niti značajki ovjesa što je u ovome radu razvijeno za 3D model četvrtine ovjesa te se izračunava direktno, bez potrebe za iteracijama ili optimizacijskim algoritmima. Nadalje, trenutno ne postoji besplatni kalkulator ovjesa koji ima razvijen optimizacijski modul za kinematiku ovjesa, te samo neki komercijalni sadrže tu mogućnost ili u beta fazi testiranja (OptimumG) ili uz potrebu za postavljanjem optimizacijskog modela iz nule (MSC Adams Car Module). Stoga je ovo prvi javno dostupan alat, prema autorovom istraživanju dostupnih rješenja, otvorenog koda sa popratnim objašnjenjima koji implementira optimizaciju kinematike ovjesa.

Uspjeh besplatnih aplikacija uvelike određuje i njihova jednostavnost korištenja. Stoga je razvijeno grafičko korisničko sučelje sa: jednostavnom mogućnošću promjena parametara trenutnog ovjesa, interaktivnom vizualizacijom trenutnog 3D modela četvrtine ovjesa, prikazom glavnih značajki trenutnog modela ovjesa i mogućnošću upisivanja parametara za optimizaciju kinematike ovjesa.

Konačno, kako je širenje znanja u duhu Formula Student zajednice, izvorni kod dostupan je na slijedećem GitHub repozitoriju:

https://github.com/brunomraz/FS-BMK

Ključne riječi: Formula Student, FSB Racing Team, ovjes, optimizacija, grafičko korisničko sučelje, C++, C#, Python, programiranje

SUMMARY

The main objective of this Master Thesis is to create an application that would automate the process of finding preferred double wishbone suspension kinematics while satisfying the constraints on suspension characteristics. This would in turn mean that students can spend less time on tasks which do not bring any new knowledge and more on optimizing the rest of suspension subsystem.

Through internet search commercial and free suspension calculators with or without open-source code can be found. Commercial are characterised by use of a 3D suspension model while free calculators use 2D models with approximations or 3D models that are solved with the help of optimisation algorithms. However, none of them give insight to the mathematical models used for calculating suspension movement or characteristics what has been done in this master thesis for the 3D quarter suspension model and it is calculated directly, without iterations or use of the optimisation algorithms. Furthermore, currently there is no free suspension calculator that has implemented a module for suspension kinematics optimisation and the commercial ones either have this ability in beta stages (OptimumG) or complete optimisation model must be made from scratch (MSC Adams Car Module). Therefore, this is the first freely available tool, to the author's best knowledge, with open-source code and accompanying explanations that implements suspension kinematics optimisation.

Finally, success of free application largely depends on their ease of use, so a graphical user interface is developed. It is characterised by simple parameter modification of the current suspension, interactive visualization of the current 3D quarter suspension model, output of main suspension characteristics of the current suspension and with the ability to enter parameters for suspension kinematics optimisation.

Finally, since sharing knowledge is common practice in Formula Student community, the source code is available at GitHub repository:

https://github.com/brunomraz/FS-BMK

Key words: Formula Student, FSB Racing Team, suspension, optimization, graphical user interface, C++, C#, Python, programming

1. INTRODUCTION

Design of Formula Student (FS) vehicle suspension systems is a complex and tedious work, which the author has familiarised with through his participation in FSB Racing Team. This is true even more because people doing it are students that often lack experience and knowledge to do it efficiently. Plenty of cooperation is also needed with aerodynamics, chassis, powertrain, and battery systems what doesn't necessarily add to complexion but prolongs the development. This can be seen with each change of any of the before mentioned systems. Changes usually imply new collisions or unacceptable component positions because of FS rules which in the end means that a person will have to spend additional couple of hours by computer and finetune suspension hardpoints until they once again produce suspension characteristics within reasonable limits.

Currently this is done with the help of Lotus Suspension Analyser (LSA) and CAD software, in case of FSB Racing Team, SolidWorks is used. There is constant copying and pasting of hardpoints between these two to ensure the best possible results. This is generally a good approach, however it can be improved. LSA can give us suspension characteristics only for individual procedures; bump, steering and rolling, there is no possibility to do a combination of steering and rolling. This is a problem because we must ensure that the outer wheel does not acquire positive Camber during turning since this would render catastrophe on vehicle performance. To avoid this currently we import hardpoints from LSA to SW and then check in SW how the wheel behaves when steered during roll. If the results are not as expected, we return to LSA and repeat the process.

Before any work was done, a study of available tools was made. The tools are divided in two categories, commercial and free software. From commercial software Lotus Suspension Analyser, OptimumG and MSC Adams stand out. All three of those have many more features than are planned to be made in the scope of this master thesis. Those features are mostly related to tyre model, forces in suspension components, dampers, and coil springs, all of which comes later in suspension design. But common point between this master thesis and mentioned tools is 3D model used for calculating suspension characteristics. Freely available software is mostly based on approximations, meaning that a 2D model is used which is considered unfit because of all the characteristics that will be calculated for this master thesis with examples being [6] and [7]. The only reference that is like this master thesis' way of calculating suspension

movement is found in [8], and it only describes from the highest conceptual level that intersections between geometrical objects should be found to calculate suspension movement.

While searching for suspension characteristics mathematical formulation of a 3D suspension model, none were found. The results that were found are common graphics that depict calculation in case of 2D suspension. So, it was decided to create both the mathematical model of suspension movement and suspension characteristics from scratch.

To programmatically describe workflow in development of application, this master thesis is divided to several chapters. First the mathematical model describing suspension kinematics is developed followed by mathematical description of suspension characteristics. With the mathematical model of a quarter suspension model fully defined it is possible to move on to optimisation chapter. In it are explained constraints, objective function, and project space. With the theoretical background done, application overview is given. In short it is described how are mathematical models implemented, which tools and which programming languages were used. how the application optimisation works. Finally, an example of suspension optimisation is given.

This application, named Formula Student Basic Motions and Kinematics (FS-BMK), is intended to be free for everyone to use and modify so it would help in future work of Formula Student teams.

Link to GitHub repository: https://github.com/brunomraz/FS-BMK

Due to future updates application described in this thesis may differ from the one found at the GitHub repository.

2. Suspension Quarter Model- Mathematical Model

Standard coordinate system from the SAE convention [1] will be used for modelling of suspension quarter model as can be seen in the following figure. Only the left side will be modelled, and the right will follow based on symmetricity. The position of origin will be arbitrary because it won't affect the suspension kinematics, it is only important to note that front suspension will have greater X values than rear suspension.

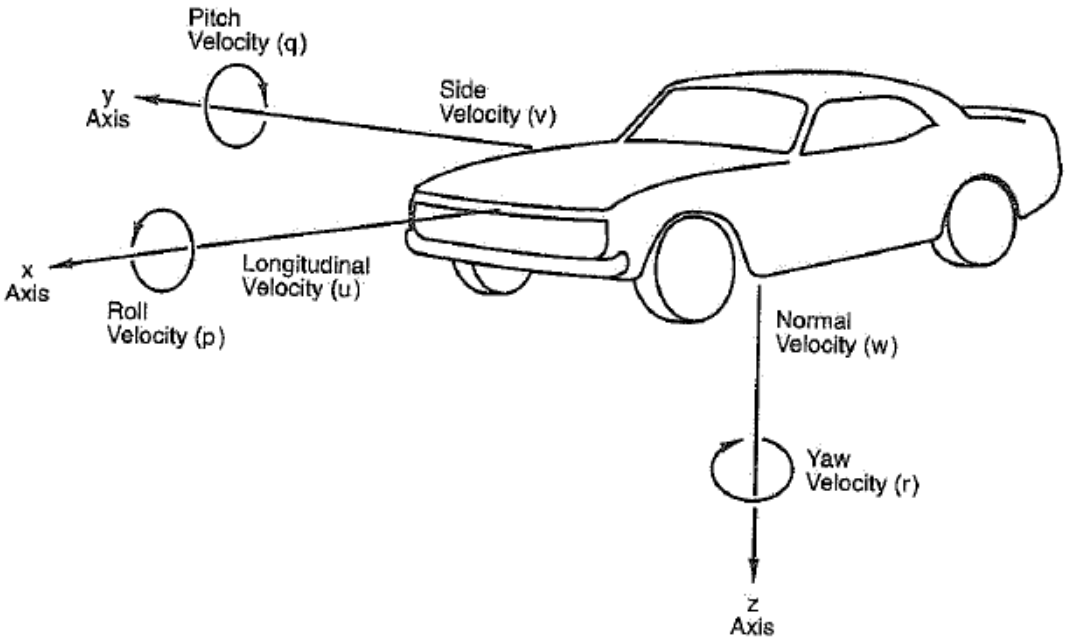

Figure 1 Vehicle coordinate system [1]

Most of the teams in FSAE use double wishbone type suspension and so will the described suspension. It consists of 10 characteristic points:

1. Upper control arm- front point– UCA_1
2. Upper control arm- rear point – UCA_2
3. Upper control arm- outer point – UCA_3
4. Lower control arm- front point – LCA_1
5. Lower control arm- rear point – LCA_2
6. Lower control arm- outer point – LCA_3
7. Tie rod- inner point – TR_1
8. Tie rod- outer point – TR_2
9. Wheel centre point – *WCN*
10. Spindle point – *SPN*

There are also 2 points that follow from dimensional reduction; UCA_{12} which is a projection of UCA_3 onto line formed by points UCA_1 and UCA_2. Same applies to LCA_{12}. Contact patch point (*CP*) represents the place where the wheel is theoretically in contact with ground. This point is obtained by constructing a line perpendicular to both the wheel rotation axis and an arbitrary line that is perpendicular to wheel rotation axis and parallel to ground. Now that all relevant points are established follows definition of relations between points during movement.

All characteristic points have a spherical joint in their place. However, inner points of control arms result in a hinge joint. Out of 6 variables (xyz coordinates for 2 points) that describe these 2 spherical joints, only 4 of them have effect on the hinge joint, for the other 2 variables we can choose arbitrary values which will not affect calculations. The rest of the points employ spherical joints.

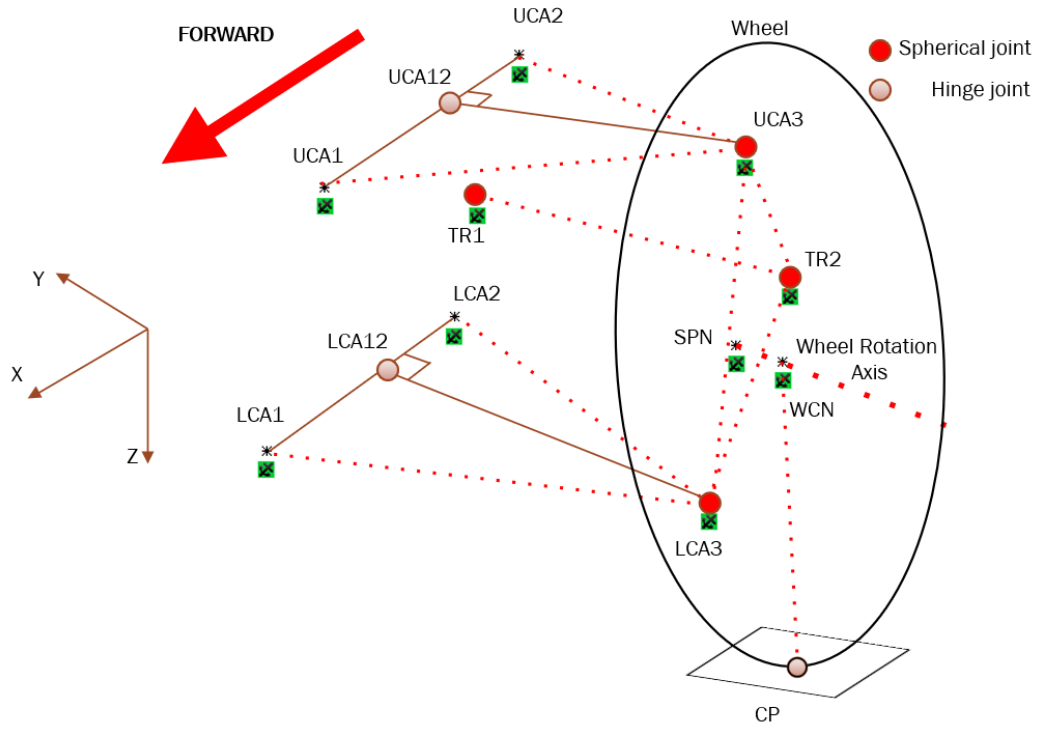

Figure 2 Double wishbone suspension schematic representation

2.1. Development of Kinematic Equations for Quarter Suspension

Suspension will have two degrees of freedom; up and down movement, and steering the wheel left and right. Steering of the wheel is achieved via tie rod, while up and down movement can be initiated either by upper or by lower control arm. Here it is decided to use the lower control arm as the starting point for moving the suspension mechanism because in all previous FSB Racing Team's vehicles the pushrod system was attached to lower control arm. So, in case of further development of this suspension model and implementation of quasi static equations, it

will be easier to transfer forces from LCA to pushrod subsystem. Principle on which the position of the wheel will be calculated is shown in the following figure.

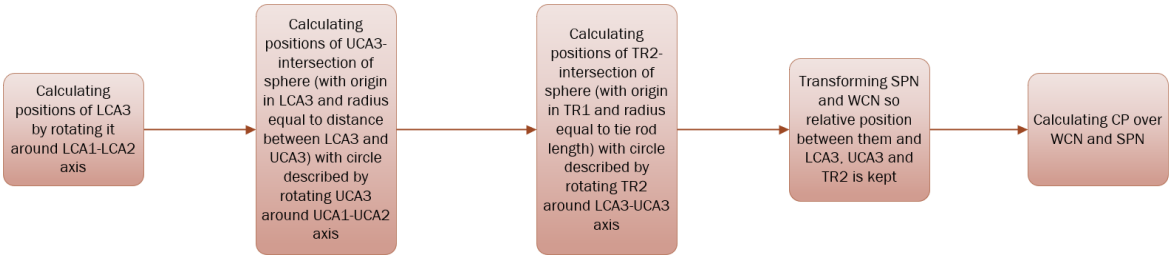

Figure 3 Calculation of suspension points during movement

After calculating positions of suspension, characteristics such as Camber angle, Mechanical trail, anti-features, etc., are calculated for each of the positions.

Following table sums up data needed to start the calculation. Since the calculation cannot be exact because some later formulas become too complex, it is needed to approximate results. Therefore, the precision parameter is needed but only for some suspension characteristics.

Table 1 Parameters for Calculating Suspension Movement

Lower Control Arm- Front	LCA_{1x}	LCA_{1y}	LCA_{1z}
Lower Control Arm- Rear	LCA_{2x}	LCA_{2y}	LCA_{2z}
Lower Control Arm- Outer	LCA_{3x}	LCA_{3y}	LCA_{3z}
Upper Control Arm- Front	UCA_{1x}	UCA_{1y}	UCA_{1z}
Upper Control Arm- Rear	UCA_{2x}	UCA_{2y}	UCA_{2z}
Upper Control Arm- Outer	UCA_{3x}	UCA_{3y}	UCA_{3z}
Tie Rod- Inner	TR_{1x}	TR_{1y}	TR_{1z}
Tie Rod- Outer	TR_{2x}	TR_{2y}	TR_{2z}
Wheel Centre	WCN_x	WCN_y	WCN_z
Spindle Point	SPN_x	SPN_y	SPN_z
Wheel Radius	*WRADIUS*		
Wheel Up Movement	*WVERT*		
Tie Rod Inner- Movement Left and Right	*WSTEER*		
Precision	*PRECISION*		

Incrementation	*INCREMENT*

As will be seen in the following chapters, kinematic relations easily become very complex. To avoid the possibility of making mistakes Sympy package in Python was used for deriving needed relations.

In all calculations right-handed Cartesian coordinate system is used.

2.1.1. Calculating Suspension Kinematic Constants

Calculation is started determining LCA local coordinate system. The needed data are origin, and two perpendicular unit vectors defining orientation. First, LCA_3 is projected to LCA_1-LCA_2 line giving the LCA_{12} point representing the origin of LCA local coordinate system in global coordinate system.

Projection consists of doing a dot product between vector LCA_2-LCA_1 and vector LCA_{12}-$LCA3$. The dot product must equal zero as the two vectors are defining axes of LCA local coordinate system:

$$\left(\overrightarrow{LCA_2 LCA_1}\right)\cdot\left(\overrightarrow{LCA_{12} LCA_3}\right)=0. \tag{1}$$

After solving equation 1, the coordinates of LCA_{12} are obtained:

$$LCA_{12x}=\frac{LCA_{1x}\left(\left(LCA_{1x}-LCA_{2x}\right)^2+\left(LCA_{1y}-LCA_{2y}\right)^2+\left(LCA_{1z}-LCA_{2z}\right)^2\right)}{\left(LCA_{1x}-LCA_{2x}\right)^2+\left(LCA_{1y}-LCA_{2y}\right)^2+\left(LCA_{1z}-LCA_{2z}\right)^2} - \frac{\left(LCA_{1x}-LCA_{2x}\right)\left(\begin{array}{l}\left(LCA_{1x}-LCA_{2x}\right)\left(LCA_{1x}-LCA_{3x}\right)+\\ \left(LCA_{1y}-LCA_{2y}\right)\left(LCA_{1y}-LCA_{3y}\right)+\\ \left(LCA_{1z}-LCA_{2z}\right)\left(LCA_{1z}-LCA_{3z}\right)\end{array}\right)}{\left(LCA_{1x}-LCA_{2x}\right)^2+\left(LCA_{1y}-LCA_{2y}\right)^2+\left(LCA_{1z}-LCA_{2z}\right)^2}, \tag{2}$$

$$LCA_{12y}=\frac{LCA_{1y}\left(\left(LCA_{1x}-LCA_{2x}\right)^2+\left(LCA_{1y}-LCA_{2y}\right)^2+\left(LCA_{1z}-LCA_{2z}\right)^2\right)}{\left(LCA_{1x}-LCA_{2x}\right)^2+\left(LCA_{1y}-LCA_{2y}\right)^2+\left(LCA_{1z}-LCA_{2z}\right)^2} - \frac{\left(LCA_{1y}-LCA_{2y}\right)\left(\begin{array}{l}\left(LCA_{1x}-LCA_{2x}\right)\left(LCA_{1x}-LCA_{3x}\right)+\\ \left(LCA_{1y}-LCA_{2y}\right)\left(LCA_{1y}-LCA_{3y}\right)+\\ \left(LCA_{1z}-LCA_{2z}\right)\left(LCA_{1z}-LCA_{3z}\right)\end{array}\right)}{\left(LCA_{1x}-LCA_{2x}\right)^2+\left(LCA_{1y}-LCA_{2y}\right)^2+\left(LCA_{1z}-LCA_{2z}\right)^2}, \tag{3}$$

$$LCA_{12z} = \frac{LCA_{1z}\left(\left(LCA_{1x}-LCA_{2x}\right)^2+\left(LCA_{1y}-LCA_{2y}\right)^2+\left(LCA_{1z}-LCA_{2z}\right)^2\right)}{\left(LCA_{1x}-LCA_{2x}\right)^2+\left(LCA_{1y}-LCA_{2y}\right)^2+\left(LCA_{1z}-LCA_{2z}\right)^2} - \frac{\left(LCA_{1z}-LCA_{2z}\right)\begin{pmatrix}\left(LCA_{1x}-LCA_{2x}\right)\left(LCA_{1x}-LCA_{3x}\right)+\\ \left(LCA_{1y}-LCA_{2y}\right)\left(LCA_{1y}-LCA_{3y}\right)+\\ \left(LCA_{1z}-LCA_{2z}\right)\left(LCA_{1z}-LCA_{3z}\right)\end{pmatrix}}{\left(LCA_{1x}-LCA_{2x}\right)^2+\left(LCA_{1y}-LCA_{2y}\right)^2+\left(LCA_{1z}-LCA_{2z}\right)^2}. \quad \textbf{(4)}$$

Following is the LCA radius calculation which is obtained by calculating distance between LCA_3 and LCA_{12}:

$$r_{LCA} = \left|LCA_{12}-LCA_3\right|. \quad \textbf{(5)}$$

$$r_{LCA} = \sqrt{\left(-LCA_{12x}+LCA_{3x}\right)^2+\left(-LCA_{12y}+LCA_{3y}\right)^2+\left(-LCA_{12z}+LCA_{3z}\right)^2}. \quad \textbf{(6)}$$

Last thing left to calculate are perpendicular unit direction vectors. They will always be pointing from backward to forward (longitudinal) and from middle to side (lateral):

$$x_{LCA} = \left(\frac{LCA_{1x}-LCA_{2x}}{\left|LCA_1-LCA_2\right|}, \frac{LCA_{1y}-LCA_{2y}}{\left|LCA_1-LCA_2\right|}, \frac{LCA_{1z}-LCA_{2z}}{\left|LCA_1-LCA_2\right|}\right), \quad \textbf{(7)}$$

$$y_{LCA} = \left(\frac{LCA_{3x}-LCA_{12x}}{r_{LCA}}, \frac{LCA_{3y}-LCA_{12y}}{r_{LCA}}, \frac{LCA_{3z}-LCA_{12z}}{r_{LCA}}\right). \quad \textbf{(8)}$$

After expanding equations 7 and 8 the following forms are obtained:

$$x_{LCAx} = \frac{LCA_{1x}-LCA_{2x}}{\sqrt{\left(LCA_{1x}-LCA_{2x}\right)^2+\left(LCA_{1y}-LCA_{2y}\right)^2+\left(LCA_{1z}-LCA_{2z}\right)^2}}, \quad \textbf{(9)}$$

$$x_{LCAy} = \frac{LCA_{1y}-LCA_{2y}}{\sqrt{\left(LCA_{1x}-LCA_{2x}\right)^2+\left(LCA_{1y}-LCA_{2y}\right)^2+\left(LCA_{1z}-LCA_{2z}\right)^2}}, \quad \textbf{(10)}$$

$$x_{LCAz} = \frac{LCA_{1z}-LCA_{2z}}{\sqrt{\left(LCA_{1x}-LCA_{2x}\right)^2+\left(LCA_{1y}-LCA_{2y}\right)^2+\left(LCA_{1z}-LCA_{2z}\right)^2}}, \quad \textbf{(11)}$$

$$y_{LCAx} = \frac{-LCA_{12x}+LCA_{3x}}{\sqrt{\left(LCA_{12x}-LCA_{3x}\right)^2+\left(LCA_{12y}-LCA_{3y}\right)^2+\left(LCA_{12z}-LCA_{3z}\right)^2}}, \quad \textbf{(12)}$$

$$y_{LCAy} = \frac{-LCA_{12y}+LCA_{3y}}{\sqrt{\left(LCA_{12x}-LCA_{3x}\right)^2+\left(LCA_{12y}-LCA_{3y}\right)^2+\left(LCA_{12z}-LCA_{3z}\right)^2}}, \quad \textbf{(13)}$$

$$y_{LCAz} = \frac{-LCA_{12z}+LCA_{3z}}{\sqrt{\left(LCA_{12x}-LCA_{3x}\right)^2+\left(LCA_{12y}-LCA_{3y}\right)^2+\left(LCA_{12z}-LCA_{3z}\right)^2}}. \quad \textbf{(14)}$$

The same process applies to upper control arm and only the solutions are given:

$$UCA_{12x} = \frac{UCA_{1x}\left(\left(UCA_{1x}-UCA_{2x}\right)^2+\left(UCA_{1y}-UCA_{2y}\right)^2+\left(UCA_{1z}-UCA_{2z}\right)^2\right)}{\left(UCA_{1x}-UCA_{2x}\right)^2+\left(UCA_{1y}-UCA_{2y}\right)^2+\left(UCA_{1z}-UCA_{2z}\right)^2} - \frac{\left(UCA_{1x}-UCA_{2x}\right)\left(\begin{array}{l}\left(UCA_{1x}-UCA_{2x}\right)\left(UCA_{1x}-UCA_{3x}\right)+\\ \left(UCA_{1y}-UCA_{2y}\right)\left(UCA_{1y}-UCA_{3y}\right)+\\ \left(UCA_{1z}-UCA_{2z}\right)\left(UCA_{1z}-UCA_{3z}\right)\end{array}\right)}{\left(UCA_{1x}-UCA_{2x}\right)^2+\left(UCA_{1y}-UCA_{2y}\right)^2+\left(UCA_{1z}-UCA_{2z}\right)^2}, \quad \textbf{(15)}$$

$$UCA_{12y} = \frac{UCA_{1y}\left(\left(UCA_{1x}-UCA_{2x}\right)^2+\left(UCA_{1y}-UCA_{2y}\right)^2+\left(UCA_{1z}-UCA_{2z}\right)^2\right)}{\left(UCA_{1x}-UCA_{2x}\right)^2+\left(UCA_{1y}-UCA_{2y}\right)^2+\left(UCA_{1z}-UCA_{2z}\right)^2} - \frac{\left(UCA_{1y}-UCA_{2y}\right)\left(\begin{array}{l}\left(UCA_{1x}-UCA_{2x}\right)\left(UCA_{1x}-UCA_{3x}\right)+\\ \left(UCA_{1y}-UCA_{2y}\right)\left(UCA_{1y}-UCA_{3y}\right)+\\ \left(UCA_{1z}-UCA_{2z}\right)\left(UCA_{1z}-UCA_{3z}\right)\end{array}\right)}{\left(UCA_{1x}-UCA_{2x}\right)^2+\left(UCA_{1y}-UCA_{2y}\right)^2+\left(UCA_{1z}-UCA_{2z}\right)^2}, \quad \textbf{(16)}$$

$$UCA_{12z} = \frac{UCA_{1z}\left(\left(UCA_{1x}-UCA_{2x}\right)^2+\left(UCA_{1y}-UCA_{2y}\right)^2+\left(UCA_{1z}-UCA_{2z}\right)^2\right)}{\left(UCA_{1x}-UCA_{2x}\right)^2+\left(UCA_{1y}-UCA_{2y}\right)^2+\left(UCA_{1z}-UCA_{2z}\right)^2} - \frac{\left(UCA_{1z}-UCA_{2z}\right)\left(\begin{array}{l}\left(UCA_{1x}-UCA_{2x}\right)\left(UCA_{1x}-UCA_{3x}\right)+\\ \left(UCA_{1y}-UCA_{2y}\right)\left(UCA_{1y}-UCA_{3y}\right)+\\ \left(UCA_{1z}-UCA_{2z}\right)\left(UCA_{1z}-UCA_{3z}\right)\end{array}\right)}{\left(UCA_{1x}-UCA_{2x}\right)^2+\left(UCA_{1y}-UCA_{2y}\right)^2+\left(UCA_{1z}-UCA_{2z}\right)^2}. \quad \textbf{(17)}$$

Radius of upper control arm is:

$$r_{UCA} = \sqrt{\left(-UCA_{12x}+UCA_{3x}\right)^2+\left(-UCA_{12y}+UCA_{3y}\right)^2+\left(-UCA_{12z}+UCA_{3z}\right)^2}. \quad \textbf{(18)}$$

Components of unit vectors describing the UCA local coordinate system are:

$$x_{UCAx} = \frac{UCA_{1x}-UCA_{2x}}{\sqrt{\left(UCA_{1x}-UCA_{2x}\right)^2+\left(UCA_{1y}-UCA_{2y}\right)^2+\left(UCA_{1z}-UCA_{2z}\right)^2}}, \quad \textbf{(19)}$$

$$x_{UCAy} = \frac{UCA_{1y}-UCA_{2y}}{\sqrt{\left(UCA_{1x}-UCA_{2x}\right)^2+\left(UCA_{1y}-UCA_{2y}\right)^2+\left(UCA_{1z}-UCA_{2z}\right)^2}}, \quad \textbf{(20)}$$

$$x_{UCAz} = \frac{UCA_{1z} - UCA_{2z}}{\sqrt{\left(UCA_{1x} - UCA_{2x}\right)^2 + \left(UCA_{1y} - UCA_{2y}\right)^2 + \left(UCA_{1z} - UCA_{2z}\right)^2}}, \quad \textbf{(21)}$$

$$y_{UCAx} = \frac{-UCA_{12x} + UCA_{3x}}{\sqrt{\left(UCA_{12x} - UCA_{3x}\right)^2 + \left(UCA_{12y} - UCA_{3y}\right)^2 + \left(UCA_{12z} - UCA_{3z}\right)^2}}, \quad \textbf{(22)}$$

$$y_{UCAy} = \frac{-UCA_{12y} + UCA_{3y}}{\sqrt{\left(UCA_{12x} - UCA_{3x}\right)^2 + \left(UCA_{12y} - UCA_{3y}\right)^2 + \left(UCA_{12z} - UCA_{3z}\right)^2}}, \quad \textbf{(23)}$$

$$y_{UCAz} = \frac{-UCA_{12y} + UCA_{3y}}{\sqrt{\left(UCA_{12x} - UCA_{3x}\right)^2 + \left(UCA_{12y} - UCA_{3y}\right)^2 + \left(UCA_{12z} - UCA_{3z}\right)^2}}. \quad \textbf{(24)}$$

Next the distance between LCA3 and UCA3 is calculated. This distance represents radius of a sphere needed for calculating position of UCA3.

$$r_{CA} = \sqrt{\left(LCA_{3x} - UCA_{3x}\right)^2 + \left(LCA_{3y} - UCA_{3y}\right)^2 + \left(LCA_{3z} - UCA_{3z}\right)^2}. \quad \textbf{(25)}$$

2.1.2. Calculating Lower Control Arm Movement

Movement of suspension starts off with moving the lower control arm. It is needed to specify how much it should move upwards and downwards from the reference position.

First the new positions of lower control arm are calculated in lower control arm local coordinate system. By using this method of calculation, trigonometric functions are avoided which might give unexpected results during optimisation. All is needed is to compute the transformation matrix once, LCA_3 coordinates in LCA local coordinate system, and then using the transformation matrix convert back to global coordinate system.

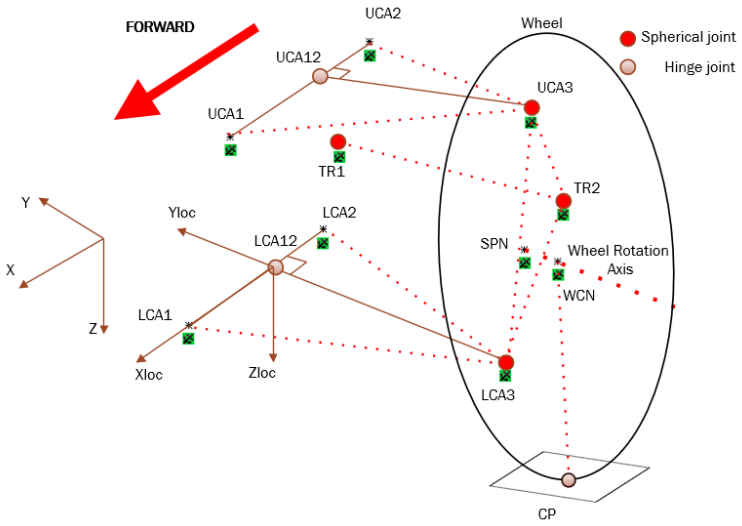

Figure 4 Lower control arm coordinate system

Global LCA_3 is obtained from the following equation which uses column major representation of points and vectors:

$$LCA_3 = R_{LCA} \cdot LCA_{3locLCA} + t_{LCA}. \quad \textbf{(26)}$$

R_{LCA} is a rotation matrix from local to global coordinate system:

$$R_{LCA} = \begin{bmatrix} x_{LCAx} & y_{LCAx} & z_{LCAx} \\ x_{LCAy} & y_{LCAy} & z_{LCAy} \\ x_{LCAz} & y_{LCAz} & z_{LCAz} \end{bmatrix}. \quad \textbf{(27)}$$

x_{LCA}, y_{LCA}, z_{LCA} are previously calculated unit vectors which represent local coordinate system axis directions in global coordinate system. Each of the axis's representations possess the following property:

$$\sqrt{x_x^2 + x_y^2 + x_z^2} = 1. \quad \textbf{(28)}$$

Without this condition the coordinate systems would be skewed.

Since z axis cannot be directly obtained from suspension hardpoints, cross product of x and y axis is used:

$$z_{LCA} = x_{LCA} \times y_{LCA}. \quad \textbf{(29)}$$

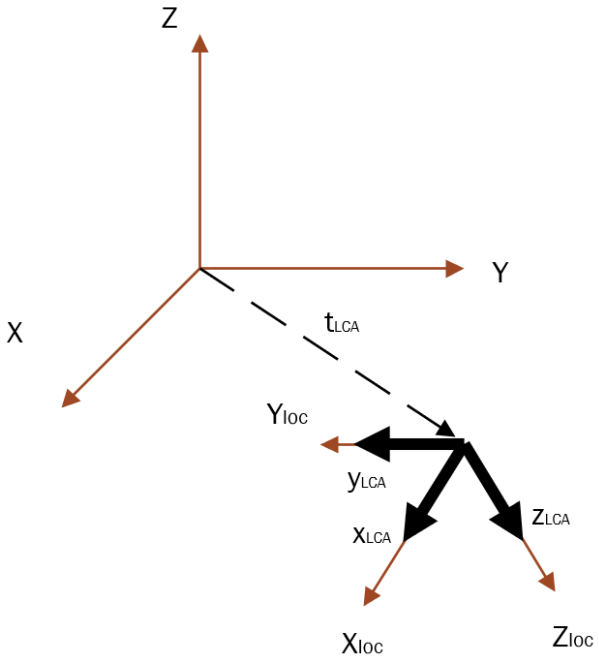

Figure 5 Local to global coordinate system transformation

t_{LCA} represents offset vector between two coordinate systems origins and is measured in global coordinate system. In this case it is equal to coordinates of point LCA_{12} and is added to point after multiplication with rotation matrix and with this addition the transformation is complete.

The only thing left to calculate is positions of LCA_3 in local coordinate system. From Figure 4 it is seen that its x value will always be zero. This simplifies the process because we can convert the problem to 2D circle.

As the wheel goes up, in case of a bump on the road, z value becomes negative and opposite for the wheel going down.

z value will be independent variable and will drive the value of y coordinate. Equation for y value is derived from Pythagorean theorem.

$$y_{LCA3locLCA}^2 + z_{LCA3locLCA}^2 = r_{LCA}^2. \quad \textbf{(30)}$$

From previous equation y_{LCA3loc} is expressed explicitly and negative sign is chosen since it is on the negative side of the y axis:

$$y_{LCA3locLCA} = -\sqrt{r_{LCA}^2 - z_{LCA3locLCA}^2}. \quad \textbf{(31)}$$

z_{LCA3loc} is defined by the user through incrementation, wheel up and wheel down movement. Wheel up and down movements are divided by number of increments and these values represent all the values for which y_{LCA3loc} is calculated.

Since the goal is to move the LCA for a certain amount up or down in a global coordinate system, it is needed to calculate that amount in local coordinate system.

It is not known in advance what will be the values of x_{LCA3} and y_{LCA3}, however, z_{LCA3} is easy to calculate:

$$z_{LCA3\max} = z_{LCA3} + WVERT, \quad \textbf{(32)}$$

$$z_{LCA3\min} = z_{LCA3} - WVERT. \quad \textbf{(33)}$$

Now that global z_{LCA3} coordinate is known it is needed to find local z_{LCA3loc} coordinate so y_{LCA3loc} can be calculated aswell. This is done in three steps:

1. plane parallel to global XY with z value equal to LCA_3 in upmost position is transformed to local coordinate system,
2. plane in local coordinate system is converted to line by setting x to zero,
3. intersect the line with circle.

Starting with first step, plane equation in global coordinate system is:

$$z - z_{LCA3} - WVERT = 0. \quad \textbf{(34)}$$

Comparing previous equation to general plane definition

$$Ax + By + Cz + D = 0, \tag{35}$$

It is noted that A and B coefficients are zero, C is one and D is -z_{LCA3}-*WVERT*. Since transformation matrix from LCA local to global coordinate system is known, starting equation will be transformation of local plane parameters to global plane parameters:

$$\begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} R_{LCA} & \vec{t}_{LCA} \\ \vec{0} & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}. \tag{36}$$

By expressing local plane parameters from previous equation, the following is obtained:

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} R_{LCA}^T & -R_{LCA}^T \cdot \vec{t}_{LCA} \\ \vec{0} & 1 \end{bmatrix} \cdot \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} R_{LCA}^T & -R_{LCA}^T \cdot \vec{t}_{LCA} \\ \vec{0} & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ LCA_{3z} + WVERT \end{bmatrix}. \tag{37}$$

Now that parameters for plane in local coordinate system are known, it is converted to line by setting x value to zero:

$$a \cdot 0 + by_{LCA3loc} + cz_{LCA3loc} + d = 0. \tag{38}$$

$z_{LCA3loc}$ is the wanted value and because it is known that there will be two pairs of solutions, the one with more negative $y_{LCA3loc}$ value is correct because of the orientation of local coordinate system. Therefore, $y_{LCA3loc}$ is expressed explicitly so $z_{LCA3loc}$ can be calculated directly by substituting $y_{LCA3loc}$:

$$y_{LCA3loc} = \frac{-cz_{LCA3loc} - d}{b}. \tag{39}$$

Lastly, after inserting previous equation to equation 30, $z_{LCA3loc}$ value for intersection between circle and line is found:

$$z_{loc} = \frac{-cd \pm b\sqrt{c^2 r_{LCA}^2 + b^2 r_{LCA}^2 - d^2}}{b^2 + c^2}. \tag{40}$$

In previous expression solution with positive square root is chosen.

Final representation of LCA_3 in local coordinate system is:

$$LCA_{3locLCA} = \left(0, -\sqrt{r_{LCA}^2 - z_{LCA3locLCA}^2}, z_{LCA3locLCA}\right). \tag{41}$$

Now by substituting equations 2, 27, and 41 into 26 position of LCA_3 in global coordinate system is obtained.

2.1.3. Calculating Upper Control Arm Movement

Position of UCA3 is obtained by intersecting spatial circle of upper control arm and a sphere with radius r_{CA} with origin in LCA3.

$$\left(x - x_{LCA3}\right)^2 + \left(y - y_{LCA3}\right)^2 + \left(z - z_{LCA3}\right)^2 = {r_{TR2}}^2, \quad \textbf{(42)}$$

$$x_{UCA3}(\theta) = c_1 + r_{UCA} \cdot \cos(\theta) a_1 + r_{UCA} \cdot \sin(\theta) b_1, \quad \textbf{(43)}$$

$$y_{UCA3}(\theta) = c_2 + r_{UCA} \cdot \cos(\theta) a_2 + r_{UCA} \cdot \sin(\theta) b_2, \quad \textbf{(44)}$$

$$z_{UCA3}(\theta) = c_3 + r_{UCA} \cdot \cos(\theta) a_3 + r_{UCA} \cdot \sin(\theta) b_3. \quad \textbf{(45)}$$

This can be done directly, by substituting x, y and z in 42 with x, y and z in 43, 44 and 45, however, using this process requires quite a lot of calculation and use of Newton-Rapson method to converge to solution for each position of LCA_3.

To avoid this, local coordinate system is used, spatial circle then becomes a planar circle and sphere also becomes a planar circle since x value is zero.

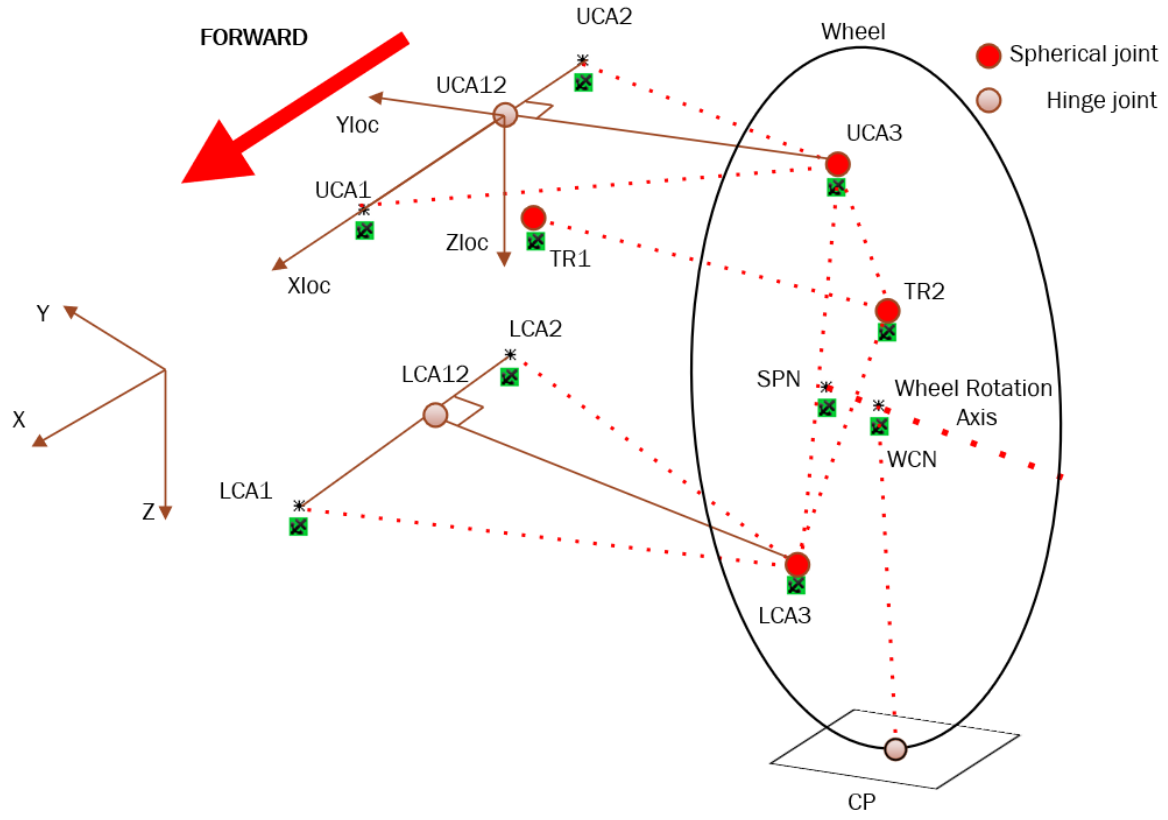

Figure 6 UCA local coordinate system

Now the problem boils down to finding intersection points of 2 circles. The following two equations represent UCA circle and spatial circle with origin in LCA3 in UCA local coordinate system:

$$y_{UCA3locUCA}^2 + z_{UCA3locUCA}^2 = r_{UCA}^2. \quad \textbf{(46)}$$

$$\left(y_{UCA3locUCA} - y_{LCA3locUCA}\right)^2 + \left(z_{UCA3locUCA} - z_{LCA3locUCA}\right)^2 = r_{CA}{}^2 - x_{LCA3locUCA}^2. \quad \textbf{(47)}$$

Before solving this set of equations coordinates of LCA_3 in UCA coordinate system will be determined. This will again be done using the rotation matrix and translation vector. In the next equation LCA_3 in global coordinates is expressed using UCA rotation matrix and offset of UCA coordinate system from global coordinate system, t_{UCA} which in this case is equal to coordinates of point UCA_{12}.

$$LCA_3 = R_{UCA} \cdot LCA_{3locUCA} + t_{UCA}. \quad \textbf{(48)}$$

Since $\mathrm{LCA}_{3\mathrm{locUCA}}$ is needed, equation 41 is rearranged into following:

$$LCA_{3locUCA} = R_{UCA}^T \cdot \left(LCA_3 - t_{UCA}\right). \quad \textbf{(49)}$$

Because rotation matrices are orthonormal it is easy to find their inverse.

UCA rotation matrix is given in the following expression:

$$R_{UCA} = \begin{bmatrix} x_{UCAx} & y_{LCAx} & z_{LCAx} \\ x_{UCAy} & y_{LCAy} & z_{LCAy} \\ x_{UCAz} & y_{LCAz} & z_{LCAz} \end{bmatrix}. \quad \textbf{(50)}$$

As is the case with LCA, here it is also needed to calculate z_{LCA} unit vector by using x_{LCA} and y_{LCA} unit vectors for rotation matrix:

$$z_{UCA} = x_{UCA} \times y_{UCA}. \quad \textbf{(51)}$$

With $\mathrm{LCA}_{3\mathrm{locUCA}}$ defined it is now possible to calculate system of equations 46 and 47.

$$y_{UCA3locUCA} = \frac{y_{LCA3locUCA}\left(-r_{CA}^2 + r_{UCA}^2 + x_{LCA3locUCA}^2 + y_{LCA3locUCA}^2 + z_{LCA3locUCA}^2\right)}{2y_{LCA3locUCA}^2 + 2z_{LCA3locUCA}^2} \pm \frac{z_{LCA3locUCA}\sqrt{\begin{matrix} -r_{CA}^4 + 2r_{CA}^2 r_{UCA}^2 + 2r_{CA}^2 x_{LCA3locUCA}^2 + 2r_{CA}^2 y_{LCA3locUCA}^2 + 2r_{CA}^2 z_{LCA3locUCA}^2 \\ -r_{UCA}^4 - 2r_{UCA}^2 x_{LCA3locUCA}^2 + 2r_{UCA}^2 y_{LCA3locUCA}^2 + 2r_{UCA}^2 z_{LCA3locUCA}^2 \\ -x_{LCA3locUCA}^4 - 2x_{LCA3locUCA}^2 y_{LCA3locUCA}^2 - 2x_{LCA3locUCA}^2 z_{LCA3locUCA}^2 \\ -y_{LCA3locUCA}^4 - 2y_{LCA3locUCA}^2 z_{LCA3locUCA}^2 - z_{LCA3locUCA}^4 \end{matrix}}}{2y_{LCA3locUCA}^2 + 2z_{LCA3locUCA}^2}, \quad \textbf{(52)}$$

$$z_{UCA3locUCA} = \frac{z_{LCA3locUCA}\left(-r_{CA}^2 + r_{UCA}^2 + x_{LCA3locUCA}^2 + y_{LCA3locUCA}^2 + z_{LCA3locUCA}^2\right)}{2\left(y_{LCA3locUCA}^2 + z_{LCA3locUCA}^2\right)} \mp \frac{y_{LCA3locUCA}\sqrt{\begin{matrix}-r_{CA}^4 + 2r_{CA}^2 r_{UCA}^2 + 2r_{CA}^2 x_{LCA3locUCA}^2 + 2r_{CA}^2 y_{LCA3locUCA}^2 + 2r_{CA}^2 z_{LCA3locUCA}^2 \\ -r_{UCA}^4 - 2r_{UCA}^2 x_{LCA3locUCA}^2 + 2r_{UCA}^2 y_{LCA3locUCA}^2 + 2r_{UCA}^2 z_{LCA3locUCA}^2 \\ -x_{LCA3locUCA}^4 - 2x_{LCA3locUCA}^2 y_{LCA3locUCA}^2 - 2x_{LCA3locUCA}^2 z_{LCA3locUCA}^2 \\ -y_{LCA3locUCA}^4 - 2y_{LCA3locUCA}^2 z_{LCA3locUCA}^2 - z_{LCA3locUCA}^4\end{matrix}}}{2\left(y_{LCA3locUCA}^2 + z_{LCA3locUCA}^2\right)}. \quad \textbf{(53)}$$

Two solutions are obtained and only one is needed. Because relative position of points is always the same, as can be seen in the following figure, it is easy to conclude that solution with more negative *z* value is correct.

Because *y* value of sphere origin ($y_{\text{LCA3locUCA}}$) is negative and dictates the sign of $z_{\text{UCA3locUCA}}$ square root value, second solution is chosen. So, $y_{\text{UCA3locLCA}}$ has negative sign in front of square root, while $z_{\text{UCA3locLCA}}$ has positive sign in front of square root.

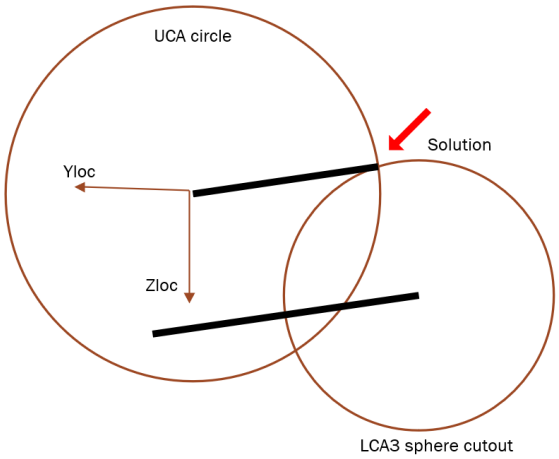

Figure 7 Location of UCA3 point after calculating intersection

The coordinates of UCA_3 in UCA local coordinate system are:

$$UCA_{3locUCA} = \left(0, y_{UCA3locUCA}, z_{UCA3locUCA}\right). \quad \textbf{(54)}$$

Finally, the coordinates of UCA_3 in global coordinate system are:

$$UCA_3 = R_{UCA} \cdot UCA_{3locUCA} + t_{UCA}. \quad \textbf{(55)}$$

2.1.4. Calculating Tie Rod Movement

Finding position of TR_2 point is done in much the same way as finding the position of UCA_3. Again, intersections between a sphere and a circle are obtained and only one of two obtained intersections is valid.

Local coordinate system is used to reduce dimensionality and simplify solution. Projecting TR_2 to line connecting UCA_3 and LCA_3 is origin of the coordinate system.

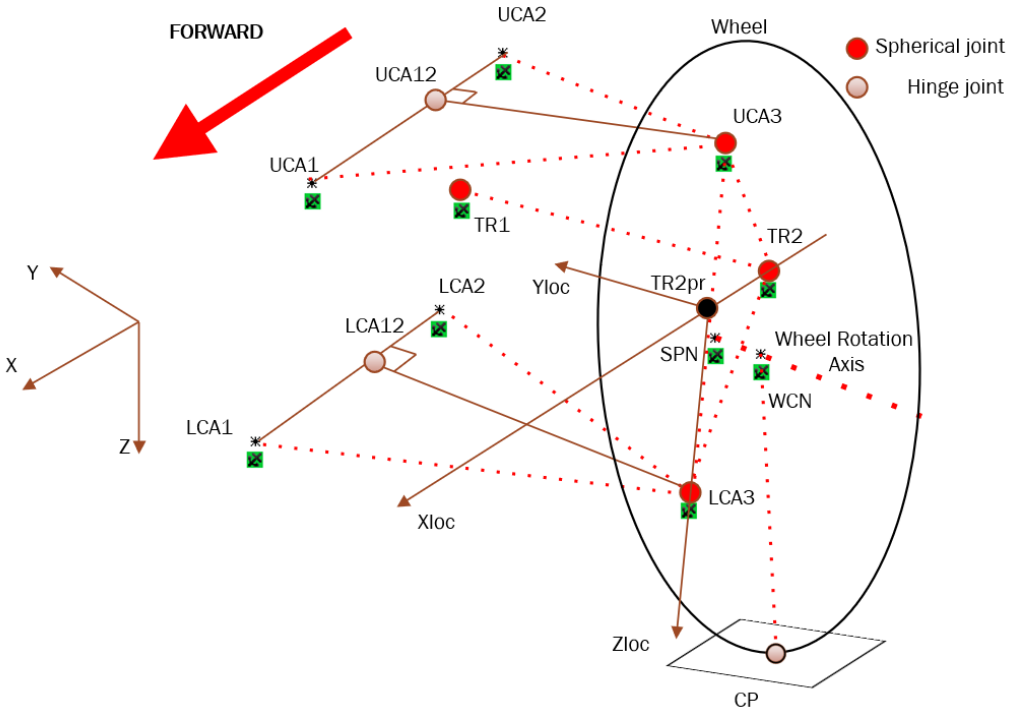

Figure 8 Tie rod local coordinate system

Sphere described by tie rod is transformed to TR coordinate system with *z* value set to zero. This leads to the same set of equation as in the case of upper control arm with the exception being that now *z* value is set to zero.

Since tie rod can be in front or rear from wheel centre, both intersections are valid, depending on suspension configuration. Switching will be done by comparing the *x* values of *WCN* and TR_2 and choosing appropriate sign in front of square root equation member.

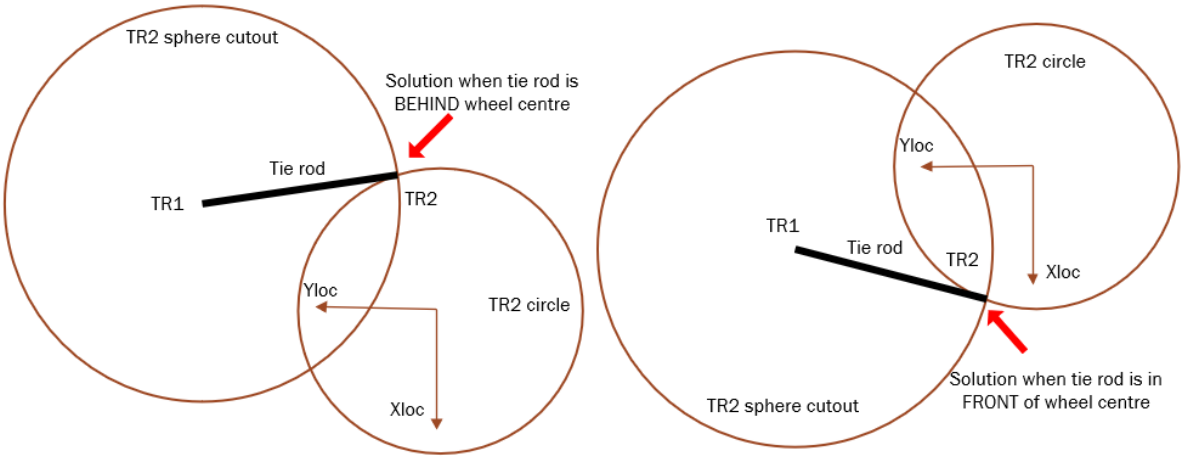

Figure 9 TR2 point location in local coordinate system

Location of TR_2 point is given by the following equation:

$$TR_2 = R_{TR} \cdot TR_{2locTR} + t_{TR}. \tag{56}$$

R_{TR} is a rotation matrix with the following form:

$$R_{TR} = \begin{bmatrix} x_{TRx} & y_{TRx} & z_{TRx} \\ x_{TRy} & y_{TRy} & z_{TRy} \\ x_{TRz} & y_{TRz} & z_{TRz} \end{bmatrix}. \tag{57}$$

Vectors x_{TR}, y_{TR} and z_{TR} are unit vectors perpendicular to each other. x_{TR} and z_{TR} can be obtained directly by manipulating suspension hardpoints, while y_{TR} is obtained as a cross product of x_{TR} and z_{TR}.

t_{TR} is an offset vector and is equal to coordinates of TR_2 projection to line connecting points UCA_3 and LCA_3. TR_2 projection is obtained as were UCA_{12} and LCA_{12} and only the result will be shown:

$$TR2_{prx} = \frac{UCA_{3x}\left(\left(LCA_{3x}-UCA_{3x}\right)^2+\left(LCA_{3y}-UCA_{3y}\right)^2+\left(LCA_{3z}-UCA_{3z}\right)^2\right)}{\left(LCA_{3x}-UCA_{3x}\right)^2+\left(LCA_{3y}-UCA_{3y}\right)^2+\left(LCA_{3z}-UCA_{3z}\right)^2} + \frac{\left(LCA_{3x}-UCA_{3x}\right)\left(\begin{array}{l}\left(LCA_{3x}-UCA_{3x}\right)\left(TR_{2x}-UCA_{3x}\right)+\\ \left(LCA_{3y}-UCA_{3y}\right)\left(TR_{2y}-UCA_{3y}\right)+\\ \left(LCA_{3z}-UCA_{3z}\right)\left(TR_{2z}-UCA_{3z}\right)\end{array}\right)}{\left(LCA_{3x}-UCA_{3x}\right)^2+\left(LCA_{3y}-UCA_{3y}\right)^2+\left(LCA_{3z}-UCA_{3z}\right)^2}, \tag{58}$$

$$TR2_{pry} = \frac{UCA_{3y}\left(\left(LCA_{3x}-UCA_{3x}\right)^2+\left(LCA_{3y}-UCA_{3y}\right)^2+\left(LCA_{3z}-UCA_{3z}\right)^2\right)}{\left(LCA_{3x}-UCA_{3x}\right)^2+\left(LCA_{3y}-UCA_{3y}\right)^2+\left(LCA_{3z}-UCA_{3z}\right)^2} + \frac{\left(LCA_{3y}-UCA_{3y}\right)\left(\begin{array}{l}\left(LCA_{3x}-UCA_{3x}\right)\left(TR_{2x}-UCA_{3x}\right)+\\ \left(LCA_{3y}-UCA_{3y}\right)\left(TR_{2y}-UCA_{3y}\right)+\\ \left(LCA_{3z}-UCA_{3z}\right)\left(TR_{2z}-UCA_{3z}\right)\end{array}\right)}{\left(LCA_{3x}-UCA_{3x}\right)^2+\left(LCA_{3y}-UCA_{3y}\right)^2+\left(LCA_{3z}-UCA_{3z}\right)^2}, \tag{59}$$

$$TR2_{prz} = \frac{UCA_{3z}\left(\left(LCA_{3x}-UCA_{3x}\right)^2+\left(LCA_{3y}-UCA_{3y}\right)^2+\left(LCA_{3z}-UCA_{3z}\right)^2\right)}{\left(LCA_{3x}-UCA_{3x}\right)^2+\left(LCA_{3y}-UCA_{3y}\right)^2+\left(LCA_{3z}-UCA_{3z}\right)^2} + \frac{\left(LCA_{3z}-UCA_{3z}\right)\begin{pmatrix}\left(LCA_{3x}-UCA_{3x}\right)\left(TR_{2x}-UCA_{3x}\right)+\\ \left(LCA_{3y}-UCA_{3y}\right)\left(TR_{2y}-UCA_{3y}\right)+\\ \left(LCA_{3z}-UCA_{3z}\right)\left(TR_{2z}-UCA_{3z}\right)\end{pmatrix}}{\left(LCA_{3x}-UCA_{3x}\right)^2+\left(LCA_{3y}-UCA_{3y}\right)^2+\left(LCA_{3z}-UCA_{3z}\right)^2}. \quad \textbf{(60)}$$

Now that distance of TR_{2pr} is known for reference position, it is needed to find its position during wheel movement. This is done using 3D line set of equations:

$$TR_{2prx} = \left(UCA_{3x} - LCA_{3x}\right)t + LCA_{3x}, \quad \textbf{(61)}$$

$$TR_{2pry} = \left(UCA_{3y} - LCA_{3y}\right)t + LCA_{3y}, \quad \textbf{(62)}$$

$$TR_{2prz} = \left(UCA_{3z} - LCA_{3z}\right)t + LCA_{3z}, \quad \textbf{(63)}$$

UCA_3 and LCA_3 are variables as the wheel moves up and down, while t is a constant and follows from the reference position:

$$t = \frac{TR_{2prxref} - LCA_{3x}}{UCA_{3x} - LCA_{3x}}. \quad \textbf{(64)}$$

Next, the steering arm length or the distance between TR_2 and TR_{2pr} is calculated:

$$r_{ST} = \sqrt{\left(TR_{2x}-TR_{2prx}\right)^2+\left(TR_{2y}-TR_{2pry}\right)^2+\left(TR_{2z}-TR_{2prz}\right)^2}. \quad \textbf{(65)}$$

This distance describes spatial circle around LCA_3UCA_3 axis.

Distance between TR_1 and TR_2 describes sphere with which the circle will be intersected:

$$r_{TR} = \sqrt{\left(TR_{2x}-TR_{1x}\right)^2+\left(TR_{2y}-TR_{1y}\right)^2+\left(TR_{2z}-TR_{1z}\right)^2}. \quad \textbf{(66)}$$

Unit vectors x_{TR} and z_{TR} are again obtained in the same way as in LCA calculation and only the result is given:

$$x_{TRx} = \frac{TR_{2prx}-TR_{2x}}{\sqrt{\left(TR_{2prx}-TR_{2x}\right)^2+\left(TR_{2pry}-TR_{2y}\right)^2+\left(TR_{2prz}-TR_{2z}\right)^2}}, \quad \textbf{(67)}$$

$$x_{TRy} = \frac{TR_{2pry}-TR_{2y}}{\sqrt{\left(TR_{2prx}-TR_{2x}\right)^2+\left(TR_{2pry}-TR_{2y}\right)^2+\left(TR_{2prz}-TR_{2z}\right)^2}}, \quad \textbf{(68)}$$

$$x_{TRz} = \frac{TR_{2prz}-TR_{2z}}{\sqrt{\left(TR_{2prx}-TR_{2x}\right)^2+\left(TR_{2pry}-TR_{2y}\right)^2+\left(TR_{2prz}-TR_{2z}\right)^2}}, \quad \textbf{(69)}$$

$$z_{TRx} = \frac{LCA_{3x} - UCA_{3x}}{\sqrt{\left(LCA_{3x} - UCA_{3x}\right)^2 + \left(LCA_{3y} - UCA_{3y}\right)^2 + \left(LCA_{3z} - UCA_{3z}\right)^2}}, \quad \textbf{(70)}$$

$$z_{TRy} = \frac{LCA_{3y} - UCA_{3y}}{\sqrt{\left(LCA_{3x} - UCA_{3x}\right)^2 + \left(LCA_{3y} - UCA_{3y}\right)^2 + \left(LCA_{3z} - UCA_{3z}\right)^2}}, \quad \textbf{(71)}$$

$$z_{TRz} = \frac{LCA_{3z} - UCA_{3z}}{\sqrt{\left(LCA_{3x} - UCA_{3x}\right)^2 + \left(LCA_{3y} - UCA_{3y}\right)^2 + \left(LCA_{3z} - UCA_{3z}\right)^2}}, \quad \textbf{(72)}$$

y_{TR} unit vector is obtained from cross product of z_{TR} and x_{TR}.

$$y_{TR} = z_{TR} \times x_{TR}. \quad \textbf{(73)}$$

Intersection points are found using the following set of equations in TR local coordinate system where z coordinate has a value of zero:

$$x_{TR2locTR}^2 + y_{TR2locTR}^2 = r_{ST}^2. \quad \textbf{(74)}$$

$$\left(x_{TR2locTR} - x_{TR1locTR}\right)^2 + \left(y_{TR2locTR} - y_{TR1locTR}\right)^2 = r_{TR}{}^2 - z_{TR1locTR}^2. \quad \textbf{(75)}$$

In equation 75 TR_1 coordinates in TR local coordinate system are still unknown. Their coordinates are calculated using the same procedure as for UCA:

$$TR_{1locTR} = R_{TR}^T \cdot \left(TR_1 - t_{TR}\right). \quad \textbf{(76)}$$

From previous equation values for x_{TR1locTR}, y_{TR1locTR} and z_{TR1locTR} are obtained.

Now all the unknowns in previous system of equations are known and it can be solved. Two possible solutions are obtained and as said before, depending on whether the tie rod is in front of or behind the wheel centre, one or the other solution is chosen.

After solving equations 74 and 75 these are the results:

$$x_{TR2locTR} = \frac{x_{TR1locTR}\left(r_{ST}^2 - r_{TR}^2 + x_{TR1locTR}^2 + y_{TR1locTR}^2 + z_{TR1locTR}^2\right)}{2\left(x_{TR1locTR}^2 + y_{TR1locTR}^2\right)} \pm \frac{y_{TR1locTR}\sqrt{\begin{matrix}-r_{ST}^4 + 2r_{ST}^2 r_{TR}^2 + 2r_{ST}^2 x_{TR1locTR}^2 + 2r_{ST}^2 y_{TR1locTR}^2 - 2r_{ST}^2 z_{TR1locTR}^2 \\ -r_{TR}^4 + 2r_{TR}^2 x_{TR1locTR}^2 + 2r_{TR}^2 y_{TR1locTR}^2 + 2r_{TR}^2 z_{TR1locTR}^2 \\ -x_{TR1locTR}^4 - 2x_{TR1locTR}^2 y_{TR1locTR}^2 - 2x_{TR1locTR}^2 z_{TR1locTR}^2 \\ -y_{TR1locTR}^4 - 2y_{TR1locTR}^2 z_{TR1locTR}^2 - z_{TR1locTR}^4\end{matrix}}}{2\left(x_{TR1locTR}^2 + y_{TR1locTR}^2\right)}. \quad \textbf{(77)}$$

$$y_{TR2locTR} = \frac{y_{TR1locTR}\left(r_{ST}^2 - r_{TR}^2 + x_{TR1locTR}^2 + y_{TR1locTR}^2 + z_{TR1locTR}^2\right)}{2\left(x_{TR1locTR}^2 + y_{TR1locTR}^2\right)} \mp \frac{x_{TR1locTR}\sqrt{\begin{matrix}-r_{ST}^4 + 2r_{ST}^2 r_{TR}^2 + 2r_{ST}^2 x_{TR1locTR}^2 + 2r_{ST}^2 y_{TR1locTR}^2 - 2r_{ST}^2 z_{TR1locTR}^2 \\ -r_{TR}^4 + 2r_{TR}^2 x_{TR1locTR}^2 + 2r_{TR}^2 y_{TR1locTR}^2 + 2r_{TR}^2 z_{TR1locTR}^2 \\ -x_{TR1locTR}^4 - 2x_{TR1locTR}^2 y_{TR1locTR}^2 - 2x_{TR1locTR}^2 z_{TR1locTR}^2 \\ -y_{TR1locTR}^4 - 2y_{TR1locTR}^2 z_{TR1locTR}^2 - z_{TR1locTR}^4\end{matrix}}}{2\left(x_{TR1locTR}^2 + y_{TR1locTR}^2\right)}. \quad \textbf{(78)}$$

$TR_{2\mathrm{locTR}}$ point in TR coordinate system has the following coordinates:

$$TR_{2locTR} = \left(x_{TR2locTR}, y_{TR2locTR}, 0.\right) \quad \textbf{(79)}$$

By inserting 79 to 56 TR_2 in global coordinate system is obtained.

If steering of the wheel is needed, then all the constants are calculated as shown above with the only difference being addition of a value to y value of TR_1:

$$TR_1 = \left(TR_{1x}, TR_{1y} + WSTEER, TR_{1z}\right). \quad \textbf{(80)}$$

Then expression above is substituted to equation 76 and steering is achieved.

2.1.5. Calculating SPN and WCN Movement

Since *WCN* and *SPN* are points on upright they will be moving together with TR_2. This means it is possible to use previously derived TR coordinate system for determining positions of mentioned hardpoints as the wheel moves.

$$WCN = R_{TR} \cdot WCN_{locTR} + t_{TR}. \quad \textbf{(81)}$$

$$SPN = R_{TR} \cdot SPN_{locTR} + t_{TR}. \quad \textbf{(82)}$$

The above equations give positions of *WCN* and *SPN* after moving the wheel to a new position. In these equations t_{TR} and R_{TR} are variables, while WCN_{locTR} and SPN_{locTR} are constants because their position relative to TR_2, LCA_3 and UCA_3 is always the same for one configuration of suspension.

$$_{in}WCN = {}_{in}R_{TR} \cdot WCN_{locTR} + {}_{in}t_{TR}. \quad \textbf{(83)}$$

$$_{in}SPN = {}_{in}R_{TR} \cdot SPN_{locTR} + {}_{in}t_{TR}, \quad \textbf{(84)}$$

where index in front of a variable denotes initial value.

By expressing WCN_{locTR} and SPN_{locTR} the following set of equations is obtained:

$$WCN_{locTR} = {}_{in}R_{TR}^{T}\left({}_{in}WCN - {}_{in}t_{TR}\right), \quad \textbf{(85)}$$

$$SPN_{locTR} = {}_{in}R_{TR}^{T}\left({}_{in}SPN - {}_{in}t_{TR}\right). \quad \textbf{(86)}$$

Equations 85 and 86 can now be inserted to 81 and 82 which will produce correct positions of *WCN* and *SPN*.

2.1.6. Calculating Contact Patch Movement

Contact patch point (*CP*) is the last of characteristic points needed to calculate to be able to calculate all suspension characteristics. There are multiple cases for calculating *CP*

- only one half of suspension is being observed
    - chassis rolling motion is simulated
    - vertical motion of chassis is simulated
    - combination of the two is simulated
- the whole suspension is being observed
    - pitching movement of chassis is simulated
    - rolling movement of chassis is simulated
    - vertical movement of chassis is simulated
    - combination of all three is simulated

Since the focus of this master thesis is suspension kinematics optimisation which is done with respect to cornering because then the vehicle is utilising tyres to the maximum, only chassis rolling motion is of concern.

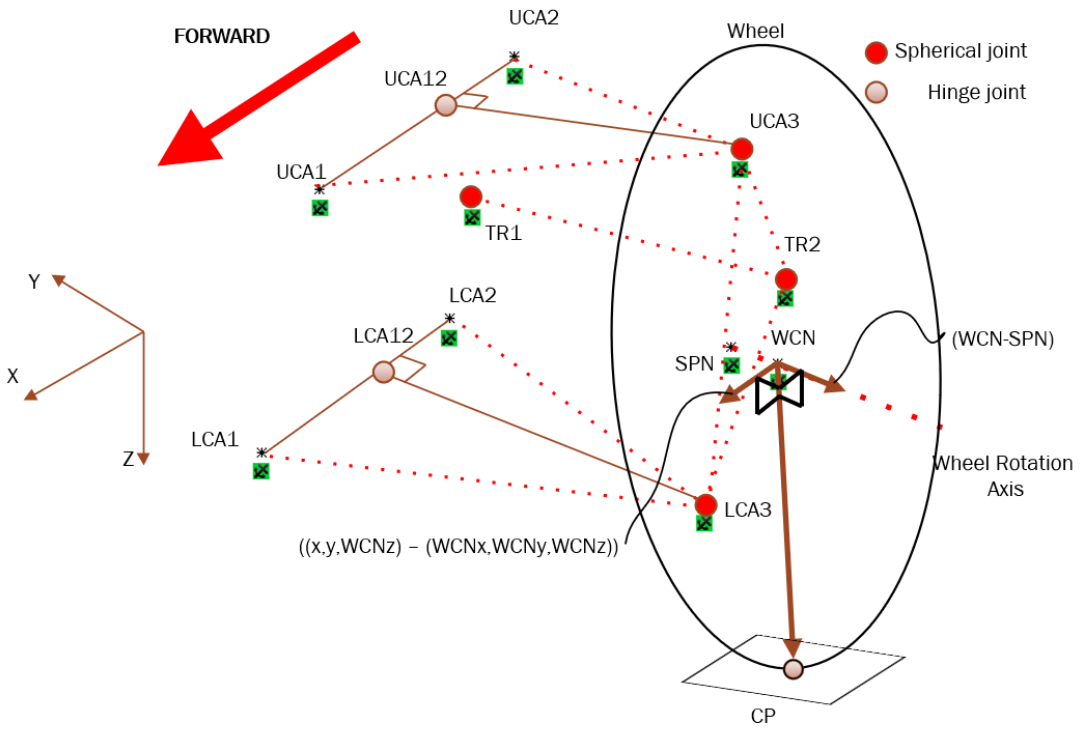

Figure 10 Calculating CP from vectors

It is assumed that ground plane is perpendicular to YZ plane of vehicle global coordinate system. This means that ground plane will be touching circles representing wheels in the point with the lowest *z* value.

CP could be calculated by doing derivation on the Z axis of parametric circle equation, however that would require use of arcus tangents function which could be problematic later when calculating results in C++.

To avoid this problem vectors are used as can be seen in the previous figure.

Location of *CP* is given by the following expression:

$$CP_i = \frac{\left(\left(\left(SPN_i - WCN_i\right)\times\overrightarrow{v_0}\right)\times\left(SPN_i - WCN_i\right)\right)}{\left|\left(\left(SPN_i - WCN_i\right)\times\overrightarrow{v_0}\right)\times\left(SPN_i - WCN_i\right)\right|}\cdot WRADIUS + WCN_i, \qquad \textbf{(87)}$$

where *i* index represents current suspension position.

In this equation only v_0 is unknown and it represents an auxiliary vector needed for calculating ground plane normal.

For only half of suspension observed this vector v_0 has the following value:

$$v_0 = (0,0,-20). \qquad \textbf{(88)}$$

z value of -20 is arbitrary, it is only important that it is negative and not too large or too small with respect to length of vector (*SPN* - *WCN*) for numerical stability.

2.2. Calculating Suspension Characteristics

After all the positions of suspension are known it is possible to calculate its characteristics. The following table sums up all the characteristic values that will be calculated.

Table 2 Mathematically described suspension characteristics

Suspension characteristic	Suspension characteristic
Camber angle	Anti-rise
Toe angle	Anti-dive
Caster angle	Anti-lift
Roll centre height	LCA_3 inside wheel free radius
Caster trail	UCA_3 inside wheel free radius
Kingpin angle	TR_2 inside wheel free radius
Scrub radius	LCA_3 distance to WCN along wheel axis
Half-track change	UCA_3 distance to WCN along wheel axis
Wheelbase change	TR_2 distance to WCN along wheel axis
Anti- squat	

2.2.1. Camber Angle

Camber angle is the angle that the wheel axis is closing with the ground plane. It can be either positive- lower half is moved towards the vehicle or negative- lower half is moved away from the vehicle.

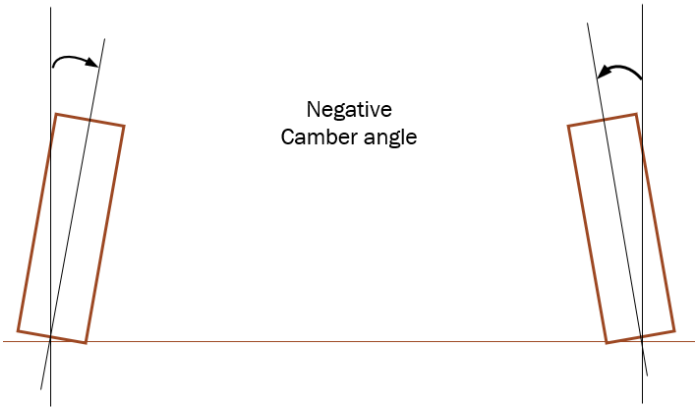

Figure 11 Negative Camber angle

It is calculated using *WCN*, *SPN* and *CP* points. Each position of wheel has a new ground plane defined by *CP*. Somewhat simplified approach is used to calculate Camber angle. It is assumed that during rolling of the chassis the left LCA_3 moves up as much as the right LCA_3 moves down in the Z axis with respect to chassis and vice versa.

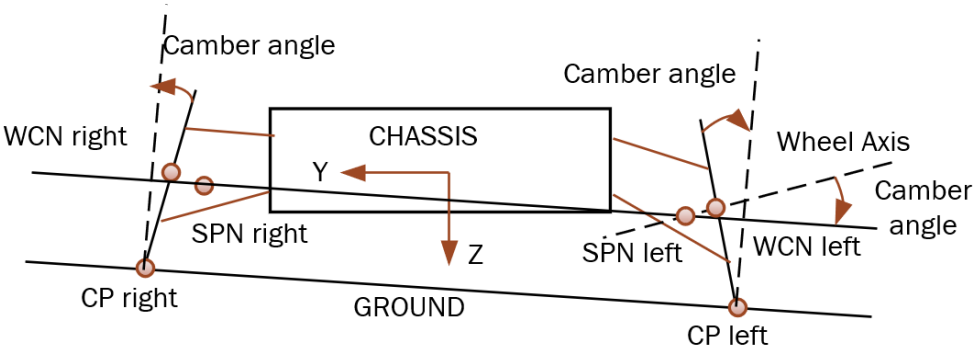

Figure 12 Camber angle during roll viewed from front to back

To calculate camber angle, it is needed to first find the ground plane. Since only one half of the suspension is being optimised, ground plane is defined as a plane going through points CP_{Left}, CP_{Right} and (0, CP_{YLeft}, CP_{ZLeft}), it is perpendicular to YZ global plane. After the plane is found, the angle between plane and wheel axis defined by *WCN* and *SPN* is calculated.

Ground plane normal can be calculated as a line perpendicular to a line going through points

- (CP_{XLeft}, CP_{YRight}, CP_{ZRight}) and
- (CP_{XLeft}, CP_{YLeft}, CP_{ZLeft}).

The resulting ground plane vector normal expression is:

$$\overrightarrow{ground} = (a,b,c) = \left(0, CP_{Zleft} - CP_{Zright}, -CP_{Yleft} + CP_{Yright}\right). \quad \textbf{(89)}$$

Since suspension is symmetric, CP_{Right} corresponds to mirrored CP_{Left} in opposite vertical wheel position. For instance, if left side has an amplitude of positions calculated:

(x_0, y_0, z_0), (x_1, y_1, z_1), (x_2, y_2, z_2),

then the corresponding CP_{right} values will be:

(x_2, -y_2, z_2), (x_1, -y_1, z_1), (x_0, -y_0, z_0).

Plane parallel to the ground is obtained using the plane vector normal and a *SPN* point. Then the *WCN* point is projected to ground plane, giving the WCN_{pr} point. Using point *WCN*, *SPN* and WCN_{pr} it is now possible to construct a triangle and measure the angle representing Camber.

$$WCN_{pr} = \begin{pmatrix} WCN_x + \dfrac{a\left(SPN_x a + SPN_y b + SPN_z c - WCN_x a - WCN_y b - WCN_z c\right)}{a^2 + b^2 + c^2}, \\ WCN_y + \dfrac{b\left(SPN_x a + SPN_y b + SPN_z c - WCN_x a - WCN_y b - WCN_z c\right)}{a^2 + b^2 + c^2}, \\ WCN_z + \dfrac{c\left(SPN_x a + SPN_y b + SPN_z c - WCN_x a - WCN_y b - WCN_z c\right)}{a^2 + b^2 + c^2} \end{pmatrix} \quad \textbf{(90)}$$

If the vector $WCN_{pr} - WCN$ has positive Z component then the Camber angle is negative, otherwise it is positive.

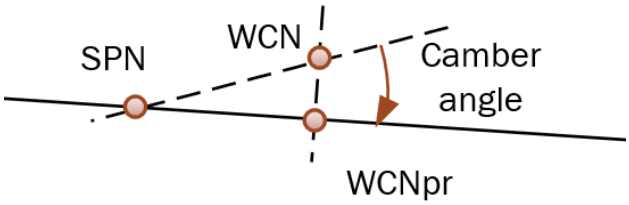

Figure 13 Camber calculation

Wheel axis vector is defined by:

$$\overrightarrow{axis} = \left(WCN_x - SPN_x, WCN_y - SPN_y, WCN_z - SPN_z,\right), \quad \textbf{(91)}$$

And lastly Camber angle is obtained:

$$\text{Camber angle} = \sin^{-1}\left(\frac{\left|\overrightarrow{\left(WCN_{pr} - WCN\right)}\right|}{\left|\overrightarrow{axis}\right|}\right). \quad \textbf{(92)}$$

2.2.2. Toe Angle

Toe angle is the angle that the wheel closes with the longitudinal axis of the vehicle as can be seen in the following figure.

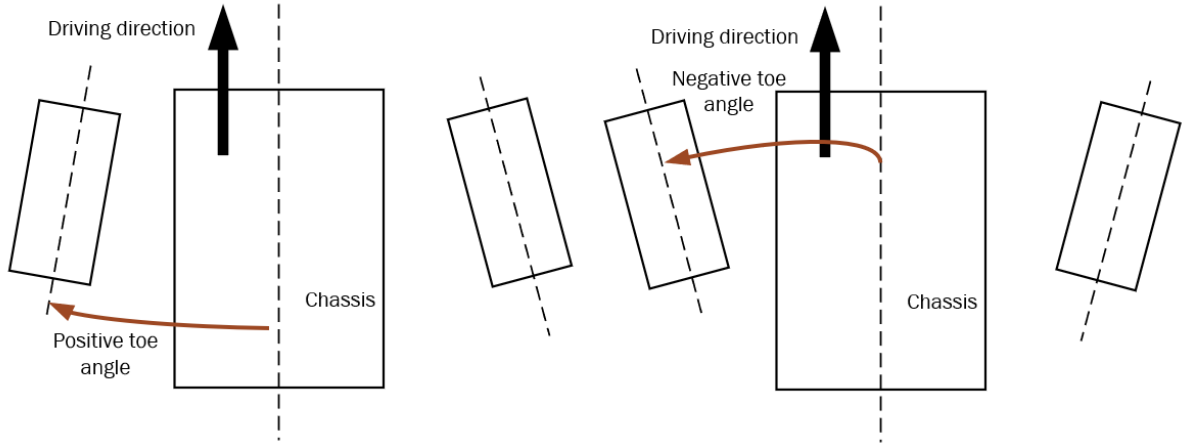

Figure 14 Positive and negative toe angle

To calculate toe angle only SPN and WCN points are needed. From those two points auxiliary point is derived:

(SPN_x, WCN_y, WCN_z).

The following figure shows relation of points and toe angle.

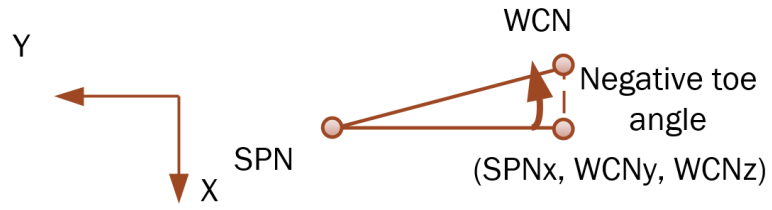

Figure 15 Toe angle calculation

Toe angle is simply calculated using arcus cosine:

$$\text{Toe angle} = \cos^{-1}\left(\frac{\left|\overrightarrow{\left(\left(SPN_x, WCN_y, WCN_z\right) - WCN\right)}\right|}{\left|\overrightarrow{\left(WCN - SPN\right)}\right|}\right). \quad \textbf{(93)}$$

2.2.3. Caster angle

Caster angle is defined as the smaller angle that the line seen from side of the chassis connecting UCA_3 and LCA_3 closes with the YZ plane of the chassis.

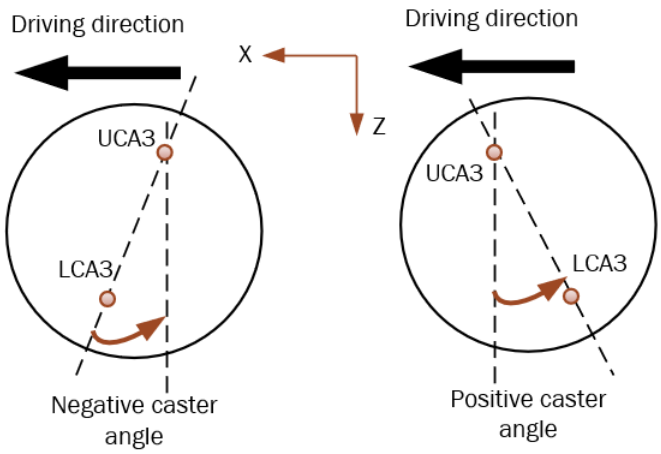

Figure 16 Negative and Positive Caster Angle

For measuring the angle two auxiliary points are created with the coordinates (SPN_x, WCN_y, SPN_z), (WCN_x, WCN_y, SPN_z).

Finally, caster angle is then defined as:

$$\text{Caster angle} = \tan^{-1}\left(\frac{-LCA_{3x} + UCA_{3x}}{-UCA_{3z} + LCA_{3z}}\right). \quad \textbf{(94)}$$

2.2.4. Roll centre height

Roll centre height is defined as can be seen in the following figure.

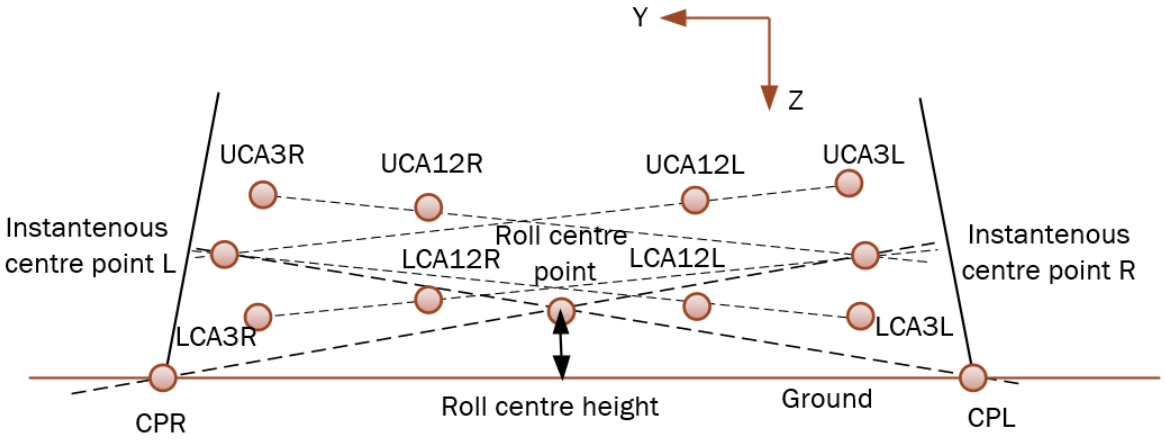

Figure 17 Roll centre height definition

Instantaneous centre points are obtained after intersecting UCA and LCA planes with projection plane parallel to YZ global plane going with X value of (CP_R + CP_L)/2.

There are 2 cases in calculating roll centre height:

- upper and lower control arm planes define parallel lines when intersected with a plane
- upper and lower control arm planes do not define parallel lines when intersected with a plane.

First intersections between UCA and LCA planes with projection plane are calculated. Since they are projections to a plane parallel to YZ global plane, only two coordinates are needed to describe them. General expressions for these lines are:

$$z_{LCALintrs} = a_{LCALintrs} \cdot y_{LCALintrs} + b_{LCALintrs}, \tag{95}$$

$$z_{UCALintrs} = a_{UCALintrs} \cdot y_{UCALintrs} + b_{UCALintrs}, \tag{96}$$

$$z_{LCARintrs} = a_{LCARintrs} \cdot y_{LCARintrs} + b_{LCARintrs}, \tag{97}$$

$$z_{UCARintrs} = a_{UCARintrs} \cdot y_{UCARintrs} + b_{UCARintrs}, \tag{98}$$

where L indicates left side and R right side of suspension.

General line defined by two points expression for YZ plane is:

$$z - z_1 = \frac{z_2 - z_1}{y_2 - y_1} \cdot (y - y_1), \tag{99}$$

which can be rewritten in the following form containing *a* and *b* parameters:

$$z - z_1 = \frac{z_2 - z_1}{y_2 - y_1} \cdot y - \frac{z_2 - z_1}{y_2 - y_1} \cdot y_1 + z_1 = ay + b. \tag{100}$$

$$a = \frac{z_2 - z_1}{y_2 - y_1}, \tag{101}$$

$$b = -\frac{z_2 - z_1}{y_2 - y_1} \cdot y_1 + z_1. \tag{102}$$

Next are given points and slopes that define intersection lines:

$$LCA_{1Lintrs} = \begin{pmatrix} -LCA_{1Lx}LCA_{2Ly}LCA_{3Lz} + LCA_{1Lx}LCA_{2Lz}LCA_{3Ly} + LCA_{1Ly}LCA_{2Lx}LCA_{3Lz} \\ -LCA_{1Ly}LCA_{2Lz}LCA_{3Lx} - LCA_{1Lz}LCA_{2Lx}LCA_{3Ly} + LCA_{1Lz}LCA_{2Ly}LCA_{3Lx} + \\ \frac{(CP_{Lx} + CP_{Rx})\begin{pmatrix} LCA_{1Ly}LCA_{2Lz} - LCA_{1Ly}LCA_{3Lz} - LCA_{1Lz}LCA_{2Ly} + \\ LCA_{1Lz}LCA_{3Ly} + LCA_{2Ly}LCA_{3Lz} - LCA_{2Lz}LCA_{3Ly} \end{pmatrix}}{2} \\ \hline LCA_{1Lx}LCA_{2Lz} - LCA_{1Lx}LCA_{3Lz} - LCA_{1Lz}LCA_{2Lx} + \\ LCA_{1Lz}LCA_{3Lx} + LCA_{2Lx}LCA_{3Lz} - LCA_{2Lz}LCA_{3Lx} \\ 0 \end{pmatrix}, \tag{103}$$

$$a_{LCALintrs} = \frac{(LCA_{1Lx} - LCA_{2Lx})(LCA_{1Lz} - LCA_{3Lz}) - (LCA_{1Lx} - LCA_{3Lx})(LCA_{1Lz} - LCA_{2Lz})}{(LCA_{1Lx} - LCA_{2Lx})(LCA_{1Ly} - LCA_{3Ly}) - (LCA_{1Lx} - LCA_{3Lx})(LCA_{1Ly} - LCA_{2Ly})}, \tag{104}$$

$$UCA_{1Lintrs} = \begin{pmatrix} \dfrac{\begin{pmatrix} -UCA_{1Lx}UCA_{2Ly}UCA_{3Lz} + UCA_{1Lx}UCA_{2Lz}UCA_{3Ly} + UCA_{1Ly}UCA_{2Lx}UCA_{3Lz} \\ -UCA_{1Ly}UCA_{2Lz}UCA_{3Lx} - UCA_{1Lz}UCA_{2Lx}UCA_{3Ly} + UCA_{1Lz}UCA_{2Ly}UCA_{3Lx} + \\ \left(CP_{Lx} + CP_{Rx}\right)\begin{pmatrix} UCA_{1Ly}UCA_{2Lz} - UCA_{1Ly}UCA_{3Lz} - UCA_{1Lz}UCA_{2Ly} + \\ UCA_{1Lz}UCA_{3Ly} + UCA_{2Ly}UCA_{3Lz} - UCA_{2Lz}UCA_{3Ly} \end{pmatrix} \end{pmatrix}}{2}}{UCA_{1Lx}UCA_{2Lz} - UCA_{1Lx}UCA_{3Lz} - UCA_{1Lz}UCA_{2Lx} + UCA_{1Lz}UCA_{3Lx} + UCA_{2Lx}UCA_{3Lz} - UCA_{2Lz}UCA_{3Lx}}, \\ 0 \end{pmatrix} \quad \textbf{(105)}$$

$$a_{UCALntrs} = \frac{\left(UCA_{1Lx} - UCA_{2Lx}\right)\left(UCA_{1Lz} - UCA_{3Lz}\right) - \left(UCA_{1Lx} - UCA_{3Lx}\right)\left(UCA_{1Lz} - UCA_{2Lz}\right)}{\left(UCA_{1Lx} - UCA_{2Lx}\right)\left(UCA_{1Ly} - UCA_{3Ly}\right) - \left(UCA_{1Lx} - UCA_{3Lx}\right)\left(UCA_{1Ly} - UCA_{2Ly}\right)}, \quad \textbf{(106)}$$

Since the point for each intersection line has *z* value zero, it can also represent *b* parameter in line equation:

$$b_{LCALntrs} = \frac{\begin{pmatrix} -LCA_{1Lx}LCA_{2Ly}LCA_{3Lz} + LCA_{1Lx}LCA_{2Lz}LCA_{3Ly} + LCA_{1Ly}LCA_{2Lx}LCA_{3Lz} \\ -LCA_{1Ly}LCA_{2Lz}LCA_{3Lx} - LCA_{1Lz}LCA_{2Lx}LCA_{3Ly} + LCA_{1Lz}LCA_{2Ly}LCA_{3Lx} + \\ \dfrac{\left(CP_{Lx} + CP_{Rx}\right)\begin{pmatrix} LCA_{1Ly}LCA_{2Lz} - LCA_{1Ly}LCA_{3Lz} - LCA_{1Lz}LCA_{2Ly} + \\ LCA_{1Lz}LCA_{3Ly} + LCA_{2Ly}LCA_{3Lz} - LCA_{2Lz}LCA_{3Ly} \end{pmatrix}}{2} \end{pmatrix}}{LCA_{1Lx}LCA_{2Lz} - LCA_{1Lx}LCA_{3Lz} - LCA_{1Lz}LCA_{2Lx} + LCA_{1Lz}LCA_{3Lx} + LCA_{2Lx}LCA_{3Lz} - LCA_{2Lz}LCA_{3Lx}}, \quad \textbf{(107)}$$

$$b_{UCALntrs} =, \frac{\begin{pmatrix} -UCA_{1Lx}UCA_{2Ly}UCA_{3Lz} + UCA_{1Lx}UCA_{2Lz}UCA_{3Ly} + UCA_{1Ly}UCA_{2Lx}UCA_{3Lz} \\ -UCA_{1Ly}UCA_{2Lz}UCA_{3Lx} - UCA_{1Lz}UCA_{2Lx}UCA_{3Ly} + UCA_{1Lz}UCA_{2Ly}UCA_{3Lx} + \\ \dfrac{\left(CP_{Lx} + CP_{Rx}\right)\begin{pmatrix} UCA_{1Ly}UCA_{2Lz} - UCA_{1Ly}UCA_{3Lz} - UCA_{1Lz}UCA_{2Ly} + \\ UCA_{1Lz}UCA_{3Ly} + UCA_{2Ly}UCA_{3Lz} - UCA_{2Lz}UCA_{3Ly} \end{pmatrix}}{2} \end{pmatrix}}{UCA_{1Lx}UCA_{2Lz} - UCA_{1Lx}UCA_{3Lz} - UCA_{1Lz}UCA_{2Lx} + UCA_{1Lz}UCA_{3Lx} + UCA_{2Lx}UCA_{3Lz} - UCA_{2Lz}UCA_{3Lx}}, \quad \textbf{(108)}$$

Same expressions apply to $LCA_{1Rintrs}$, $LCA_{2Rintrs}$, $UCA_{1Rintrs}$ and $UCA_{2Rintrs}$ by changing index L to R.

Now that all parameters of UCA and LCA intersection lines are known, slopes are checked for equality. If percentage difference between values is smaller than previously defined *PRECISION*, a line with LCA intersection line slope is defined through *CP* point. The process of checking parallelism of LCA and UCA is done for both left and right side.

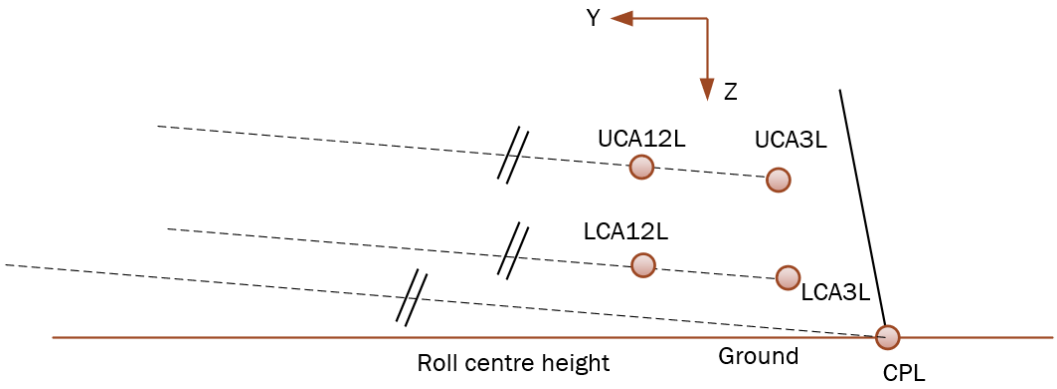

Figure 18 Instantaneous centre of rotation for parallel UCA and LCA

If the lines are not parallel, then they are intersected resulting in an instantaneous rotation point. Intersection is obtained by solving a system of two equations and two unknowns for both the left and the right side:

$$IC_{Ly} = \frac{a_{LCALintrs} b_{UCALintrs} - a_{UCALintrs} b_{LCALintrs}}{a_{LCALintrs} - a_{UCALintrs}}, \tag{109}$$

$$IC_{Lz} = \frac{-b_{LCALintrs} + b_{UCALintrs}}{a_{LCALintrs} - a_{UCALintrs}}, \tag{110}$$

$$IC_{Ry} = \frac{a_{LCARintrs} b_{UCARintrs} - a_{UCARintrs} b_{LCARintrs}}{a_{LCARintes} - a_{UCARintrs}}, \tag{111}$$

$$IC_{Rz} = \frac{-b_{LCARintrs} + b_{UCARintrs}}{a_{LCARintrs} - a_{UCARintrs}}. \tag{112}$$

Finally, coordinates of left and right instantaneous centres in case of unparallel LCA and UCA lines are:

$$IC_L = \left(\frac{CP_{xR} + CP_{xL}}{2}, IC_{Ly}, IC_{Lz} \right), \tag{113}$$

$$IC_R = \left(\frac{CP_{xR} + CP_{xL}}{2}, IC_{Ry}, IC_{Rz} \right). \tag{114}$$

When the lines defined by instantaneous centres are known it is possible to calculate roll centre height as the distance between ground plane and intersection point of the two lines defined by *CP*s and instantaneous centres.

Lines defined by instantaneous centres have the following form:

$$y_{ICL} = a_{ICL} \cdot z_{ICL} + b_{ICL}, \tag{115}$$

$$y_{ICR} = a_{ICR} \cdot z_{ICR} + b_{ICR}. \quad \textbf{(116)}$$

In case of unparallel LCA and UCA lines, that is instantaneous line going through *CP* and *IC*, slope and segment on Y axis values have the following expressions:

$$a_{ICL} = \frac{IC_{Ly} - CP_{Ly}}{IC_{Lz} - CP_{Lz}}, \quad \textbf{(117)}$$

$$b_{ICL} = -CP_{Lz} \cdot \frac{IC_{Ly} - CP_{Ly}}{IC_{Lz} - CP_{Lz}} + CP_{Ly}. \quad \textbf{(118)}$$

$$a_{ICR} = \frac{IC_{Ry} - CP_{Ry}}{IC_{Rz} - CP_{Rz}}, \quad \textbf{(119)}$$

$$b_{ICR} = -CP_{Rz} \cdot \frac{IC_{Ry} - CP_{Ry}}{IC_{Rz} - CP_{Rz}} + CP_{Ry}. \quad \textbf{(120)}$$

In case of parallel LCA and UCA lines, that is, instantaneous centre line going through *CP* and is parallel to LCA (UCA), the values are:

$$a_{ICL} = a_{LCAL}, \quad \textbf{(121)}$$

$$b_{ICL} = CP_{Ly} - a_{LCAL} CP_{Lz}, \quad \textbf{(122)}$$

$$a_{ICR} = a_{LCAR}, \quad \textbf{(123)}$$

$$b_{ICR} = CP_{Ry} - a_{LCAR} CP_{Rz}, \quad \textbf{(124)}$$

Now that parameters for instantaneous centre lines are known they are intersected:

$$y_{ICL} = y_{ICR} = RC_y, \quad \textbf{(125)}$$

$$a_{ICR} \cdot RC_z + b_{ICR} = a_{ICL} \cdot RC_z + b_{ICL}, \quad \textbf{(126)}$$

$$RC_z = \frac{b_{ICL} - b_{ICR}}{a_{ICR} - a_{ICL}}, \quad \textbf{(127)}$$

$$RC_y = a_{ICL} \cdot \frac{b_{ICL} - b_{ICR}}{a_{ICR} - a_{ICL}} + b_{ICL}. \quad \textbf{(128)}$$

Since roll centre height is being estimated in a plane parallel to YZ global plane, ground plane is defined as a line and distance between the roll centre point and ground line is:

$$\text{RC height} = \frac{\left(CP_{Ry} - CP_{Ly}\right)\left(CP_{Lz} - RC_z\right) - \left(CP_{Ly} - RC_y\right)\left(CP_{Rz} - CP_{Lz}\right)}{\sqrt{\left(CP_{Rz} - CP_{Lz}\right)^2 + \left(CP_{Ry} - CP_{Ly}\right)^2}}. \quad \textbf{(129)}$$

When RC point is under ground the distance has a negative value otherwise positive.

2.2.5. Caster trail

Caster trail is generally defined as the distance between intersection of a line defined by UCA_3 and LCA_3 with the ground and the wheel axis line projected to ground.

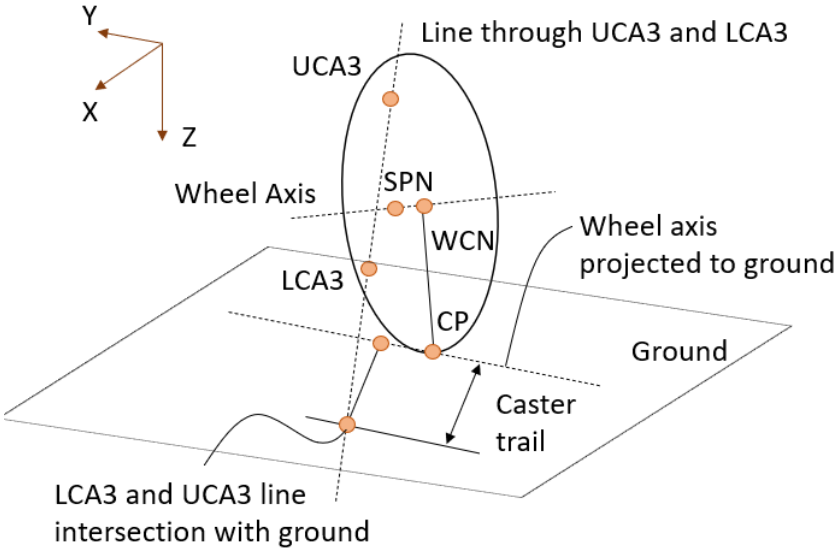

Figure 19 Caster trail schematic

Equation od ground plane defined by left and right *CP* is:

$$A\left(x-CP_{Lx}\right)+B\left(y-CP_{Ly}\right)+C\left(z-CP_{Lz}\right)=0. \tag{130}$$

Where *A*, *B* and *C* parameters are plane normal:

$$A=0, \tag{131}$$

$$B=-CP_{Lx}\left(-CP_{Lz}+CP_{Rz}\right), \tag{132}$$

$$C=CP_{Lx}\left(-CP_{Ly}+CP_{Ry}\right). \tag{133}$$

Wheel axis is defined by *WCN* and *SPN* which are projected to ground plane:

$$WCN_{pr}=\begin{pmatrix}\dfrac{A\left(ACP_{Lx}-AWCN_x+BCP_{Ly}-BWCN_y+CCP_{Lz}-CWCN_z\right)}{A^2+B^2+C^2}+WCN_x,\\ \dfrac{B\left(ACP_{Lx}-AWCN_x+BCP_{Ly}-BWCN_y+CCP_{Lz}-CWCN_z\right)}{A^2+B^2+C^2}+WCN_y,\\ \dfrac{C\left(ACP_{Lx}-AWCN_x+BCP_{Ly}-BWCN_y+CCP_{Lz}-CWCN_z\right)}{A^2+B^2+C^2}+WCN_z\end{pmatrix}, \tag{134}$$

$$SPN_{pr} = \begin{pmatrix} \frac{A\left(ACP_{Lx} - ASPN_x + BCP_{Ly} - BSPN_y + CCP_{Lz} - CSPN_z\right)}{A^2 + B^2 + C^2} + SPN_x, \\ \frac{B\left(ACP_{Lx} - ASPN_x + BCP_{Ly} - BSPN_y + CCP_{Lz} - CSPN_z\right)}{A^2 + B^2 + C^2} + SPN_y, \\ \frac{C\left(ACP_{Lx} - ASPN_x + BCP_{Ly} - BSPN_y + CCP_{Lz} - CSPN_z\right)}{A^2 + B^2 + C^2} + SPN_z \end{pmatrix}. \quad \textbf{(135)}$$

Intersection of ground plane and LCA_3UCA_3 line is given by the following relation:

$$LCA_3UCA_{3intersection} = \begin{pmatrix} LCA_{3x} - \frac{\left(LCA_{3x} - UCA_{3x}\right)\begin{pmatrix} -ACP_{Lx} + ALCA_{3x} - BCP_{Ly} \\ +BLCA_{3y} - CCP_{Lz} + CLCA_{3z} \end{pmatrix}}{ALCA_{3x} - AUCA_{3x} + BLCA_{3y} - BUCA_{3y} + CLCA_{3z} - CUCA_{3z}}, \\ LCA_{3y} - \frac{\left(LCA_{3y} - UCA_{3y}\right)\begin{pmatrix} -ACP_{Lx} + ALCA_{3x} - BCP_{Ly} \\ +BLCA_{3y} - CCP_{Lz} + CLCA_{3z} \end{pmatrix}}{ALCA_{3x} - AUCA_{3x} + BLCA_{3y} - BUCA_{3y} + CLCA_{3z} - CUCA_{3z}}, \\ LCA_{3z} - \frac{\left(LCA_{3z} - UCA_{3z}\right)\begin{pmatrix} -ACP_{Lx} + ALCA_{3x} - BCP_{Ly} \\ +BLCA_{3y} - CCP_{Lz} + CLCA_{3z} \end{pmatrix}}{ALCA_{3x} - AUCA_{3x} + BLCA_{3y} - BUCA_{3y} + CLCA_{3z} - CUCA_{3z}} \end{pmatrix} \quad \textbf{(136)}$$

Caster trail is given by the following relation representing distance between LCA_3UCA_{3int} point and line defined by projections of *WCN* and *SPN*:

$$\text{Caster trail} = \frac{\left(LCA_3UCA_{3int} - SPN_{pr}\right) \times \left(LCA_3UCA_{3int} - WCN_{pr}\right)}{\left|WCN_{pr} - SPN_{pr}\right|}. \quad \textbf{(137)}$$

When LCA_3UCA_3 intersection point is in front of *CP* then caster trail has positive value, otherwise negative.

2.2.6. Kingpin angle

Kingpin angle is defined as the smaller angle that the line seen from front of the chassis connecting UCA_3 and LCA_3 closes with the XZ plane of the chassis.

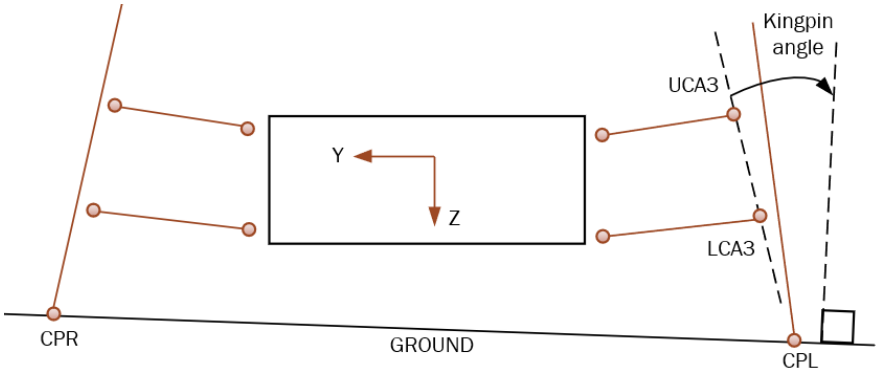

Figure 20 Kingpin angle measuring

It is measured between ground normal and a line going through projection of LCA_3 and UCA_3 points to a plane parallel with YZ global plane. Vector components are:

$$LCA3UCA3_{pr} = \left(0, -UCA3_y + LCA3_y, -UCA3_z + LCA3_z\right), \tag{138}$$

$$ground = \left(0, -CP_{Rz} + CP_{Lz}, CP_{Ry} - CP_{Ry}\right). \tag{139}$$

Kingpin angle is then:

$$\text{Kingipn angle} = \cos^{-1}\left(\frac{LCA3UCA3_{pr} \cdot ground}{\left|LCA3UCA3_{pr}\right| \cdot \left|ground\right|}\right). \tag{140}$$

If UCA_{3y} absolute value is lower than LCA_{3y} absolute value then kingpin angle has a positive value, otherwise negative.

2.2.7. Scrub radius

Scrub radius is defined as distance between point where LCA_3UCA_3 line intersects ground and *CP* when viewed from the front of the wheel. When CP_y absolute value is smaller than LCA_3UCA_3 intersection *y* absolute value then scrub radius has a positive value, otherwise negative.

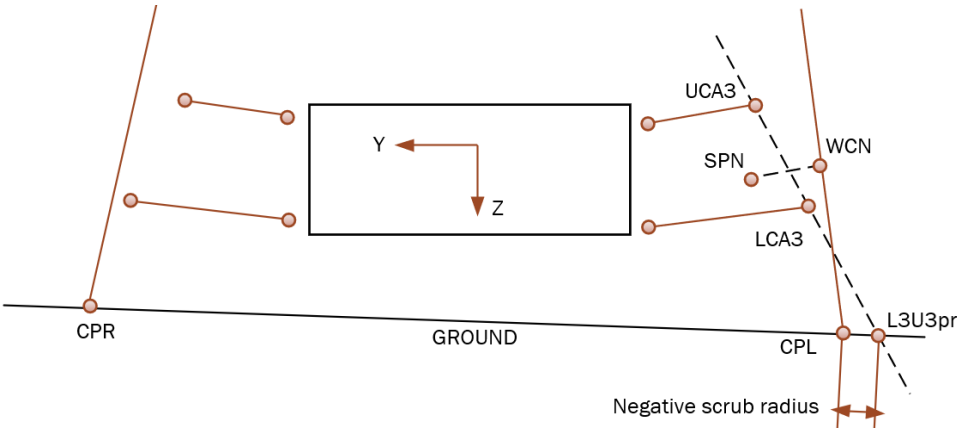

Figure 21 Scrub radius measurement

Mathematically, scrub radius is defined as distance of LCA_3UCA_{3pr} to plane defined by normal consisting of *WCN* and *SPN* projections to ground plane and *CP* as the point through which the plane passes.

LCA_3UCA_{3intrs}, WCN_{pr} and SPN_{pr} are known from Caster trail calculation. Only plane normal to WCN_{pr}-SPN_{pr} line going through CP_L point is left to define:

$$\begin{aligned}&\left(WCN_{prx}-SPN_{prx}\right)\left(x-CP_{Lx}\right)+\\&\left(WCN_{pry}-SPN_{pry}\right)\left(y-CP_{Ly}\right)+\\&\left(WCN_{prz}-SPN_{prz}\right)\left(z-CP_{Lz}\right)=0.\end{aligned} \quad \textbf{(141)}$$

Lastly, distance from $LCA_3UCA_{3interscetion}$ to newly defined plane is:

$$\text{Scrub radius} = \frac{\left|\begin{aligned}&\left(WCN_{prx}-SPN_{prx}\right)\left(L3U3_{prx}-CP_{Lx}\right)+\\&\left(WCN_{pry}-SPN_{pry}\right)\left(L3U3_{pry}-CP_{Ly}\right)+\\&\left(WCN_{prz}-SPN_{prz}\right)\left(L3U3_{prz}-CP_{Lz}\right)\end{aligned}\right|}{\sqrt{\left(WCN_{prx}-SPN_{prx}\right)^2+\left(WCN_{pry}-SPN_{pry}\right)^2+\left(WCN_{prz}-SPN_{prz}\right)^2}}. \quad \textbf{(142)}$$

2.2.8. Half-track change

Half-track change is difference between *y* value of *CP* in reference position and *y* value of *CP* in any other wheel position. When the *y* value in any other position is smaller than in reference, the value is negative and vice versa.

2.2.9. Wheelbase change

Wheelbase change is difference between *x* value of *CP* in reference position and *x* value of *CP* in any other wheel position. When the *x* value in any other position is smaller than in reference, the value is negative and vice versa.

2.2.10. Anti-features

All anti features have the same starting point for calculation, determination of wheel instantaneous centre of rotation from side view. Since calculation is based on a plane parallel to global XZ plane, everything will be translated to 2D scenario. First, intersection of planes defined by UCA and LCA with side view plane is done.

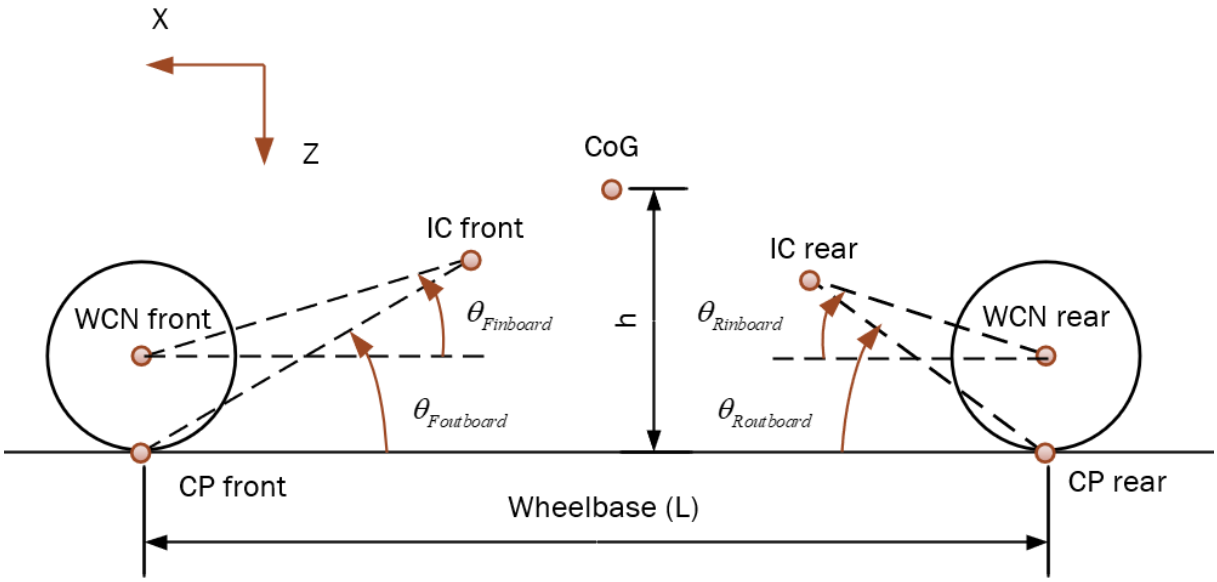

Figure 22 Anti features important parameters

If the brakes or drive is outboard then θ_{outboard} is used, otherwise θ_{inboard}.

Only defining point and direction vector will be given for each line:

$$LCA_{\text{int }Pt} = \left(LCA_{\text{int }Ptx}, LCA_{\text{int }Ptz}\right) =$$

$$\begin{pmatrix} CP_y\left(\begin{matrix} LCA_{1x}LCA_{2z} - LCA_{1x}LCA_{3z} - LCA_{1z}LCA_{2x} + \\ LCA_{1z}LCA_{3x} + LCA_{2x}LCA_{3z} - LCA_{2z}LCA_{3x} \end{matrix}\right) + \\ \dfrac{\begin{matrix} LCA_{1x}LCA_{2y}LCA_{3z} - LCA_{1x}LCA_{2z}LCA_{3y} - LCA_{1y}LCA_{2x}LCA_{3z} + \\ LCA_{1y}LCA_{2z}LCA_{3x} + LCA_{1z}LCA_{2x}LCA_{3y} - LCA_{1z}LCA_{2y}LCA_{3x} \end{matrix}}{\begin{matrix} LCA_{1y}LCA_{2z} - LCA_{1y}LCA_{3z} - LCA_{1z}LCA_{2y} + \\ LCA_{1z}LCA_{3y} + LCA_{2y}LCA_{3z} - LCA_{2z}LCA_{3y} \end{matrix}}, \\ 0 \end{pmatrix} \tag{143}$$

$$LCA_{\text{int }Dir} = \frac{\left(LCA_{1y} - LCA_{2y}\right)\left(LCA_{1z} - LCA_{3z}\right) - \left(LCA_{1y} - LCA_{3y}\right)\left(LCA_{1z} - LCA_{2z}\right)}{-\left(LCA_{1x} - LCA_{2x}\right)\left(LCA_{1y} - LCA_{3y}\right) + \left(LCA_{1x} - LCA_{3x}\right)\left(LCA_{1y} - LCA_{2y}\right)}, \tag{144}$$

$$UCA_{\text{int }Pt} = \left(UCA_{\text{int }Ptx}, UCA_{\text{int }Ptz}\right) =$$

$$\begin{pmatrix} CP_y\left(\begin{matrix} UCA_{1x}UCA_{2z} - UCA_{1x}UCA_{3z} - UCA_{1z}UCA_{2x} + \\ UCA_{1z}UCA_{3x} + UCA_{2x}UCA_{3z} - UCA_{2z}UCA_{3x} \end{matrix}\right) + \\ \dfrac{\begin{matrix} UCA_{1x}UCA_{2y}UCA_{3z} - UCA_{1x}UCA_{2z}UCA_{3y} - UCA_{1y}UCA_{2x}UCA_{3z} + \\ UCA_{1y}UCA_{2z}UCA_{3x} + UCA_{1z}UCA_{2x}UCA_{3y} - UCA_{1z}UCA_{2y}UCA_{3x} \end{matrix}}{\begin{matrix} UCA_{1y}UCA_{2z} - UCA_{1y}UCA_{3z} - UCA_{1z}UCA_{2y} + \\ UCA_{1z}UCA_{3y} + UCA_{2y}UCA_{3z} - UCA_{2z}UCA_{3y} \end{matrix}}, \\ 0 \end{pmatrix} \tag{145}$$

$$UCA_{\text{int }Dir} = \frac{\left(UCA_{1y}-UCA_{2y}\right)\left(UCA_{1z}-UCA_{3z}\right)-\left(UCA_{1y}-UCA_{3y}\right)\left(UCA_{1z}-UCA_{2z}\right)}{-\left(UCA_{1x}-UCA_{2x}\right)\left(UCA_{1y}-UCA_{3y}\right)+\left(UCA_{1x}-UCA_{3x}\right)\left(UCA_{1y}-UCA_{2y}\right)}. \quad \textbf{(146)}$$

Resulting lines are also intersected and instantaneous centre of rotation point is obtained:

$$IC = \begin{pmatrix} \dfrac{LCA_{intDir}LCA_{intPtx}-LCA_{intPtz}-UCA_{intDir}UCA_{intPtx}+UCA_{intPtz}}{LCA_{intDir}-UCA_{intDir}}, \\ LCA_{intDir}\left(LCA_{intPtx}UCA_{intDir}-LCA_{intPtz}-UCA_{intDir}UCA_{intPtx}+UCA_{intPtz}\right)+ \\ \dfrac{LCA_{intPtz}\left(LCA_{intDir}-UCA_{intDir}\right)}{LCA_{intDir}-UCA_{intDir}} \end{pmatrix}. \quad \textbf{(147)}$$

Next, whether the drive and brakes are outboard, or inboard are θ_F and θ_R calculated:

$$\tan\theta_{inboard} = \frac{IC_z - WCN_z}{IC_x - WCN_x}, \quad \textbf{(148)}$$

$$\tan\theta_{outboard} = \frac{IC_z - CP_z}{IC_x - CP_x}. \quad \textbf{(149)}$$

In all anti features there is a special case if a corresponding bias is zero, then the value of that anti feature is also zero. Otherwise, this would in case of inboard placed components result in infinity since there would be zero division.

Anti-features are signed in accordance with the figure below.

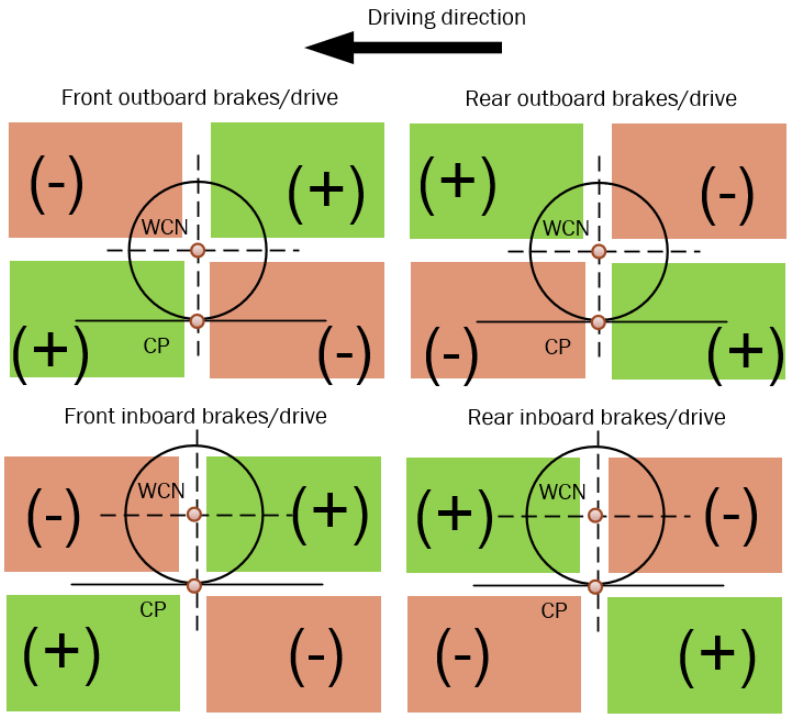

Figure 23 Signs of Suspension Anti Features

To be able to calculate all anti features the following parameters must be known:

- Wheelbase (l [mm])
- Centre of gravity height (h [mm])
- Front suspension:
    - Brake bias front- 0…1 (1- brake bias rear)
    - Brake's position- outboard or inboard
    - Drive bias front- 0...1 (1- drive bias rear)
    - Drive position- outboard or inboard
- Rear suspension:
    - Brake bias rear- 0…1 (1- brake bias front)
    - Brake's position- outboard or inboard
    - Drive bias rear- 0…1 (1- drive bias front)
    - Drive position- outboard or inboard

2.2.10.1. Anti-rise

Front suspension resistance to moving front end of the chassis up during acceleration in case of front wheel drive.

In case of outboard drive (usually an electromotor is inside the wheel) formula is:

$$\%\text{Anti rise} = \frac{\tan\theta_{outboardF}}{h} l \cdot (\text{front drive bias}) \cdot 100\%. \quad \textbf{(150)}$$

In case of inboard drive (use of drive shafts to transfer power to wheels) formula is:

$$\%\text{Anti rise} = \frac{\tan\theta_{inboardF}}{h \cdot (\text{front drive bias})} l \cdot 100\%. \quad \textbf{(151)}$$

2.2.10.2. Anti-squat

Rear suspension resistance to moving rear end of the chassis down during acceleration in case of rear wheel drive.

In case of outboard drive (usually an electromotor is inside the wheel) formula is:

$$\%\text{Anti squat} = \frac{\tan\theta_{outboardR}}{h} l \cdot (\text{rear drive bias}) \cdot 100\%. \quad \textbf{(152)}$$

In case of inboard drive (that is, use of drive shafts to transfer power to wheels) formula is:

$$\%\text{Anti squat} = \frac{\tan\theta_{inboardR}}{h \cdot (\text{rear drive bias})} l \cdot 100\%. \quad \textbf{(153)}$$

2.2.10.3. Anti-dive

Front suspension resistance to moving down front end of the chassis during braking.

In case of outboard brakes (brake discs mounted to wheel hubs) formula is:

$$\%\text{Anti dive} = \frac{\tan\theta_{outboardF}}{h} l \cdot (\text{front brake bias}) \cdot 100\%. \quad \textbf{(154)}$$

In case of inboard brakes (use of drive shafts to transfer braking to wheels) formula is:

$$\%\text{Anti dive} = \frac{\tan\theta_{inboardR}}{h \cdot (\text{front brake bias})} l \cdot 100\%. \quad \textbf{(155)}$$

2.2.10.4. Anti-lift

Rear suspension resistance to moving up rear end of the chassis during braking.

In case of outboard brakes (brake discs mounted to wheel hubs) formula is:

$$\%\text{Anti lift} = \frac{\tan\theta_{outboardR}}{h} l \cdot (\text{rear brake bias}) \cdot 100\%. \quad \textbf{(156)}$$

In case of inboard brakes (use of drive shafts to transfer braking to wheels) formula is:

$$\%\text{Anti lift} = \frac{\tan\theta_{inboardR}}{h \cdot (\text{rear brake bias})} l \cdot 100\%. \quad \textbf{(157)}$$

2.2.11. Inside wheel free space

Solution given by the optimisation might have the best kinematics, but if the hardpoints don't fit inside the wheel it is useless. Therefore, two more suspension characteristics not related to kinematics but to component design are introduced.

2.2.11.1. Free radius

LCA_3, UCA_3 and TR_2 are checked how far are they from the wheel axis. This is done by calculating distance of each mentioned point from the wheel axis defined by *WCN* and *SPN*:

$$\text{Distance} = \frac{\overrightarrow{(PT - WCN)} \times \overrightarrow{(PT - SPN)}}{\left|\overrightarrow{(PT - WCN)} \times \overrightarrow{(PT - SPN)}\right| \cdot \left|\overrightarrow{(WCN - SPN)}\right|}. \quad \textbf{(158)}$$

2.2.11.2. Distance from WCN along wheel axis

Another important aspect is how deep inside the wheel are 3 before mentioned points. This is important because of brake discs placement and steering capabilities.

To calculate this distance a plane is made perpendicular to wheel axis and going through *WCN*. Then signed distance between plane and point is calculated.

All points that are between the defined plane and centre of the vehicle have a negative value while those that are outside have positive.

For plane in form of linear equation *A*, *B*, *C*, and *D* parameters are defined as:

$$A = WCN_x - SPN_x, \tag{159}$$

$$B = WCN_y - SPN_y, \tag{160}$$

$$C = WCN_z - SPN_z, \tag{161}$$

$$D = -WCN_x \cdot A - WCN_y \cdot B - WCN_z \cdot C. \tag{162}$$

Finally, signed distance is

$$\text{Distance} = \frac{A \cdot PT_x + B \cdot PT_y + C \cdot PT_z + D}{\sqrt{A^2 + B^2 + C^2}}. \tag{163}$$

3. Quarter Suspension Optimisation Model

Finding suspension kinematics is a crucial process and is present from the very beginning of suspension design. However, as much as it is crucial it is also dull and time consuming. For this reason, there is a need to automate this process and one of possibilities in this automation is use of optimisation. In the following text quarter suspension optimisation model will be formulated.

3.1. Design variables

Although suspension consists of 10 hardpoints each of which has 3 coordinates, resulting in 30 parameters needed to fully define suspension, only 20 of those parameters are variables in optimisation model.

Table 3 Model parameters and variables

	x	*y*	*z*		*x*	*y*	*z*
LCA_1				UCA_3			
LCA_2				TR_1			
LCA_3				TR_2			
UCA_1				*WCN*			
UCA_2				*SPN*			

Fields marked red in table above are hardpoint coordinates related optimisation parameters while the rest are optimisation variables.

As can be seen in chapter describing derivation of suspension kinematics, control arms can be observed as beams rotating around an axis in space. To define any axis in space only 2 points are needed, and those 2 points can lay anywhere in space. If those 2 points are placed on 2 parallel planes, all axes that are of interest to this optimisation can still be defined. The ones that cannot be defined are parallel to these two planes. Finally, it is assumed that for a control arm first plane has a value x equal to the x value of first point of control arm and second plane equal to the second point x value.

With this simplification 4 variables are removed from optimisation problem.

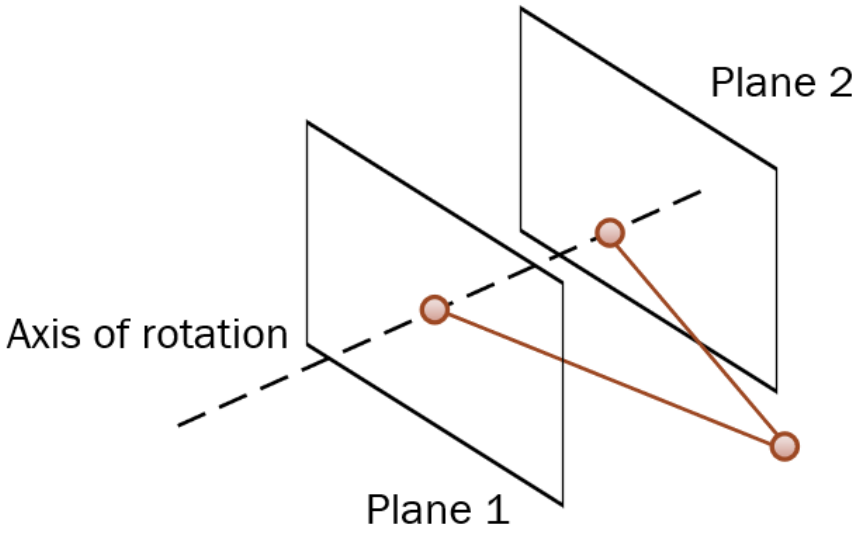

Figure 24 Axis of rotation defined by 2 arbitrary points

Since it is possible to get preferred Camber and toe angle in reference position just from wheel axis defined by WCN and SPN, WCN and SPN are also fully defined by the user before optimisation. With this the project space is reduced to 20 variables.

3.2. Objective function

During optimisation only certain suspension characteristics in certain wheel position are of interest. Wheel positions are defined for rolling movement of chassis, when one wheel goes up, the other goes down and vice versa. Furthermore, since it is possible to determine static Camber and toe angle values without optimisation, those values are of no interest. During one iteration, upper, lower and reference suspension positions are calculated and for each of those positions only some characteristics are calculated. As was said, optimisation is done with rolling motion of chassis assumed.

The following table shows which suspension characteristics are calculated in which suspension position.

Table 4 Suspension characteristics calculated with respect to wheel movement during optimisation

Suspension characteristics	Wheel position		
	Lower	Reference	Upper
Camber angle			
Toe angle			

Caster angle			
Roll centre height			
Caster trail			
Scrub radius			
Kingpin angle			
Anti-drive			
Anti-brake			
Half-track change			
Wheelbase change			
LCA_3 inside wheel free radius			
UCA_3 inside wheel free radius			
TR_2 inside wheel free radius			
LCA_3 distance to *WCN* along *WCN-SPN* axis			
UCA_3 distance to *WCN* along *WCN-SPN* axis			
TR_2 distance to *WCN* along *WCN-SPN* axis			

Constant parameters during optimisation are:

- Suspension position
- Wheel radius
- Wheelbase
- Centre of gravity height
- Drive bias
- Brake bias
- Drive position
- Brake's position
- Vertical movement from reference position

Objective function consists of multiple members and as such is multidimensional. Every member can be assigned a different significance value which results in a weight factor where the sum of all weight factors equals one.

Since all members have different sensitivity to changes, each of them has its own parameters such as target, peak width, and peak flatness values:

$$\text{Obj. func.} = 1 - w_i e^{-\frac{|x_i - \text{target}_i|^{k_i}}{m_i}} - w_{i+1} e^{-\frac{|x_{i+1} - \text{target}_{i+1}|^{k_{i+1}}}{m_{i+1}}} - \ldots - w_n e^{-\frac{|x_n - \text{target}_n|^{k_n}}{m_n}}, \quad \textbf{(164)}$$

with weight factor being calculated as:

$$w_i = \frac{s_i}{\sum_{j=1}^{n} s_j}, \quad \textbf{(165)}$$

and having the property

$$\sum_{i=1}^{n} w_i = 1. \quad \textbf{(166)}$$

The rest of parameters are:

m_i- arbitrarily chosen number greater than zero, determines width of "bell" around target value, greater than zero,

x_i- suspension characteristic variable,

$target_i$- wanted values for respective suspension characteristics,

k_i- value determining "flatness" of objective function around target value,

s_i- significance value determining weight factor value, must be greater than zero,

n- number of objective function members.

As can be seen in above equation, objective function's value will range from zero to one with the goal of optimisation being minimisation.

There are 21 criterions identified as possible members of objective function in optimisation and these variables correspond to constraints that are about to be described.

Because shape of objective function member is not easily understandable the following graphs are given. In them the behaviour of the function can be seen, however, clear connection between parameters and function shape cannot be established.

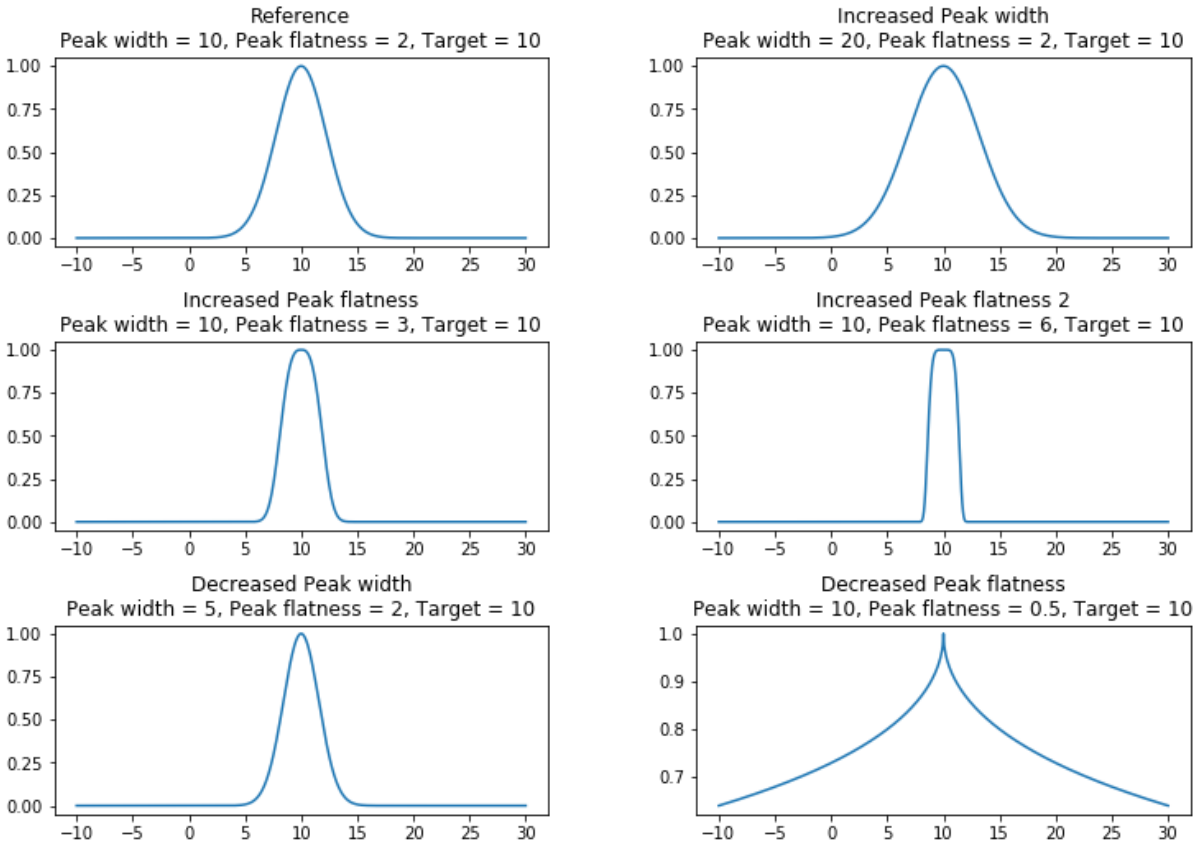

Figure 25 Shapes of objective function member

3.3. Constraints and bounds

Constraints are defined both for derived values and input variables. Some of them are in direct conflict with bounds because for different values they can provide different project space what may not be clear to the user at first glance. This will not make any problems during optimisation, but it is important to note possibility of redundance.

All constraints and bounds have defined a lower and higher value inside which the solution should reside. Because of SciPy limitations all constraints and bounds are defined in greater than or equal to zero form.

The following equation is rewritten in suitable expression:

$$g_{iLower} \leq g_i \leq g_{iUpper}. \tag{167}$$

Suitable expressions are:

$$g_i - g_{iLower} \geq 0, \tag{168}$$

$$-g_i + g_{iUpper} \geq 0, \tag{169}$$

All values for g_i, g_{iLower} and g_{iUpper} are given in the following table.

Table 5 Constraints

g_i	g_{iLower}	g_{iUpper}
Camber angle$_{Down\ position}$	Camber Angle$_{Down\ position\ Lower}$	Camber Angle$_{Down\ position\ Upper}$
Camber angle$_{Up\ position}$	Camber Angle$_{Up\ position\ Lower}$	Camber Angle$_{Up\ position\ Upper}$
Toe angle$_{Down\ position}$	Toe Angle$_{Down\ position\ Lower}$	Toe Angle$_{Down\ position\ Upper}$
Toe angle$_{Up\ position}$	Toe Angle$_{Up\ position\ Lower}$	Toe Angle$_{Up\ position\ Upper}$
Caster Angle	Caster Angle$_{Lower}$	Caster Angle$_{Upper}$
Roll centre height	Roll centre height$_{Lower}$	Roll centre height$_{Upper}$
Caster Trail	Caster Trail$_{Lower}$	Caster Trail$_{Upper}$
Scrub Radius	Scrub Radius$_{Lower}$	Scrub Radius$_{Upper}$
Kingpin Angle	Kingpin Angle$_{Lower}$	Kingpin Angle$_{Upper}$
Anti-drive	Anti-drive$_{Lower}$	Anti-drive$_{Upper}$
Anti-brake	Anti-brake$_{Lower}$	Anti-brake$_{Upper}$
Half-track change$_{Down\ position}$	Half-track change$_{Down\ position\ Lower}$	Half-track change$_{Down\ position\ Upper}$
Half-track change$_{Up\ position}$	Half-track change$_{Up\ position\ Lower}$	Half-track change$_{Up\ position\ Upper}$
Wheelbase change$_{Down\ position}$	Wheelbase change$_{Down\ position\ Lower}$	Wheelbase change$_{Down\ position\ Upper}$
Wheelbase change$_{Up\ position}$	Wheelbase change$_{Up\ position\ Lower}$	Wheelbase change$_{Up\ position\ Upper}$
LCA_3 in wheel radius	LCA_3 in wheel radius$_{Lower}$	LCA_3 in wheel radius$_{Upper}$
UCA_3 in wheel radius	UCA_3 in wheel radius$_{Lower}$	UCA_3 in wheel radius$_{Upper}$
TR_2 in wheel radius	TR_2 in wheel radius$_{Lower}$	TR_2 in wheel radius$_{Upper}$
LCA_3 distance to WCN	LCA_3 distance to WCN_{Lower}	LCA_3 distance to WCN_{Upper}
UCA_3 distance to WCN	UCA_3 distance to WCN_{Lower}	UCA_3 distance to WCN_{Upper}
TR_2 distance to WCN	TR_2 distance to WCN_{Lower}	TR_2 distance to WCN_{Upper}

Entries with the same colour in table above and the following table can be in conflict, but still, all are kept so the final user can adjust the application to suit his/her needs.

Depending on the optimisation algorithm implementation characteristics, design variables bounds (Table 6) may be given to optimisation directly as bounds, or if it is not possible, they can be given as constraints.

Table 6 Design variables bounds

g_i	g_{iLower}	g_{iUpper}
LCA_{1y}	$LCA_{1yLower}$	$LCA_{1yUpper}$
LCA_{1z}	$LCA_{1zLower}$	$LCA_{1zUpper}$
LCA_{2y}	$LCA_{2yLower}$	$LCA_{2yUpper}$
LCA_{2z}	$LCA_{2zLower}$	$LCA_{2zUpper}$
LCA_{3x}	$LCA_{3xLower}$	$LCA_{3xUpper}$
LCA_{3y}	$LCA_{3yLower}$	$LCA_{3yUpper}$
LCA_{3z}	$LCA_{3zLower}$	$LCA_{3zUpper}$
UCA_{1y}	$UCA_{1yLower}$	$UCA_{1yUpper}$
UCA_{1z}	$UCA_{1zLower}$	$UCA_{1zUpper}$
UCA_{2y}	$UCA_{2yLower}$	$UCA_{2yUpper}$
UCA_{2z}	$UCA_{2zLower}$	$UCA_{2zUpper}$
UCA_{3x}	$UCA_{3xLower}$	$UCA_{3xUpper}$
UCA_{3y}	$UCA_{3yLower}$	$UCA_{3yUpper}$
UCA_{3z}	$UCA_{3xLower}$	$UCA_{3xUpper}$
TR_{1x}	$TR_{1xLower}$	$TR_{1xUpper}$
TR_{1y}	$TR_{1yLower}$	$TR_{1yUpper}$
TR_{1z}	$TR_{1xLower}$	$TR_{1xUpper}$
TR_{2x}	$TR_{2xLower}$	$TR_{2xUpper}$
TR_{2y}	$TR_{2yLower}$	$TR_{2yUpper}$
TR_{2z}	$TR_{2zLower}$	$TR_{2zUpper}$

3.4. Optimization method

Finding of optimal kinematics in this master thesis has been formulated as a single-objective optimisation problem. Since it is featured by constraints and is without a function to describe variable gradients, Constrained Optimization by Linear Approximation (COBYLA) method is used.

It is a numerical optimization method for constrained problems where the derivative of the objective function is not known. It works by iteratively approximating the actual constrained optimization problem with linear programming problems. During an iteration, an approximating linear programming problem is solved to obtain a candidate for the optimal solution. The candidate solution is evaluated using the original objective and constraint functions, yielding a new data point in the optimization space. This information is used to improve the approximating linear programming problem used for the next iteration of the algorithm. When the solution cannot be improved anymore, the step size is reduced, refining the search. When the step size becomes sufficiently small, the algorithm finishes [2].

4. Software Application Formula Student Suspension Design

Application is intended to be used by anyone who wishes, free of any obligations. Source code is also freely available on GitHub so it can be adapted to suit one's needs.

Application consists of three modules:

- Suspension kinematics and characteristics calculator (C++)
- Optimisation module (Python)
- Graphical user interface (GUI) (C#)

Relation between these 3 modules is given in the following figure.

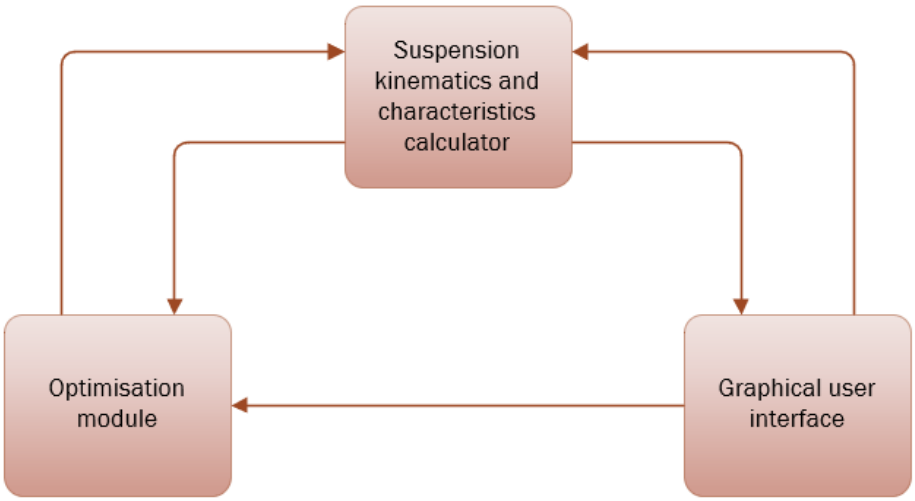

Figure 26 Relations between application's modules

Modules are communicating by sending arrays of data for suspension hardpoints, characteristics and setup. GUI and optimisation modules have one way communication because optimisation module takes parameters from GUI and when it is done, it outputs a csv file with results.

User can then easily study this csv file in applications such as Excel.

4.1. Suspension Kinematics and Characteristics

Implementation of mathematical expressions is in order as were the equations in Kinematics chapter written. It is important to conveniently determine input and output parameters from the very beginning since the whole programme will depend on it. Since C++ is chosen as the backend language because of its speed it is also chosen to use already available mathematical package, Eigen, a C++ library for linear algebra: matrices, vectors, numerical solvers, and related algorithms.

Two separate methods, both based on the same class for calculating all suspension related data are exposed, one for optimisation module and the other for GUI.

Input to C++ module is given in the following table. Rows marked green are also modified inside C++ module and as such are also the output. Rows marked red are inputs only in case of optimisation, rows marked blue are only for GUI module and the rest are inputs for both optimisation and GUI module.

Table 7 Input parameters for C++ module

Parameters came from module	Parameters modified in C++	Input parameters
both	-	Pointer to suspension hardpoints array in optimisation module
both	-	Suspension position
both	-	Wheel radius
both	-	Wheelbase
both	-	Centre of gravity height
both	-	Drive bias
both	-	Brake bias
both	-	Drive position
both	-	Brake's position
both	-	Vertical movement
optimisation	-	Pointer to objective function width parameters array
optimisation	-	Pointer to objective function flatness parameters array
both	-	Steering rack movement (for optimisation set to 0)
both	-	Number of vertical increments (for optimisation set to 1)
both	-	Number of steering increments (for optimisation set to 0)
both	-	Precision
optimisation	-	Pointer to objective function target values array
optimisation	-	Pointer to objective function weight factors array
both	modified	Objective function result as reference

both	modified	Pointer to output characteristics array
GUI	modified	Pointer to moved hardpoints array

4.2. Optimisation module

Optimisation is started from GUI after setting all parameters as preferred. Parameters are then passed through command prompt to python script which then starts the optimisation. Since COBYLA method works by finding a local objective function minimum, multiple optimisations with different initial positions are done so the user will have multiple solutions with some of them being good enough to use for further suspension development. Initial positions are picked from a uniformly distributed continuous set of values defined by hardpoints limits.

Parameters passed to optimisation module are listed previously in tables 4 and 5. Apart from those 2 parameters, two more optimisation module specific parameters are needed, number of processes to use and optimisation duration.

If multiple processes are used there is more strain on the hardware, however, more results can be obtained in less time. Optimisation duration is needed because it is possible that an impossible combination of parameters has been set up. If the optimisation was developed to stop after a certain objective function value has been reached, this might mean that optimisation would never stop and would have to be manually terminated. Another reason why it would not be ideal to run the optimisation session only until the required objective function value is obtained is because all parameters that have been set up are highly subjective. This means that solution with a objective function value of 0.1 might be better than a solution with value of 0.05. Since multiple optimisations are running and each of them usually needs at most 0.2 seconds to complete there is no fear that a lengthy optimisation might be interrupted and time would be wasted when terminating optimisations module by time expiration.

Upon finishing optimisation session, results are written to csv file. Each entry consists of the following:

- Objective function value
- Optimised hardpoints
- Suspension characteristics values, as could be seen in table 2
- Duration of this instance optimisation

It is then up to the user to sort and pick out a fitting solution and input it back to application for further observation.

There are few reasons why Python was chosen as the programming language in optimisation module. First is because of SciPy, a well-known and used library in academic circles that contains multiple optimisation algorithms with one of them being COBYLA. Second is that users can easily experiment with Python and implement even more advanced methods for finding the optimal kinematics, such as Reinforcement Learning.

4.3. Graphical User Interface in C#

Primary goal of this application is speeding up suspension kinematics iteration and this is done with the help of optimisation module. However, optimisation module requires setting up quite a few parameters. In case of a console application this would be a dealbreaker for a great deal of users which is why it was decided that a graphical user interface is a must.

GUI is divided in 3 sections:

1. Optimisation parameters
2. Currently observed parameters
3. Currently observed suspension visualization and characteristics

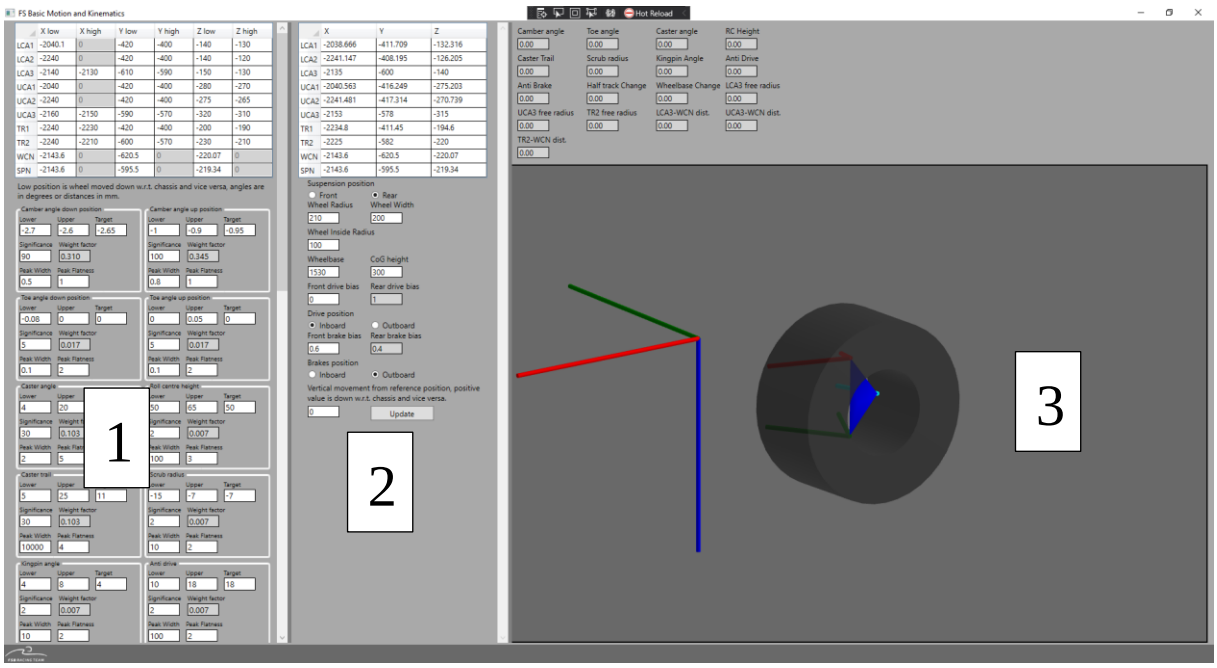

Figure 27 Application Main Window

Optimisation region comes with predefined values for suspension characteristics that have been deemed as the most suitable. Hardpoint limits are arranged in a table and since some coordinates are constant during optimisation, they have only one value associated with them.

	X low	X high	Y low	Y high	Z low	Z high
LCA1	-2040.1	0	-420	-400	-140	-130
LCA2	-2240	0	-420	-400	-140	-120
LCA3	-2140	-2130	-610	-590	-150	-130
UCA1	-2040	0	-420	-400	-280	-270
UCA2	-2240	0	-420	-400	-275	-265
UCA3	-2160	-2150	-590	-570	-320	-310
TR1	-2240	-2230	-420	-400	-200	-190
TR2	-2240	-2210	-600	-570	-230	-210
WCN	-2143.6	0	-620.5	0	-220.07	0
SPN	-2143.6	0	-595.5	0	-219.34	0

Figure 28 Hardpoint limits in GUI

After setting the hardpoints limits comes setting the suspension characteristics. Each of characteristics has 7 values with 6 of them being editable. Lower and upper textboxes set the boundaries for constraints inside which the solution should reside. Target is the value toward which objective function should gravitate. Significance is a factor which determines weight factor of current suspension characteristic. It must be greater than 0. If it is not relevant which value a characteristic should have, but should only satisfy the limits, significance can be set to zero.

As was described in Optimisation chapter, based on sum of all characteristics significances a weight factor is calculated. Sum of all weight factors equals one.

Finally, the two values defining suspension characteristic objective function, peak width, and flatness, are given.

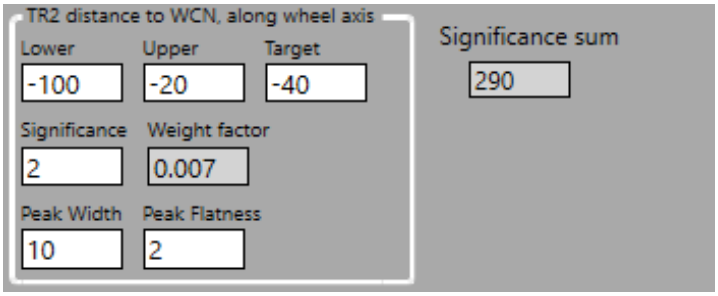

Figure 29 Suspension characteristic GUI parameters and Significance Sum

Basic suspension settings are left to define. These are parameters that are decided even before searching for optimal kinematics. Since only a quarter of suspension is being optimised it is needed to define is it front or rear, wheelbase, etc.

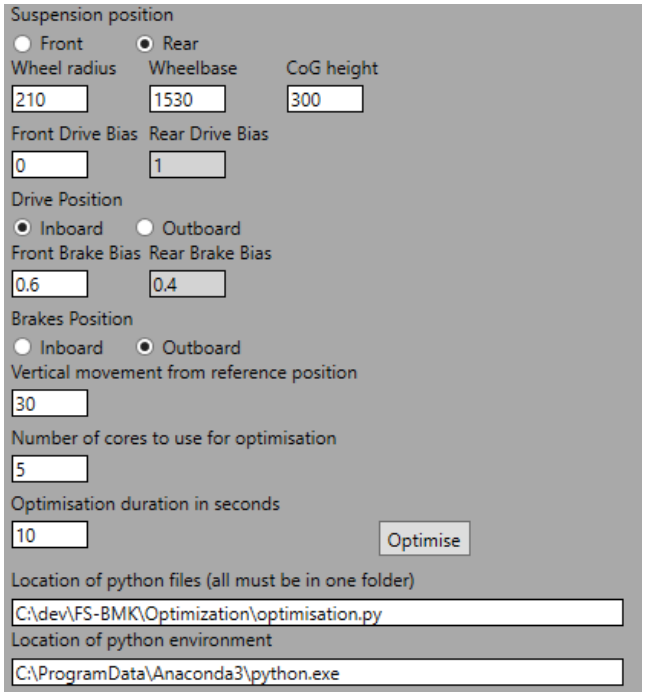

Figure 30 General suspension parameters

After the results from optimisation are obtained, they are entered to current suspension region. There are also present general suspension parameters which should be kept the same as in optimisation region.

Upon entering hardpoints the update button displays the new suspension configuration. By changing the vertical movement value, suspension movement can be observed and suspension characteristics for various wheel positions in case of rolling movement of the chassis. In the background a coordinate system is drawn with the red rod representing X, green Y and blue Z axis.

By changing the inside wheel radius value, the user can quickly see if there are collisions between control arms or tie rods with the wheel.

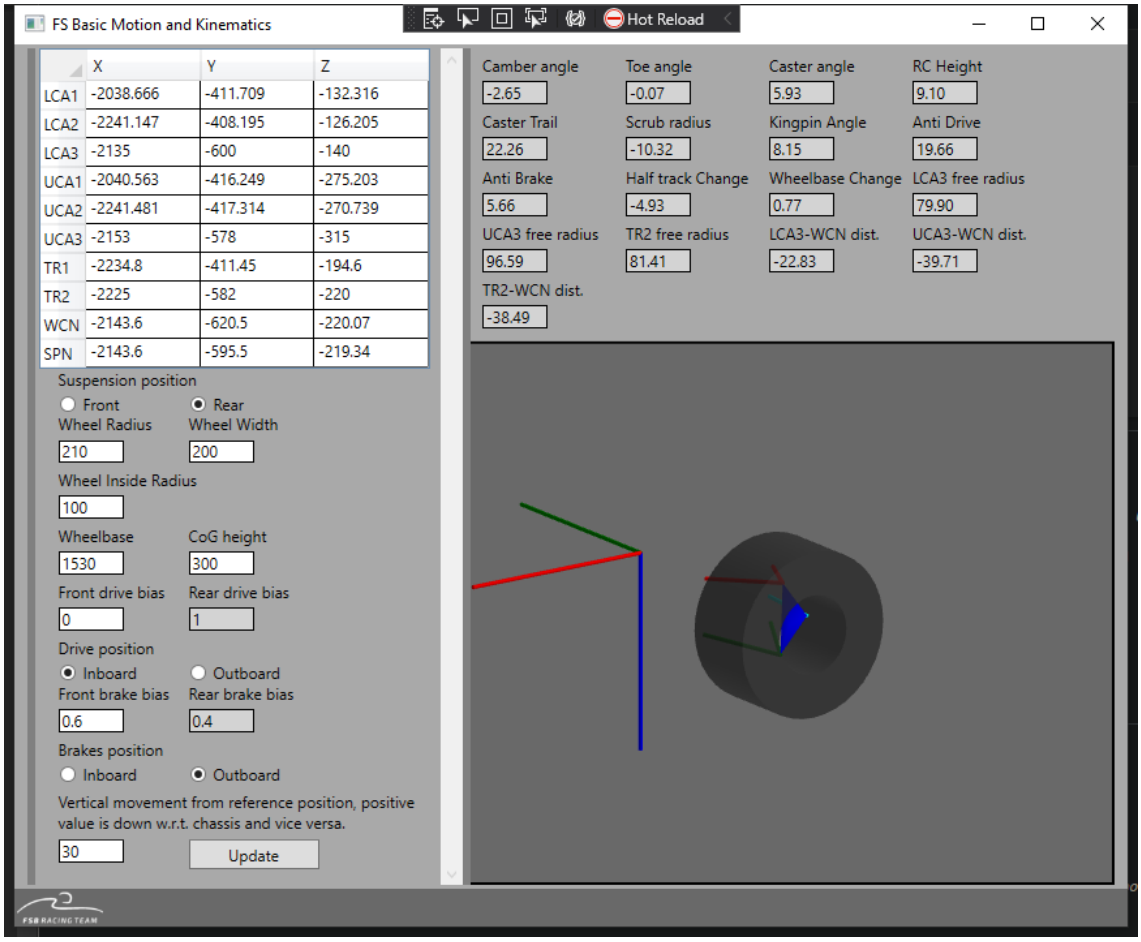

Figure 31 Current suspension parameters, characteristics, and visualization

5. Suspension Kinematics Design Example

Two types of optimisation, that is user scenarios have been recognized, when the user is starting from scratch and has only general suspension parameters, and the other when the user is improving previous year's vehicle's suspension. The first one is characterized by wide limits on all 20 variables, whereas the second one has tight limits and some of the variables may become parameters, e.g., chassis pick up points.

Examples are based on a rear of previous year's vehicle, VulpesR. Since numerical values of preferred suspension characteristics are known in advance and already implemented, it is certain that suspension optimisation shown here will not have an impossible combination of parameters.

Table 8 VulpesR hardpoints and optimisation suspension hardpoint limits

	x	*y*	*z*
LCA_1	-2038,67	-411,71	-132,32
LCA_2	-2241,15	-408,20	-126,21
LCA_3	-2135	-600	-140
UCA_1	-2040,56	-416,25	-275,20
UCA_2	-2241,48	-417,31	-270,74
UCA_3	-2153	-578	-315
TR_1	-2234,8	-411,45	-194,6
TR_2	-2225	-582	-220
WCN	-2143,6	-620,5	-220,07
SPN	-2143,6	-595,5	-219,34

Characteristic limits are the same both for wide and narrow limits optimisation. Only some values that will be mentioned later will change in narrow limits optimisation and those will represent values that the user wants improved.

Table 9 VulpesR suspension characteristics values and optimisation suspension characteristics limits and weight factors

	Value/Target	*Value lower*	*Value upper*	*Weight Factor*
Camber angle$_{\text{Down position}}$	-2,65	-2,7	-2,6	0,31
Camber angle$_{\text{Up position}}$	-0,98	-1	-0,9	0,345
Toe angle$_{\text{Down position}}$	-0,07	-0,07	0	0,017
Toe angle$_{\text{Up position}}$	0,04	0	0,04	0,017
Caster Angle	5,87	5	10	0,103
Roll centre height	55,99	50	60	0,007
Caster Trail	21,96	10	20	0,103
Scrub Radius	-10,3	-15	-5	0,007
Kingpin Angle	7,17	0	8	0,007
Anti-drive	16,91	10	20	0,007
Anti-brake	5,47	0	10	0,007
Half-track change$_{\text{Down position}}$	-4,93	-5	5	0,007
Half-track change$_{\text{Up position}}$	0,76	-5	5	0,007
Wheelbase change$_{\text{Down position}}$	0,77	-2	2	0,007
Wheelbase change$_{\text{Up position}}$	-0,74	-2	2	0,007
LCA_3 in wheel radius	79,9	80	90	0,007
UCA_3 in wheel radius	96,59	90	100	0,007
TR_2 in wheel radius	81,41	80	90	0,007
LCA_3 distance to *WCN*	-22,83	-30	-15	0,007
UCA_3 distance to *WCN*	-39,71	-50	-30	0,007
TR_2 distance to *WCN*	-38,49	-50	-30	0,007

Target values are set to VulpesR characteristics values and lastly objective function parameters, weight factors, width, and flatness, are adjusted as seems fit.

Peak width and flatness values do not give any obvious information on how the objective function will look like for a set of parameters. For this reason, it is necessary to plot objective functions and determine flatness and width values by visually studying objective function.

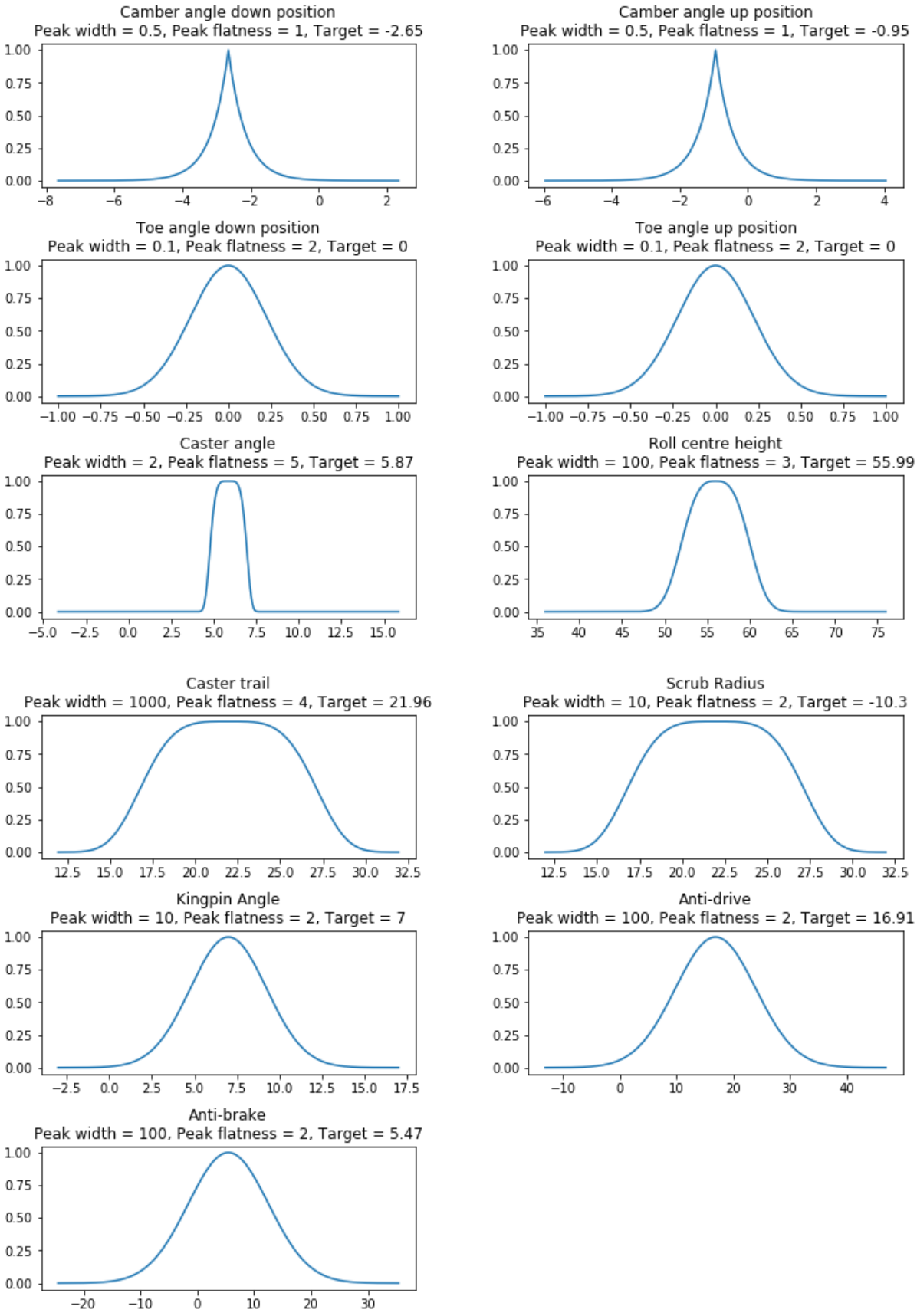

Figure 32 Optimisation Suspension Characteristics parameters 1

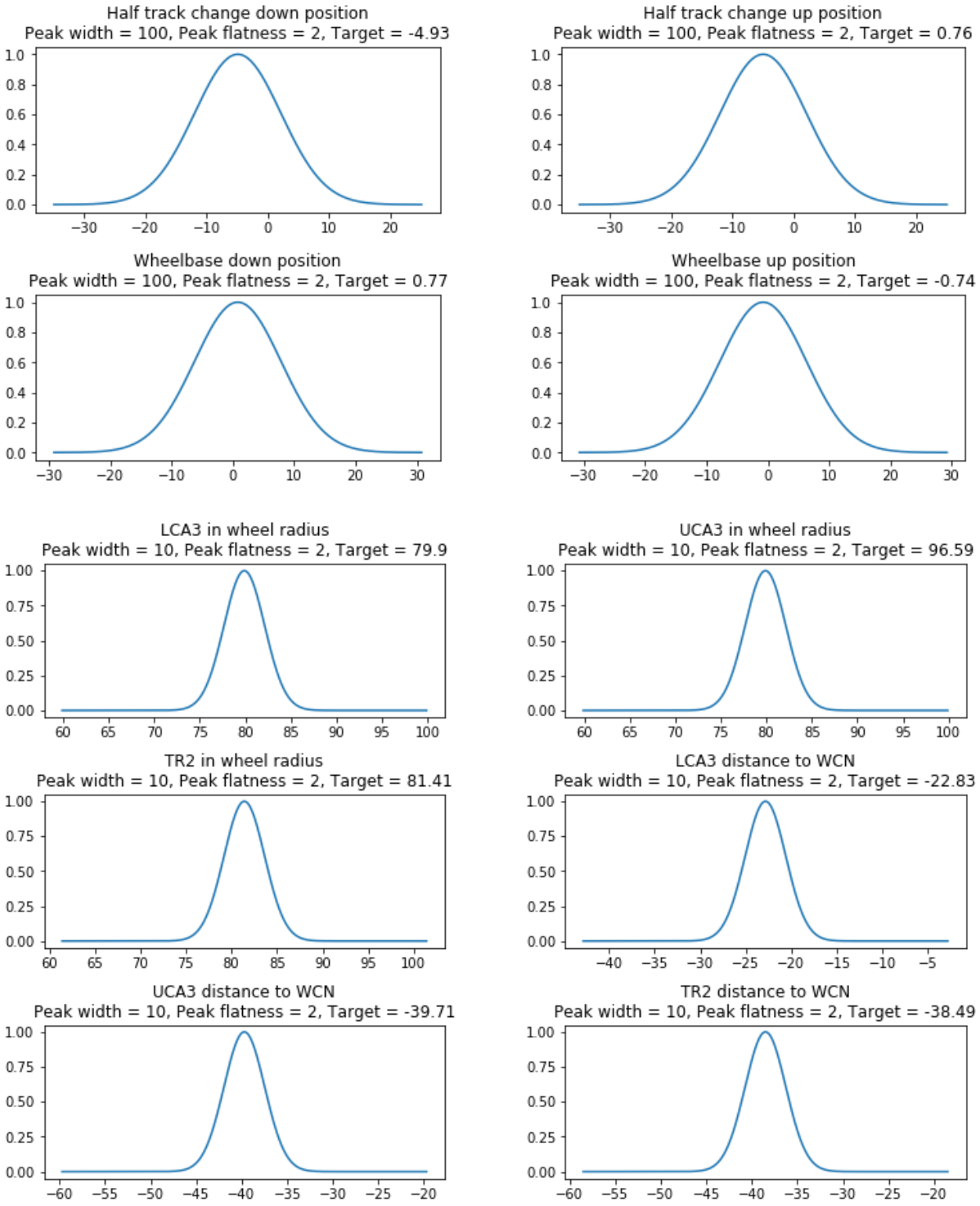

Figure 33 Optimisation Suspension Characteristics parameters 2

When setting the objective function individual characteristic parameters, one should have in mind that if actual width of objective function is smaller than the optimisation constraint limits it can happen that optimisation algorithm will not know in which direction should it move since objective function value becomes practically constant.

General suspension parameters will be the same in both simulations presented here:

- Rear suspension
- Wheel radius- 210 mm
- Wheelbase- 1530 mm
- Centre of gravity height- 300 mm
- Rear drive bias- 1
- Rear brake bias- 0,4
- Drivetrain position- inboard
- Brake position- outboard
- Vertical movement from reference position- 30 mm
- Hardware specific parameters:
    - Number of cores to use- 6
    - Optimisation duration- 600 seconds

5.1.1. Wide Limits Optimisation Problem

Hardpoint limits are determined subjectively with respect to VulpesR coordinates. Since uniform distribution is used to pick out initial suspension hardpoints, there is no bias toward the same solution as in VulpesR.

Table 10 Wide limits optimisation variable bounds

	x lower	*x upper*	*y lower*	*y upper*	*z lower*	*z upper*
LCA_1	-2038,67	0	-461,71	-361,71	-182,32	-82,32
LCA_2	-2241,15	0	-458,2	-358,2	-176,21	-76,21
LCA_3	-2185	-2085	-650	-550	-190	-90
UCA_1	-2040,56	0	-466,25	-366,25	-325,2	-225,2
UCA_2	-2241,48	0	-467,31	-367,31	-320,74	-220,74
UCA_3	-2203	-2103	-628	-528	-365	-265
TR_1	-2284,8	-2184,8	-461,45	-361,45	-244,6	-144,6
TR_2	-2275	-2175	-632	-532	-270	-170
WCN	-2143,6	0	-620,5	0	-220,07	0
SPN	-2143,6	0	-595,5	0	-219,34	0

A value of zero denotes that this coordinate is a parameter, not a variable during optimisation. It should be noted that the goal is not finding the exact same hardpoints, but a set of hardpoints that gives acceptable suspension characteristics.

Multiple solutions were obtained and the one with the following suspension hardpoints and characteristics was chosen.

Table 11 Optimised suspension hardpoints from wide limits optimisation

	x	*y*	*z*
LCA_1	-2038,67	-362,466	-134,284
LCA_2	-2241,15	-410,313	-133,096
LCA_3	-2141,72	-602,343	-149,736
UCA_1	-2040,56	-421,124	-265,416
UCA_2	-2241,48	-380,416	-249,538
UCA_3	-2163,94	-592,005	-305,091
TR_1	-2235,79	-392,911	-205,477
TR_2	-2236,13	-599,692	-242,185
WCN	-2143,6	-620,5	-220,07
SPN	-2143,6	-595,5	-219,34

Table 12 Optimised suspension characteristics from wide limits optimisation

Camber angle$_{\text{Down position}}$	-2,65	Half-track change$_{\text{Down position}}$	-4,37	Anti-brake	5,17
Camber angle$_{\text{Up position}}$	-0,91	Half-track change$_{\text{Up position}}$	0,64		
Toe angle$_{\text{Down position}}$	0,00	Wheelbase change$_{\text{Down position}}$	0,69		
Toe angle$_{\text{Up position}}$	0,00	Wheelbase change$_{\text{Up position}}$	-0,82		
Caster Angle	8,14	LCA_3 in wheel radius	69,8		
Roll centre height	50,00	UCA_3 in wheel radius	88,19		
Caster Trail	21,84	TR_2 in wheel radius	95,27		
Scrub Radius	-14,99	LCA_3 distance to *WCN*	-20,18		
Kingpin Angle	3,81	UCA_3 distance to *WCN*	-26,00		
Anti-drive	10	TR_2 distance to *WCN*	-20,15		

As can be seen, some suspension characteristics values have broken some of the constraints and, in those cases, it should be checked in the current suspension module of the application whether this is satisfactory or not. If not, it can be adjusted in the current suspension module or the solution can be entered to optimisation module by using its characteristics and hardpoints limits but with tighter limits as is shown in the following example.

5.1.2. Narrow Limits Optimisation Problem

Again, the hardpoint limits are determined with respect to VulpesR. However, some of the limits are made very small to simulate constants without the need to change the constraints and bounds in the optimisation model.

Table 13 Narrow limits optimisation variable bounds

	x lower	*x upper*	*y lower*	*y upper*	*z lower*	*z upper*
LCA_1	-2038,67	0	-411,81	-411,61	-132,42	-132,22
LCA_2	-2241,15	0	-408,3	-408,1	-126,31	-126,11
LCA_3	-2145	-2125	-610	-590	-150	-130
UCA_1	-2040,56	0	-416,35	-416,15	-275,3	-275,1
UCA_2	-2241,48	0	-417,41	-417,21	-270,84	-270,64
UCA_3	-2163	-2143	-588	-568	-325	-305
TR_1	-2234,9	-2234,7	-411,55	-411,35	-194,7	-194,5
TR_2	-2235	-2215	-592	-572	-230	-210
WCN	-2143,6	0	-620,5	0	-220,07	0
SPN	-2143,6	0	-595,5	0	-219,34	0

As was said, in this optimisation only camber angle parameters will be modified. As can be seen in the following table.

Table 14 Camber angle values for narrow optimisation

	Value/Target	*Value lower*	*Value upper*	*Weight Factor*
$\text{Camber angle}_{\text{Down position}}$	-2,2	-2,3	-2,2	0,31
$\text{Camber angle}_{\text{Up position}}$	-1,3	-1,5	-1,3	0,345

In this optimisation multiple solutions were found as well, however, they were more like each other than in the first case.

Narrow limits optimisation works as was expected, with narrower limits there are fewer possible solutions meaning that all the solutions are similar. However, violation of some constraints can still be seen.

Also, it is worth noting that a solution with desired camber angle values cannot be achieved for the entered parameters.

So, to conclude, as the overall suspension design is being finalized, more and more variables will become constants, and with each variable becoming a constant the user will be imposing new targets for optimisation and consequently will be ensuring that the final suspension is optimal.

Table 15 Optimised suspension hardpoints from narrow limits optimisation

	x	*y*	*z*
LCA_1	-2038,67	-411,619	-132,42
LCA_2	-2241,15	-408,3	-126,11
LCA_3	-2134,02	-593,289	-140,617
UCA_1	-2040,56	-416,35	-275,1
UCA_2	-2241,48	-417,41	-270,64
UCA_3	-2147,41	-568,287	-318,515
TR_1	-2234,72	-412,957	-197,975
TR_2	-2221,72	-574,481	-226,387
WCN	-2143,6	-620,5	-220,07
SPN	-2143,6	-595,5	-219,34

Table 16 Optimised suspension characteristics from narrow limits optimisation

Camber angle$_{\text{Down position}}$	-2,44	Half-track change$_{\text{Down position}}$	-5,46	Anti-brake	5,63
Camber angle$_{\text{Up position}}$	-1,24	Half-track change$_{\text{Up position}}$	1,22		
Toe angle$_{\text{Down position}}$	-0,08	Wheelbase change$_{\text{Down position}}$	0,78		
Toe angle$_{\text{Up position}}$	0,05	Wheelbase change$_{\text{Up position}}$	-0,74		
Caster Angle	4,31	LCA_3 in wheel radius	79,2		
Roll centre height	65	UCA_3 in wheel radius	100		
Caster Trail	19,4	TR_2 in wheel radius	78,49		
Scrub Radius	-14,99	LCA_3 distance to *WCN*	-29,52		
Kingpin Angle	8	UCA_3 distance to *WCN*	-49,32		
Anti-drive	17,6	TR_2 distance to *WCN*	-45,82		

6. Conclusion

The main objective of this master thesis is speeding up the development of Formula Student suspension. Because kinematics development was recognized as one of the bottlenecks with lots of space for improvement, it was decided to tackle with this problem.

Until now members of FSB Racing Team were finding kinematics by choosing some initial suspension values and then modified its individual hardpoints until packaging and kinematics constraints were met. This very much resembles how optimisation works and so it was decided to use optimisation methods for doing the same task.

Optimisation implementation was first given a purely mathematical background with description of quarter suspension model, then the constraints, project space and objective function were described, followed by listing key points in creating the application for suspension kinematics development. Lastly, an example of optimisation was given.

There are two possible use scenarios of this application in terms of optimisation. The first is when nothing about the suspension hardpoints is known in advance, just the general suspension parameters. Then the result of optimisation will be many different solutions because of wide limits on suspension hardpoint coordinates. The other use case is when an initial suspension is already known. Then the hardpoint limits will be narrower and the focus is improving certain characteristics while keeping the other constant. Idea is that as the suspension development progresses, optimisation problem will be moving from wide limits to narrow limits, and more variables will be becoming parameters. With this approach it will be guaranteed that the resulting suspension has optimal kinematics. However, sometimes it can be seen that variables are violating the constraints even in case of hardpoint coordinates. For this reason, SLSQP optimisation method could be used which has strictly defined bounds within which it has to operate.

In the end, the objective of this master thesis is achieved, creating an application for suspension optimisation. Although the application is enabling faster suspension iteration, some experience in suspension kinematics is still necessary. It is quite easy to set up an impossible combination of characteristics constraints and allowed hardpoints position. For this reason, the application also has a separate section for investigating known suspension.

Future work could consist of remodelling of the mathematical model to a more readable and more efficient vector form, adding possibilities to calculate forces, testing different methods for achieving solutions, such as machine learning techniques, particularly

reinforcement learning. Of course, the GUI can always be improved so the user does not feel overwhelmed with all the information thrown at him.

LITERATURE

[1] William F. Milliken, Douglas L. Milliken: Race Car Vehicle Dynamics, 1995

[2] https://handwiki.org/wiki/COBYLA

[3] https://docs.scipy.org/doc/scipy/reference/optimize.minimize-cobyla.html

[4] Rod Stephens, WPF 3D Three-Dimensional Graphics with WPF and C#, 2018

[5] https://eigen.tuxfamily.org

[6] http://suspensioncalculator.com/index.html

[7] https://www.racingaspirations.com/apps/suspension-geometry-calculator/

[8] https://drracing.wordpress.com/2018/01/18/new-projects-and-suspension-kinematics-excel-tool-part-2/

[9] https://www.sympy.org/