UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

DIPARTIMENTO DI INGEGNERIA MECCANICA E INDUSTRIALE

DOTTORATO DI RICERCA IN

Ingegneria Meccanica e Industriale

IIND-02/A - Meccanica Applicata alle Macchine

XXXVI Ciclo

Research of Innovative Methodologies for the System Integration of Vehicles through Driving Simulator

Supervisore: Chiar.mo Prof. Marco Gadola

Co-Supervisore: Chiar.mo Prof. Daniel Chindamo

Dottorando:

Paolo Magri

Abstract (Italiano)

L'uso del simulatore di guida per lo sviluppo di veicoli è un argomento attuale sia nella ricerca che nell'industria, colmando il divario tra la mera simulazione e le sensazioni e i feedback che un guidatore può sperimentare, anche prima che sia disponibile un prototipo fisico.

Per System Integration, si fa riferimento al processo di combinazione e coordinamento di vari componenti, sistemi e tecnologie per creare un ambiente di simulazione driver-in-the-loop coeso e funzionale. Questa integrazione implica l'allineamento di elementi hardware e software, consentendo al simulatore di guida di replicare accuratamente scenari di guida reali in conformità con un modello veicolare affidabile.

Questo lavoro si basa sulla modellazione virtuale del veicolo con un focus principale sulle sospensioni, considerando sia la cinematica che l'elasto-cinematica. Vengono proposti algoritmi e modelli che consentono sia la progettazione che l'analisi di questi sottosistemi, nonché l'integrazione con la simulazione in tempo reale. È stato sviluppato un software con interfaccia grafica, 3DSusp, per l'analisi cinematica delle sospensioni. Sono proposti modelli completi per il calcolo dei carichi sulla sospensione e per la valutazione delle cedevolezze, caratterizzate principalmente dalle boccole in gomma con proprietà rigorosamente non lineari.

Vengono inoltre presentati metodi e modelli riguardanti la dinamica del veicolo, con particolare attenzione alla modellazione del differenziale LSD e agli algoritmi di stima per l'angolo di assetto e la loro integrazione con il simulatore di guida. Viene fornito un confronto tra due metodi per stimare l'angolo di assetto: uno basato su reti neurali artificiali e l'altro basato sul filtro di Kalman.

Infine, viene esplorata la possibilità di estendere le simulazioni per lo sviluppo di sistemi ADAS utilizzando ViWorldSim®. In questo lavoro vengono inoltre esaminati metodi di integrazione hardware, come pad tattili e dashboard.

Gli ambienti di sviluppo utilizzati per l'integrazione con il simulatore di guida sono principalmente Matlab Simulink® e ViCarRealTime®. I test online sono stati effettuati utilizzando il simulatore di guida Vi-Grade disponibile presso l'Università di Brescia.

Abstract

The use of a driving simulator for vehicle development is a current topic in both research and industry, bridging the gap between pure simulation and the feeling and feedback a driver may experience, even before a physical prototype is available.

System integration refers to the process of combining and coordinating various components, systems, and technologies to create a cohesive and functional driver-in-the-loop simulation environment. This integration involves aligning hardware and software elements to work together seamlessly, allowing the driving simulator to accurately replicate real-world driving scenarios in accordance with a reliable vehicle model.

This work is based on virtual vehicle modelling focusing mainly on suspension, considering both kinematics and elasto-kinematics. Algorithms and models are proposed that allow both the design and analysis of these subsystems, and the integration with real-time simulation. Software with a graphic interface is developed for the suspension kinematic analysis, 3DSusp. Comprehensive models are proposed for calculating the loads on the suspension and for assessing its compliance characterised by rubber bushings with strictly non-linear characteristics.

Methods and models concerning vehicle dynamics are also presented, focusing on LSD differential modelling and estimation algorithms for the side slip angle and their integration with the driving simulator. A comparison of two methods for estimating the side slip angle is presented, one based on artificial neural networks, and one based on the Kalman filter. Finally, the possibility of extending simulations for the development of ADAS systems through ViWorldSim® is also explored. Methods for hardware integration such as tactile pads and dashboards are also explored in this work.

The development environments used for integration with the driving simulator are mainly Matlab Simulink® and ViCarRealTime®. Online tests are conducted using the Vi-Grade driving simulator available at the University of Brescia.

Keywords

Driving simulator; Vehicle dynamics; Suspension; Differential; Vehicle Side slip Angle; Elasto-kinematics; Driver in the loop; System integration; ADAS;

1 Introduction

The purpose of this research is to investigate the potential of the driving simulator. To provide methodologies for the software and hardware implementation of vehicle subsystems, embracing so-called system integration. The focus of this thesis is primarily on the integration of models to simulate automotive suspension systems, considering both their kinematic and elastokinematic aspects, also suitable for real-time applications.

The use of simulations is now an established practice in every field of engineering. The matter is getting more complicated when these technologies have a direct interface with the human, in which case it becomes necessary to also consider subjective aspects, which are often difficult to quantify numerically, while remaining in the field of simulations. The driving simulator attempts to bridge this gap between real testing and simulations by introducing the concept of driver-in-the-loop, where the driver is immersed in the simulation loop. Many studies have been done on the interaction between professional driving simulators and the human driver. There are many types of simulators, ranging from static ones to dynamic ones, which are considered more “realistic” because they can reproduce the movements of the vehicle body, specifically accelerations, inclinations, and angular velocities. Figure 1 shows a dynamic simulator.

Figure 1. Vi-Grade® Dynamic Simulator DiM400

It is not guaranteed that a dynamic simulator is inherently more effective than a static one [1], [2]. The articles highlight how the validity of a driving simulator depends on the specific use case, with low-fidelity simulators potentially being as suitable for certain scenarios as high-fidelity simulators.

The first article presents a study conducted on six different driving simulators, ranging from static systems to highly dynamic motion systems with varying levels of fidelity. The researchers compared driving behaviour, perception, simulator sickness, and subjective feedback across the simulators, as well as with real-world driving data when available. The second article discusses the advantages and disadvantages of within-subject

and between-subject experimental designs in the context of driving simulator validation. The authors argue that the choice of research design can significantly influence the outcomes, and that careful consideration of the use case and experimental variables is crucial to ensure reliable validation.

Both articles emphasize the need for further research to determine which driving simulators are best suited for investigating specific use cases. The findings suggest that while high-fidelity simulators may offer certain advantages, such as reduced simulator sickness, low-fidelity simulators can still provide valid results for specific driving tasks. The choice of simulator should be guided by the specific research requirements and the need to balance realism, cost, and practical considerations.

The aim of this work is not to study the relationship between humans and the simulator, nor its validity, but to explore methodologies for integrating the vehicle and its sub-components into the simulation environment. This aspect appears to be less studied in the scientific literature, likely because, in the author's opinion, it is a topic of greater relevance within the industry, where methodologies are not always shared. However, paper [3] describes a Hardware-in-the-Loop (HiL) setup developed by Cranfield University for the design and validation of vehicle dynamics control systems. The HiL integrates real actuators (steering and brakes), a high-fidelity vehicle model, and rapid prototyping tools to test control systems in a realistic environment, including a real driver in the simulation loop. Applications include characterizing actuators, rapid prototyping of control strategies, and testing systems under critical conditions. The setup is presented as a powerful tool for developing and validating ADAS and active safety systems.

The two keywords for system integration are software-in-the-loop (Sil) and hardware-in-the-loop (HiL), representing the integration of physical systems and mathematical models interacting with the driving simulator, respectively. This means that both the software and hardware, which will be installed on the actual vehicle in the future, can first be developed and tested using the driving simulator by integrating them into the simulation loop.

The work carried out in this PhD is mainly divided into two strands: The first purely focuses on suspension modelling, the development of methodologies for driving simulator integration and assisted design. This purely research work also includes comparisons between different methods for the vehicle side slip angle estimation and some hardware integrations. The second strand groups together a series of works resulting from research projects carried out with the Automotive Engineering and Design group at the University of Brescia, with the companies YCOM® and Adas Mobile®.

The first part of the work focuses on the continuation of the development of 3DSusp, developed during the author's master's thesis, a user-friendly software with a graphical user interface, for suspension design and kinematic analysis. The software was improved, and new features were added, including new elements and integration with ViCarRealTime®, software for vehicle dynamics simulations, both offline and for the Vi-Grade® static driving simulator at the University of Brescia (Figure 2).

Figure 2. Vi-Grade® static driving simulator at University of Brescia.

Much of the research then focuses on the analysis of automotive suspension loads and compliance. Methods and algorithms have been proposed for calculating constraint reactions and solving the elastokinematics of suspensions. In particular, a complete method is presented that also takes into account the presence of rubber bushing with non-linear characteristics, validated using multibody software. These algorithms were written in Matlab® and then also used in the Simulink® environment to integrate them in co-simulation with the driving simulator to work also in real-time.

Suspension and steering design play a major role in ensuring the correct dynamic behaviour of road vehicles. Passenger cars are especially demanding from this point of view: NVH (Noise, Vibration and Harshness) and ride comfort requirements often collide with active safety-related requirements such as road holding in steady-state conditions and stability in transients. Driving pleasure is also important for market success, therefore accurate steering feedback and predictable handling properties are additional priorities.

In motorsport applications, on the other hand, proper calculation of suspension compliance can make a difference in performance and feedback for the driver. In this field, minimizing compliance is important and comfort is not a target to chase, spherical joints, the so-called Uniballs, are used to make the connections between suspension elements. In passenger cars, on the other hand, flexible rubber bushing is used, which can absorb shock and vibration. These two different connections are shown in the Figure 3.

Figure 3. a) Uniballs used in motorsport application. b) Rubber bushings used in passenger cars.

Modern passenger cars require an intensive effort for the design of suspension elastokinematic properties because of their relevant impact on ride, handling, and steering feel as well [4]. This is reflected in a vast engineering literature on the subject. The use of multi-body models for instance, although being the industry standard, is often considered demanding because of the high level of details required. The development and application of relatively simple, dedicated design and simulation tools instead is often considered more practical, especially when designing a new suspension concept ground up. This tendency is clearly visible in the related literature as many publications describe self-developed, peculiar methodologies, often responding to a varied level of complexity as well as to a different computational approach. A multibody model is often used as validation in these cases.

The so-called multi-link rear suspension concept was introduced in the early eighties by Mercedes with the W201 (190) model, which represented a leap forward in terms of elastokinematics. The design requirements aimed at decoupling longitudinal compliance and wheel guidance, as well as the innovative way they were achieved are accurately described in [1], which is considered a milestone in the related literature. FEM analysis on suspension link deflection was used in order to simulate the deflection of rubber bushings.

Among other efforts, significant contributions on design requirements for suspension kinematics and elastokinematics have been released by Volvo [2], Nissan [7], [8], Hyundai-Kia [9], and Ford [10], with the latter focused on the so-called integral link geometry.

With specific regard to computational methodology, the work done stems from the study of industrial robot kinematics [11]. Specifically, the automotive suspension is seen as a Stewart-Gough platform where the end effector represents the wheel hub carrier. In the literature, much work has been done to study the kinematics and compliance properties of parallel robots and manipulators where arms and joints having finite stiffness are analysed [12], [13], [14], [15], [16].

Analysing the literature in the automotive field, pure kinematics is dealt with by M.B. Gerrard in [17]. Another effort from the same author, this time regarding kinematics synthesis is [18], where significant limitations also occur, small displacements around the design configuration apply, and the interposition of elements between

suspension linkages is not considered, a layout like the integral link cannot be modelled. Kinematics is also on focus in [19], while the synthesis of five-link kinematics is discussed in [20], [21]. This is a relevant subject, even when neglecting elastokinematics, due to the optimization complexity with so many design variables available to play with.

Pure kinematics computation is also used for applications related to motorsport [22], [23], [24] and for educational purposes as well [25]. The integration of a dedicated software with a toolchain for vehicle dynamics simulation is presented in [26]. It features a built-in graphical user interface conceived to facilitate the kinematics design process.

Papers from well-known authors Knapczyk et al. are cited quite often. Among many, [27] still refers to [5] as a case study and reports a method for computing the force/displacement relationship of a five-rod multi-link suspension based on the vector algebraic method. Rubber bushings are described in detail, although their stiffness is considered in the radial and axial directions only. While [28] mainly deals with vertical dynamics, an approach similar to [27] is found in [29], where the role of the subframe (also referred to as "bogie" in its typical arrangement, i.e. connected to the chassis by four, vertical-axis, large rubber bushings) is also taken into account in order to simulate the elastokinematic behaviour of the rear axle as a whole. Again, from the same authors comes [30], dealing once again with a rear, five-link suspension, although with attention to vertical dynamics as well.

The contribution of rubber bushings is considered in [31], where a comprehensive model of a double-wishbone front suspension is used for durability simulations, with specific focus on the angular travel of ball joints fitted at the outer end of both the lower and upper control arms as well as at both ends of the steering rod. Bushings feature non-linear characteristics in the axial, radial, torsional, and conical directions.

Various authors adopted the equivalent stiffness approach in order to correlate suspension geometry and rubber bushing characteristics to the stiffness at the wheel centre. This method is strongly focused on the design of kinematics and compliance properties of the suspension as a whole. Compliance properties are computed at the wheel centre as a function of linear bushing characteristics in [32]. Kinematics is neglected in this case, as it is in [33]. In this case the same authors take link flexibility into account as well.

Another interesting effort based on the stiffness matrix is [34], from the same author of [17], [18]. Once again however the paper deals with small displacements and linear bushings, which should be seen as an important limitation given their typically non-linear behaviour [31].

Elastokinematics is the subject of papers focusing on the MacPherson layout as well. Examples are [35], [36], where the bushing stiffness is considered in the radial and axial directions and fully linear.

An extensive sensitivity analysis is carried out in [37], focusing on the interactions between steering kinematics, anti-roll bar rate, elastokinematics, and handling response of a front-wheel-drive passenger car by means of multi-body modelling. The correlation between elastokinematic modelling and experimental testing

on Michelin's well-known K&C (Kinematics & Compliance) test rig is the subject of [38]. Rubber bushings are modelled quite accurately, while according to the authors ball joint friction and the visco-elastic properties of rubber also play an important role in the overall suspension behaviour.

Starting again from experimental data, reverse engineering of the three translational bushing characteristics is the aim of [39]. In this case rubber joints can be placed either side of each suspension link. All bushing characteristics however are deemed as linear: this is a typical case study of an identification process. It also seems that the assumption of variable length suspension links might require further assessment.

Non-linear stiffness of bushings along all six degrees of freedom is instead adopted in [40], aimed at the computation of joint forces on the chassis side of a double-wishbone suspension. In this work, which in the author's opinion seems to be one of the most complete, the wheel-side bushings are not considered, and all elements except the spring are rigid. However, the spring is considered as a simple elastic element without considering that it is often mounted on the frame and suspension by means of rubber bushings. The nonlinearity of stiffnesses is considered by means of force-displacement curves composed of several linear segments.

Finally, the methodology behind an earlier version of the well-known commercial tool Shark® by Lotus Engineering is described in [41]. In this tool, both wheel-side and frame-side bushings can be inserted, but limitations remain: stiffnesses can only be linear. Also, the kinematics of the system are not taken into account; the tool calculates the stiffness matrix of the suspension, linearizing it, but this is only good for small deformations.

As a general rule, most of the papers in the literature tend to neglect one or more factors of the real-world design. Either the bushings are represented with linear characteristics, or with non-linear curves but only for translational displacements i.e. in the axial and radial directions only, while the angular deflection rates are not considered. Another limitation that is often found is the fact that rubber bushings are featured on the chassis side of suspension arms only, while in current suspension systems they are sometimes located between the suspension arms and the wheel carrier as well, or even in the middle of the trackrod. Some papers also tend to linearize suspension behaviour in a small displacement range around the static or design configuration with a single stiffness matrix.

Since flexible bushings are used as interface between the suspension arms and the chassis, extra degrees of freedom make the design process a complex task. While the use of a multi-body model is common practice in the industry, a dedicated computational tool can be more practical and straightforward, especially when undertaking the design of a new suspension concept ground-up.

One chapter of this thesis is dedicated to the description of some algorithms for calculating suspension elasto-kinematics, both for motorsport and non-motorsport applications. The various models were developed during the author's PhD period, starting from simpler algorithms to a comprehensive one.

Two simplest algorithms are based on calculating the loads acting on the suspension joints, but they consider spherical joints for load computation and consider the arms as ideal connecting rods, thus subjected only to axial load. In fact, in the first subchapter, before describing the algorithms for elasto-kinematics, a method a method is described to be able to calculate the constraining reactions on the suspension joints. This method was created not only for the purpose of developing elasto-kinematics but so that it could also be implemented in 3DSusp to include additional functionality in the program that would allow one to be able to know the loads on the suspensions to perform a first design process and/or use these results as input for FEM analysis.

These elasto-kinematics models will also be implemented in the Vi-CarRealTime® environment and also for real-time simulations. It is well known that correct modelling of elastokinematics, taking non-linearities into account, is crucial for several aspects: design [31], for handling [37] and also for ride & comfort [42]. The integration of models characterizing elastokinematics in vehicle dynamics simulations is a recurring topic, with the main challenge being the speed of computation [43].

With regard to real-time simulations, the current trend in the literature shows an approach based on modelling suspension compliance through artificial neural networks (ANN) [44]. In this way, it is also possible to consider dynamic effects as opposed to a quasi-static solver. However, a significant amount of data is needed to train the network, which can be particularly complicated or even impossible during the design phase. The purpose of this work is to provide a method for solving the suspension elastokinematic problem using a general procedure that enables the design of any layout under any combination of jounce and steering or load. Bushings can be described with real-world, non-linear stiffness curves for all six degrees of freedom. They can be located on either side of each suspension arm, i.e., on the chassis side and/or the wheel side. The axial flexibility of a track rod can also be represented by means of equivalent, non-linear bushings. The wheel bearing stiffness can be considered as well. Wheel movements, hence variations of vehicle dynamics-relevant parameters like camber, side view angle, toe, track, wheelbase, and vertical displacement, can be computed under any combination of road loads, also considering steering due to rack translation. Loads through suspension joints, components, and chassis pick-up points can also be computed.

In general, elastokinematics is implemented in vehicle dynamics simulations using look-up tables, often derived from experimental data obtained through K&C or SPMD, then applying the superposition principle [45]. This method is characterized by requiring negligible computational resources.

One of the aims of this work is to show how the superposition principle, which can work well in some cases, does not allow for a correct characterization of compliance when large lateral and longitudinal accelerations are involved, as mentioned in [45]. A comparison, through simulation, with a model based on look-up tables and the superposition principle is proposed to show how vehicle behavior can change in a test with large accelerations. The simulation is also carried out to demonstrate the possibility of using the method proposed in this work in real-time.

Research work on the estimation of the vehicle side slip angle, one of the most important parameters for characterising vehicle dynamics, is also proposed. A comparison between two models is proposed, the first based on artificial neural networks, the second based on a Kalman filter. Finally, a methodology for implementing the neural network on the Wintax data acquisition software on the Vi-Grade® driving simulator is also presented.

Vehicle sideslip angle (VSA) estimation has been a big challenge since the introduction of the very first on-board active systems controlling vehicle stability, such as the electronic stability control (ESC, aka ESP) in the early 1990s [46]. It can be used to investigate the cornering behaviour in terms of understeer/oversteer and stability, which are related to active safety and drivability, for either passenger cars or high-performance vehicles. Moreover, active safety systems such as ESC can be used in conjunction with VSA to extend the vehicle performance envelope and/or enhance driver perception and comfort [47] as the average driver feels uncomfortable when large VSA values occur, during emergency maneuvers for instance [48]. It is also worth mentioning that vehicle drivability is strictly related to VSA: whenever it is large, the authority of the steering wheel angle in generating a yaw moment becomes significantly reduced. VSA can also be used for autonomous driving strategies and advanced motion control of high-performance electric vehicles, thus improving active safety and efficiency which are considered key factors in the automotive market [49].

VSA estimation is usually classified as Kalman Filter-based and Neural Network-based. According to [50] however none of the proposed methods succeeded as the most effective, and the problem is not yet fully resolved. The aim of chapter 4.1 is to further investigate estimation performances of both methods through a direct comparison with real-world data.

Work on hardware integration is also presented, special touch-sensitive paddles were mounted on the driving simulator to manage the shift logic in an original way. Methods for creating customised dashboards that communicate with the driving simulator are also presented.

The research project in collaboration with YCOM® led to the study and modelling of mainly two vehicle subsystems. Firstly, the inboard part of the suspension, i.e. everything including spring elements, shock absorbers and anti-roll bars, with the possibility of being able to control them fully during the simulation, with the aim also of developing active suspension controls. As second the modelling, again in Simulink®, of a limited slip differential (LSD) with the aim of creating active controls by actuating the clutch pack.

Finally, the research project with Adas Mobile® allowed the author to explore the possibility of developing ADAS systems through the driving simulator, to create traffic scenarios, through the ViWorldSim® software.

2 3DSusp

The first part of the work starts from my master's thesis: the development of suspension design software, a screenshot is shown in Figure 4.

This tool can calculate the kinematics of any type of independent wheel suspension, such as Multilink, Double Wishbone, MacPherson, Trailing Arm, etc. Much work has been devoted to the development of the graphical interface; in fact, the suspension is drawn in both orthogonal projections and a perspective view. The graphics were developed using the OpenGL libraries that allow them to communicate directly with the computer's graphics card.

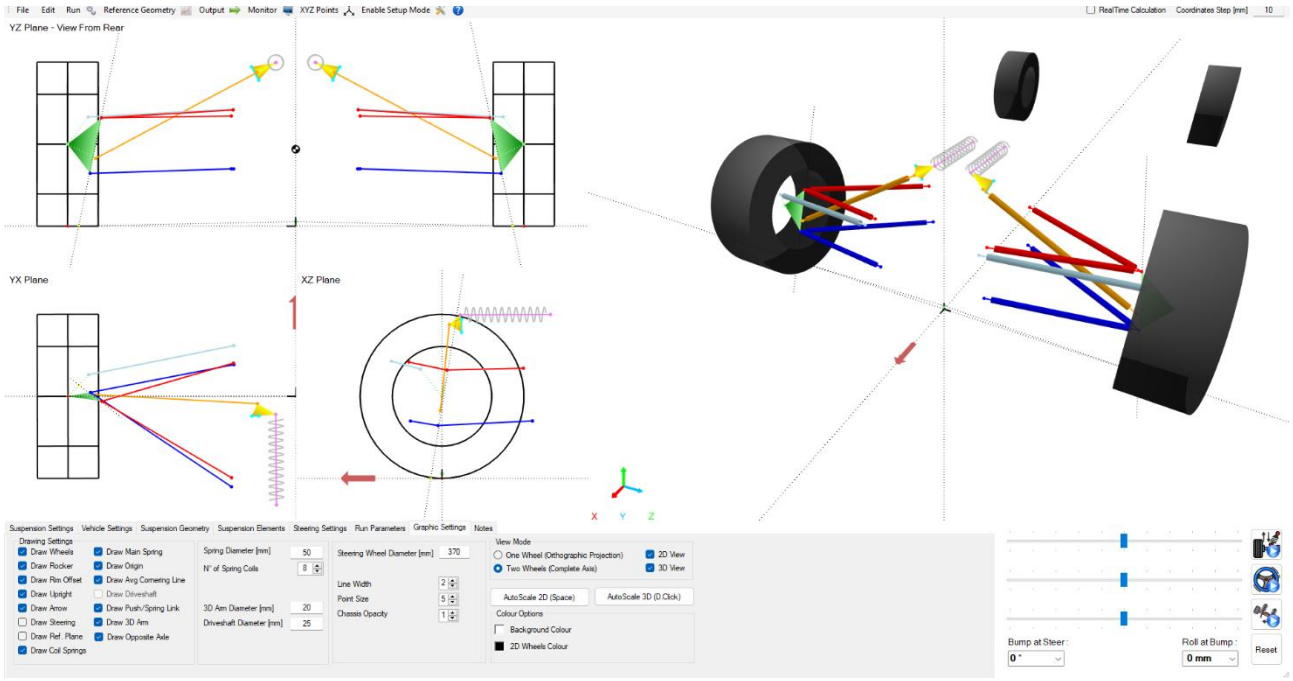


Figure 4. 3Dsusp software screenshot representing a double wishbone suspension of an F4 race car.

The calculation method is derived from robotics and is based on that used for the platform of Stewart-Gough and using the multi-variable Newton-Raphson method as the solver of the nonlinear equations.

The software has been developed with a user-friendly interface. For example, it is possible to select different suspension points with the mouse and move them quickly with the header and monitor the various parameters calculated by the software in real-time.

This tool also has a rather innovative feature that is not present in other competing software, the so-called *design mode*: where it is possible to simulate different setup changes to be made on the suspension, simulating the adjustments that are made for example on the track during a race.

It is obviously not the purpose of this text to describe the operation and methodologies of software development, already discussed extensively in the master's thesis.

During the PhD, however, a need arose to develop new features and improve the tool. Significant improvements were made on the computational speed and quality of the GUI (Figure 5) by streamlining the functions from a computational point of view and by studying the functionality of the OpenGL graphics libraries more thoroughly. For example, now the software to perform a suspension bump calculation with 100 steps takes less than 0.2 seconds, instead, in the version created for the master's thesis it took about 8 seconds. Most of the improvement was achieved by better studying how the VB.net programming language handles memory when calling functions.

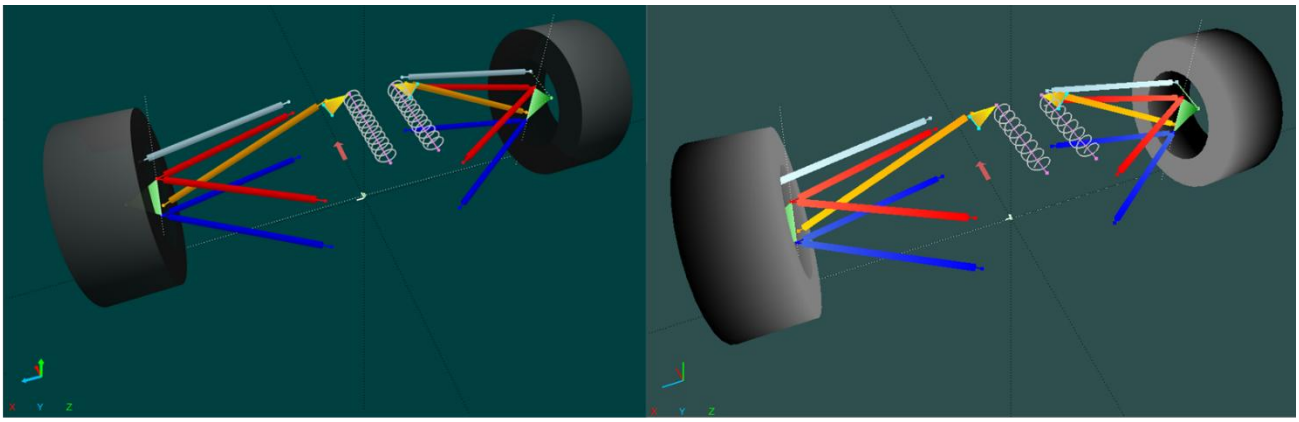


Figure 5. Comparison of software graphics now a) and just finished master's degree b), 4 four years ago.

Instead, the development of other features arises mainly from two factors, the first being a research contract made with YCOM, a race car manufacturer specializing in composites, and the second being the presence of the Vi-Grade driving simulator purchased by the department in 2019.

The research project in collaboration with YCOM involved the development of a fully electric race car. Specifically, for the design of its suspension, the need arose to add the ability to model other elements such as anti-roll bars, heave elements, and roll elements. Other external projects also made it necessary to include an additional suspension model in 3DSusp, the integral link.

With the purchase of the Vi-Grade driving simulator and with the use of their Vi-CarRealTime® vehicle dynamics simulation software, the need arose to interface these with 3DSusp so that a suspension could be designed on the software and in a few steps could be entered into the virtual vehicle model and eventually driven on the simulator as well.

2.1 New suspension elements

As mentioned earlier, new suspension elements were modelled in the 3DSusp software to allow for their design. These elements do not directly affect the suspension kinematics and wheel movement but connect the two suspensions of the same axle and are used to control heave, roll, and pitch movements.

Specifically, these new elements are:

- C-Shaped Antiroll Bar
- T-Shaped Antiroll Bar with Third Element
- Heave Element and Roll Element

2.1.1 C-Shaped Antiroll Bar

The c-shape is one of the simplest anti-roll bars (ARBs), also used in passenger cars. It consists of a "C"-shaped bar where the middle part functions as a torsional spring and two connecting rods that connect the bar to the two wheels of the axle, respectively. The special feature of this element is its simplicity, which is why it is used in ordinary cars and unlike the other elements mentioned in later chapters does not make it necessary to have a suspension scheme with rocker and pushrod/pullrod. In fact, it is possible to attach the anti-roll bar to the hub carrier or to one of the wishbones or to one of the links if the suspension is a multilink type. On the other hand, if it is a race car, the bar is often tied to the rockers. A schematic of a C-shaped anti-roll bar can be seen in the Figure 6.

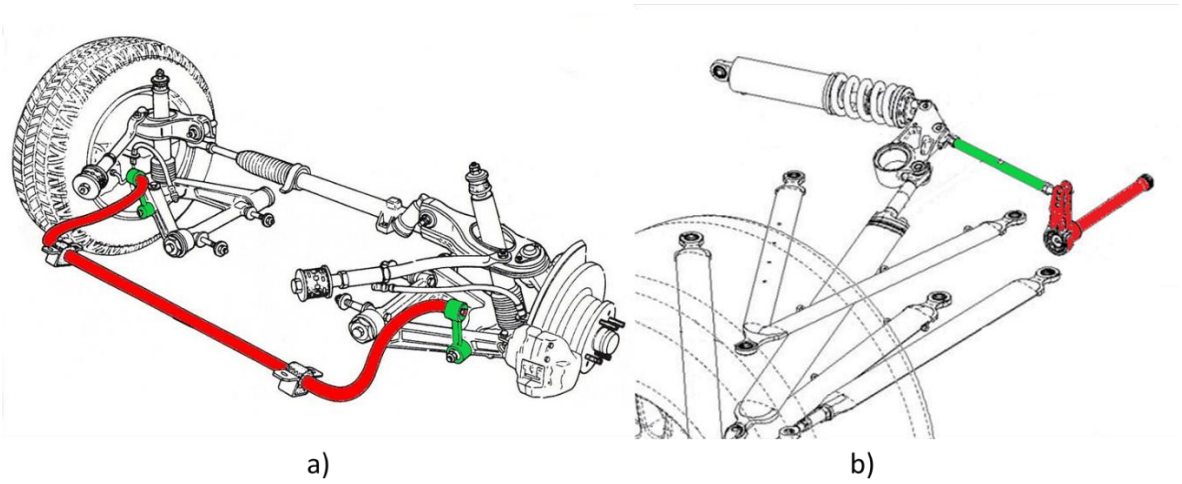


Figure 6. A schematic of a C-shaped anti-roll bar. a) ARB attached through the wishbones. b) ARB attached through the rockers. In red the torsion "C" bar and in green the links.

Three points in space are enough to model the ARB:

- **E**: Link attachment point on the element that can be the rocker or a wishbone or a rod.
- **B**: Link attachment point with the bar.
- **P**: Pivot point defining the torsion axis of the bar.

These three points totally identify the position of the ARB as they define the position on the left wheel; the points on the right wheel are symmetrical to the median plane of the axle.

First, a reference system integral to the element to which the *ARB* is constrained is defined, which is called the ERS (Element Reference System). The orientation of this reference system with respect to the Global Reference System (GRS) is known from the kinematics calculation performed in the software. In particular,

the transformation matrix M_{ERS} containing the direction cosines and translation vector of ERS with respect to GRS is known. In this work, the global reference system will always be defined with the X-axis parallel to the longitudinal axis of the vehicle and direction coinciding with the direction of driving, the Z-axis is instead vertical, and the Y-axis is accordingly defined as positive to the left.

Next, a second reference system called BRS is identified, by convention the x-axis of this reference system is placed along the torsion axis of the bar. A diagram for identifying the C-shaped anti-roll bar can be seen in the Figure 7.

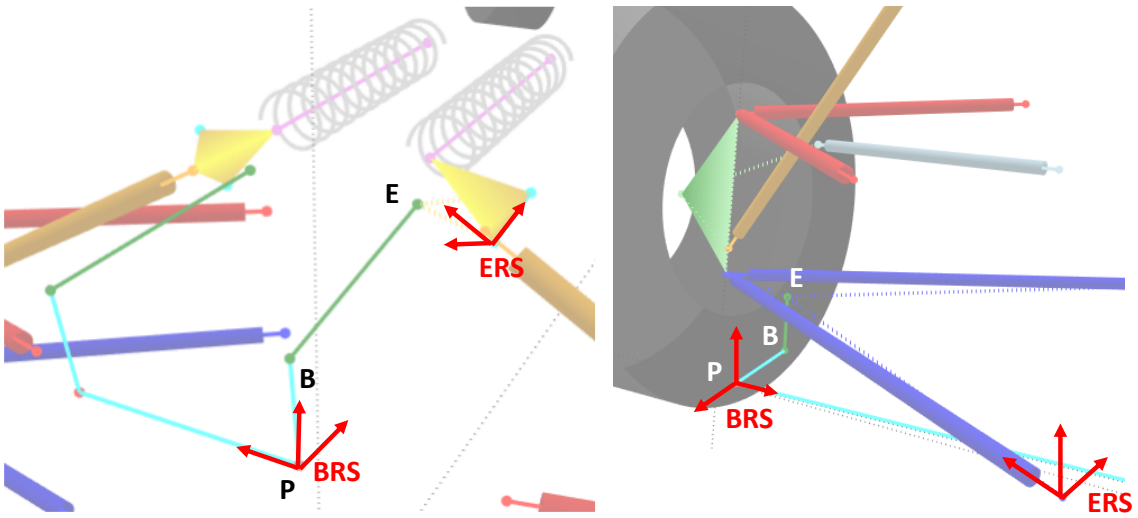


Figure 7. Schematic of the C-shaped anti-roll bar with reference systems. a) Bar attached to the rocker. b) Bar attached to the lower wishbone.

The purpose of modelling the bar is to be able to calculate its motion, and particularly its torsion during the movements of the two wheels. What happens is that the BRS reference system will have to rotate a certain angle around the torsion axis of the bar. Given the two rotation angles, one for each wheel, the difference expresses the torsion angle of the bar.

In this case, the system has only one degree of freedom, which is the rotation of the bar around its own axis, passing through point P and directed as the Y-axis of the GRS. The degrees of bar rotation are calculated through the Newton-Rapshon algorithm by imposing that the length of the EB rod does not vary during the motion of the suspension. The point E is rigid with the reference system of the element (ERS) to which it is bound, so its position is known during the movement of the suspension. The point B is solid with the BRS reference system, which can only rotate around the rotating axis of the ARB. Thus, the position of point B is function of the angle of rotation of the bar θ. The actual value is easily calculable by knowing the coordinates of the points E and B, thanks to the transformation matrices that define the different reference systems as described in the following equations.

$$X_{ERS} = M(\theta) \cdot X_{GRS} \tag{2.1}$$

$$X_{GRS} = M(\theta)^{-1} \cdot X_{ERS} \tag{2.2}$$

where X_{ERS} and X_{GRS} are the vectors that represent the coordinates of a point in the ERS and GRS reference systems respectively and are defined as in equation 2.3.

$$X = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.3}$$

The $M(\theta)$ matrix represents the transformation matrix containing the directors cosines and the translation vector of the ERS reference system relative to GRS. The distance between two points is calculated by calculating the difference between the two vectors representing the coordinates, eq. 2.4.

$$\overline{EB} = \|X_E - X_B\| \tag{2.4}$$

The function $f(\theta)$ is then defined in eq. 2.5, representing the length difference between the current value of the EB segment, that depend to θ and the initial one.

$$f(\theta) = \overline{EB}_{Actual}(\theta) - \overline{EB}_{Initial} \tag{2.5}$$

Now it is possible to find the correct θ value by solving the equation $f(\theta) = 0$ using the Newton-Raphson iterative algorithm, described in eq. 2.6.

$$\theta_{i+1} = \theta_i - \frac{f(\theta_i)}{f'(\theta_i)} \tag{2.6}$$

Until θ_{i+1} is smaller than a number close to 0. $f'(\theta_i)$ represents the derivative relative to θ of the function $f(\theta)$ and is calculated numerically by applying an infinitesimal vertical shift to the suspension.

It is therefore possible to know for each bump value of the suspension, the relative rotation of one side of the bar. It is obvious that for symmetrical bump (Heave) the bar moves but does not undergo any twist, so it is interesting to calculate the twist of the bar for different values of shaking between right and left wheels, therefore for different roll angles.

In addition, the motion ratio MR of the bar can also be calculated. The motion ratio of an element is generally defined as the derivative of its motion (twisting in this case) with respect to a wheel motion. By imposing small displacements on the left and right wheels, it is possible to calculate the torsion of the bar relative to that small displacement. Two different motion ratios are calculated, the first is defined as the derivative on the roll angle and the second as the derivative on the vertical displacement of the wheel, considering the other fixed. Specifically, the former will be a dimensionless number, the latter will have units in $°/mm$.

Shown in the Figure 8 is the graph, in the 3DSusp GUI, showing the trend of bar torsion during the roll of an example suspension.

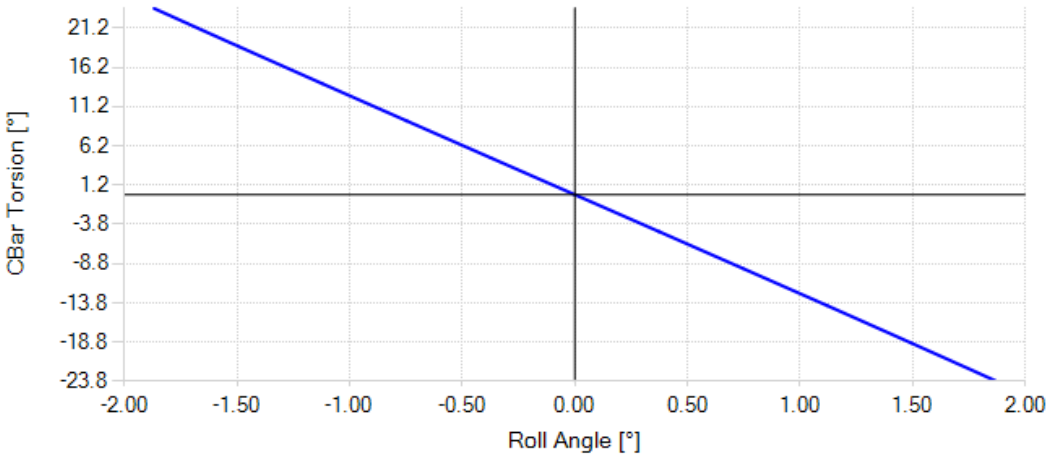


Figure 8. C-Shaped ARB torsion angle vs roll angle in 3DSusp interface.

2.1.2 T-Shaped Antiroll Bar with Third Element

Another type of anti-roll bar, used in motorsport, is the T-Shaped Antiroll Bar. Again, there are two rods connecting the suspension to the ARB, in this case, however, the connecting rods are always mounted on the rocker. The bar is free to rotate about its axis of rotation, which is always parallel to the Y-axis of the GRS. This movement occurs during the parallel bump (Heave) movement. During roll movement, on the other hand, the bar is deformed, and in general, stiffness is controlled by making the vertical section of the bar work in torsion. In some cases, knives can also be used to make the horizontal sections of the ARB work in bending.

This scheme also allows for the possibility of mounting the so-called third element, which is a spring/shock absorber bound on one side to the chassis and on the other side to the bar, usually in its symmetry's axis. The purpose of this element is to provide stiffness during heave motion while maintaining a zero contribution to roll stiffness. A schematic of a T-shaped anti-roll bar can be seen in the Figure 9.

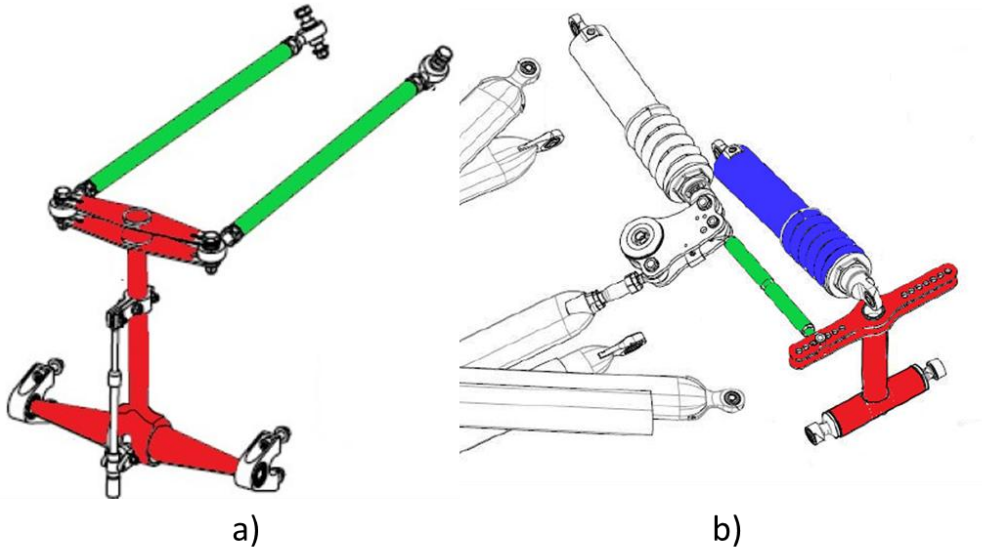


Figure 9. A schematic of a T-shaped anti-roll bar. a) ARB without third element. b) ARB with third element. In red the torsion "T" bar, in green the links and in blue the third element.

Exactly as in the "C" Bar, three points in space are enough to model the ARB:

- **E**: Link attachment point on the element that can be the rocker.
- **B**: Link attachment point with the bar.
- **P**: Pivot point of the bar.

These three points totally identify the position of the ARB as they define the position on the left wheel; the points on the right wheel are symmetrical to the median plane of the axle. For point P, only X and Z coordinates can be varied since Y is fixed at 0.

For the third element, however, it is necessary to define two points: the **TB** point, connected to the bar, and the **TC** point, connected to the frame.

As explained in the previous subchapter, the GRS and ERS reference systems are defined and a diagram for identifying the T-shaped ARB and the third element can be seen in the Figure 10. The points E' and B' represent the corresponding symmetrical points on the opposite wheel of the same axis.

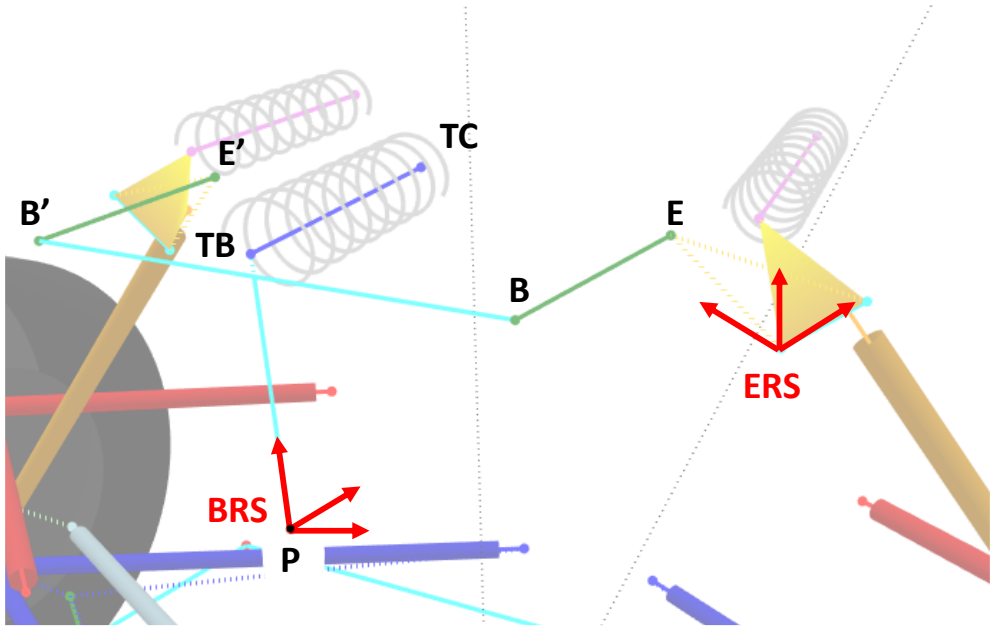


Figure 10. Schematic of the T-shaped anti-roll bar with reference systems.

Mathematically modelling this scheme allows to know the motion of the bar during bump and roll motion. The bump motion can also be useful in the absence of the third element to study the overall dimensions during design. The purpose is to calculate the angle of rotation of the BRS reference system around the torsion axis of the bar and eventually the compression of the third element.

Unlike the C-Shaped ARB, this system has two degrees of freedom: the rotation about the transverse axis (parallel to GRS Y-axis and passing through P) and rotation about the torsion axis of the bar, which is the axis passing from the point P and the midpoint between B and B'. This rotation represents the torsion of the ARB.

In this case, to calculate the two bar rotations, the other suspension wheel must also be considered. By causing the two wheels to make different movements, the points E and E' will move, and the two degrees of freedom, called the θ and δ, can be calculated through the Newton-Rapshon algorithm by imposing the length of the EB and $E'B'$ connecting rods as constant. It is then necessary to define the function $f(\theta, \delta)$ to be imposed equal 0 as a vector function defined in the eq. 2.7.

$$f(\theta,\delta) = \begin{bmatrix} f_1(\theta,\delta) \\ f_2(\theta,\delta) \end{bmatrix} = \begin{bmatrix} \overline{EB}_{Actual}(\theta,\delta) - \overline{EB}_{Initial} \\ \overline{E'B'}_{Actual}(\theta,\delta) - \overline{E'B'}_{Initial} \end{bmatrix} \tag{2.7}$$

The position of the points E, E', B, and B' during the movement of the suspension are calculated similarly to what was seen in the previous subchapter. The θ and δ values are then calculated with the Newton-Raphson algorithm, this time defined for multi-variable vector functions, eq. 2.8.

$$(\theta,\delta)_{i+1} = (\theta,\delta)_i - [J(\theta,\delta)_i]^{-1} \cdot f(\theta,\delta)_i \tag{2.8}$$

Until RMS of θ and δ are less than a small number. Where $J(\theta, \delta)$ is the Jacobian matrix containing the partial derivatives of the function, as in the eq. 2.9.

$$J(\theta,\delta) = \begin{bmatrix} \dfrac{\partial f_1(\theta,\delta)}{\partial\theta} & \dfrac{\partial f_1(\theta,\delta)}{\partial\delta} \\ \dfrac{\partial f_2(\theta,\delta)}{\partial\theta} & \dfrac{\partial f_2(\theta,\delta)}{\partial\delta} \end{bmatrix} \tag{2.9}$$

Having solved the function, the position of the bar is known and the θ angle corresponds to its torsion, and it is also possible to calculate the position of the bar-bound TB point of the third element to calculate its compression by the distance between TB and TC. In the same way as in the previous subchapter, it is also possible to calculate the motion ratios of the bar and the third element, both in bump and roll. Shown in the Figure 11 is the graph, in the 3DSusp GUI, showing the trend of bar torsion during suspension roll.

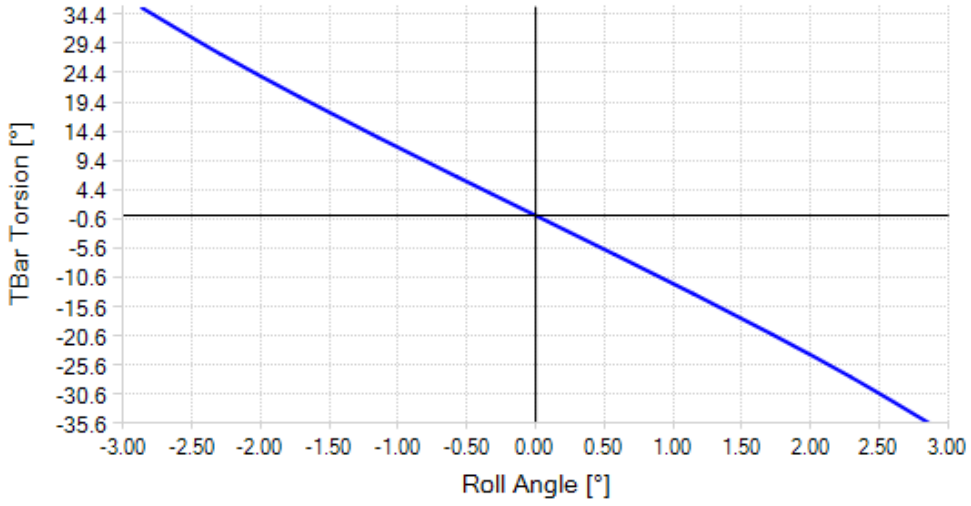


Figure 11. T-Shaped ARB torsion angle vs roll angle in 3DSusp interface.

2.1.3 Heave Element and Roll Element

The heave and roll element are two devices that are used in race cars. Each of these elements consists of a spring/shock absorber that connects the two rockers of the same axle, as can be seen in the Figure 12.

Figure 12. Heave element and roll element installed on a race car.

The purpose of these elements is to provide stiffness (or damping) for heave (parallel bump) and roll motion, respectively, by making them work completely separately. In fact, typically the heave element is mounted horizontally so that it compresses or extends only for parallel vertical movements of the suspension, while remaining almost unchanged in length during opposite vertical wheel movements. The roll element, on the other hand, is mounted so that its length varies only at roll and not during heave.

The modelling of these two elements is quite simple; for each element it is necessary to identify two points, each constrained to one of the two rockers. The heave element is defined through the points H1 and H2, the roll element through R1 and R2, as can be seen from the Figure 13.

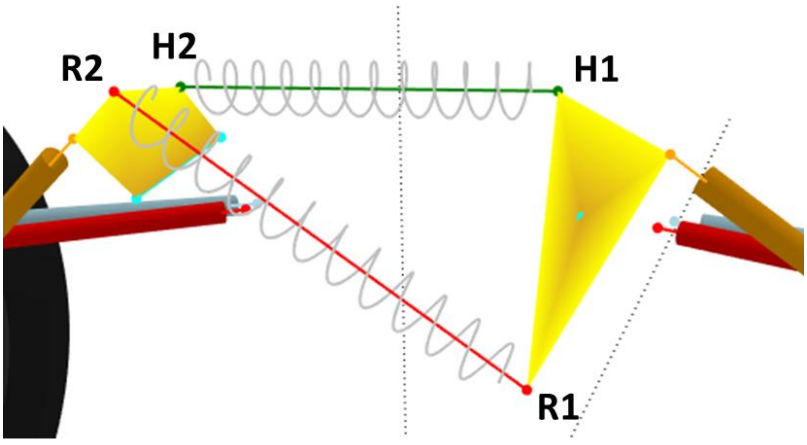


Figure 13. Schematic of the heave element and roll element

As seen in previous sub-chapter 2.1.2, points are constrained to the relevant element reference systems (ERS), and through the associated transformation matrices, coordinates are derived in the global reference system

(GRS). By calculating the distance between the points, the compression of the elements is derived, and by applying infinitesimal displacements to the wheels, the derivative of the compression with respect to wheel motion, that is the motion ratio, can also be calculated.

Figure 14 shows, in the 3DSusp interface, of the length of the two elements as a function of the movement of heave of an example suspension. The suspension bump, while being the independent variable is plotted on the y-axis so that it is more intuitive in that a positive bump value represents the suspension compressing, thus the wheel rising upward relative to the chassis.

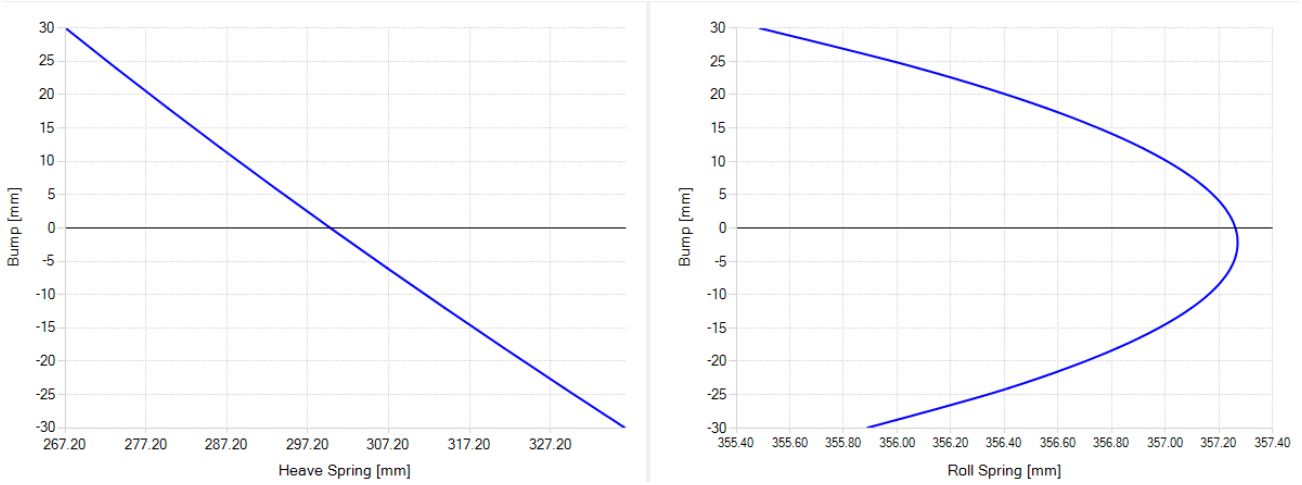


Figure 14. Heave element and roll element length vs bump travel in 3DSusp interface

It is possible to see that the roll element, called the roll spring in 3DSusp, also works slightly in heave, but the magnitude of its deformation is negligible compared to that of the heave element. The opposite phenomenon also happens for the heave element during suspension roll, as can be seen in the Figure 15.

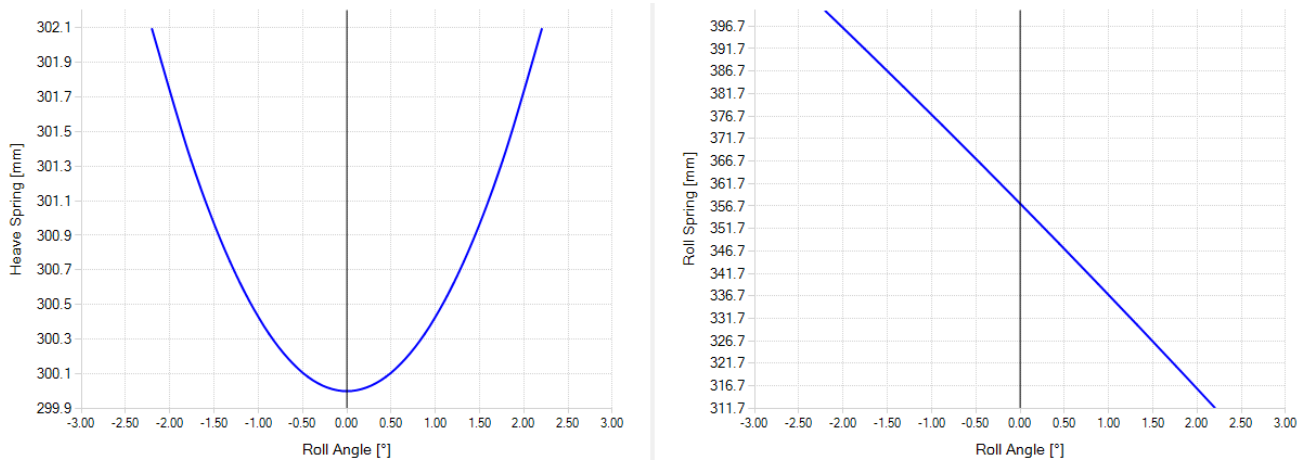


Figure 15. Heave element and roll element length vs roll angle in 3DSusp interface

2.2 Integral Link

A new type of suspension that can be modelled, the so-called Integral Link, has been implemented in 3DSusp.

The algorithm for calculating suspension kinematics is inspired by that used in the field of robotics for parallel robots, particularly the Stewart-Gough platform, Figure 16. In this platform, the frame connects the moving part via 6 connecting rods to control its 6 degrees of freedom. The similarity to an automotive suspension is obvious: the wheel is connected to the frame via connecting rods that give 5 degrees of constraint.

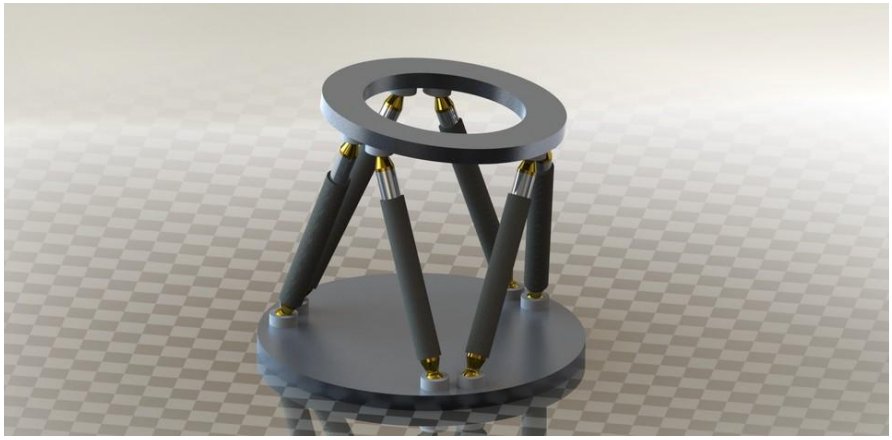


Figure 16. Stewart-Gough platform.

The sixth connecting rod, on the other hand, is a segment constrained to the ground point of the wheel, vertical and of infinite length (ten orders of magnitude larger to simulate a big number) to control the vertical movement of the wheel. The goal is to find the position of the wheel, defined by 3 coordinates and 3 angles by making sure that the 6 connecting rods are of a fixed length. Again, this involves solving 6 nonlinear equations, so the Newton-Rapshon algorithm with the appropriate 6x6 Jacobian matrix is used. The operation of the kinematics resolution algorithm is not the subject of this thesis, as it is explained in depth in the master's thesis of the author.

With this type of model, it is possible to build almost any type of independent-wheel suspension, those where individual arms connect the chassis of the car to the upright. By positioning the 5 connecting rods appropriately, it is possible to model suspension types such as Multilink, Double Wishbone, Trailing arm, etc.

There is, however, a particular type of suspension called “integral link”, which has been used a lot lately, where there is a connecting rod that does not directly connect the hub carrier with the frame, but generally connects it with the lower arm. With this 'parallel robot' model, it is not possible to model the integral link. Some examples on real vehicle suspensions, such as the Ferrari 488 and Tesla Model 3, are shown in the Figure 17.

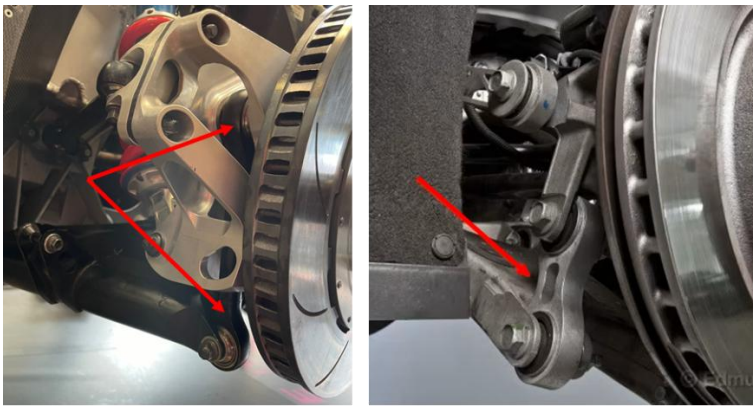


Figure 17. Integral suspension with link connecting arm and upright on real cars. a) Ferrari 488 rear suspension. b) Tesla Model 3 rear suspension

Figure 17 show a small connecting rod (highlighted with the red arrow) that connects the lower arm to the hub carrier. This is a solution that is implemented to simplify the suspension architecture from the chassis side. The number of total degrees of constraint do not change, but it was necessary to implement a change to the algorithm in order to consider that the coordinates of this connecting rod (integral link), during the suspension movement, change on both the wheel side and the frame side. From the computational point of view, nothing new is considered; a reference system integral to the arm is considered so that the new coordinates of the integral link joint can be computed during the kinematic computation. The key difference is that in all other models, the hardpoints on the chassis side remain fixed during movement. In the integral link, however, one of these points moves with the suspension, and its position must be recalculated at each iteration.

In the Figure 18 it is possible to see the implementation of the integral link in 3DSusp.

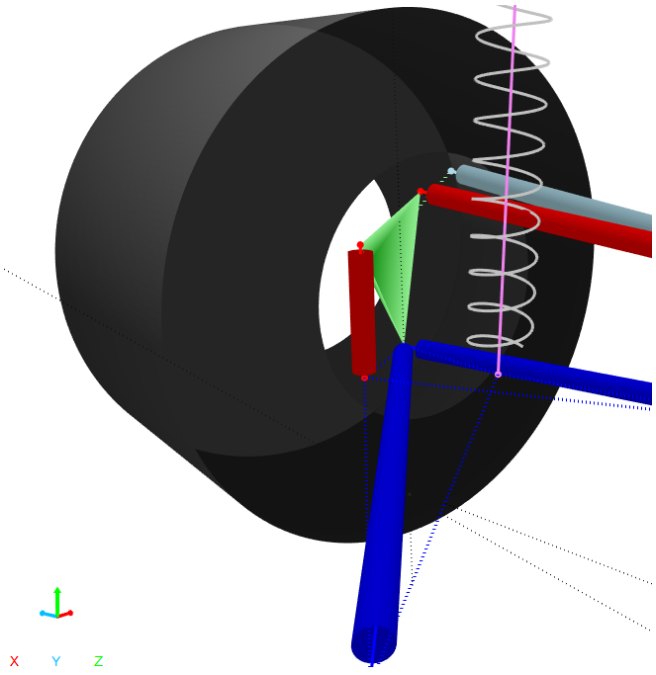
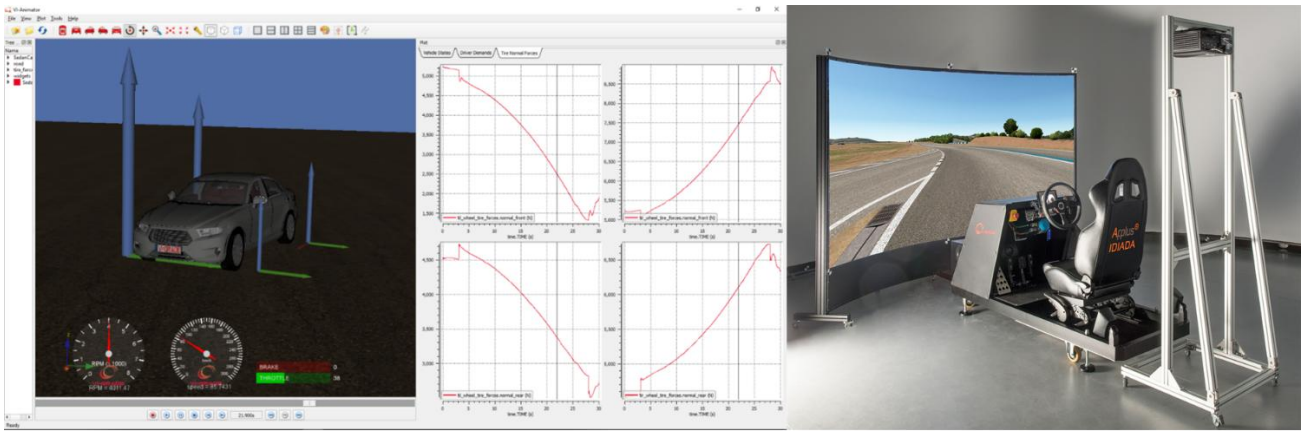


Figure 18. Integral link (vertical red link) in 3DSusp connecting the lower arm (blue) with the upright (green).

2.3 Integration with Vi-CarRealTime®

Vi-CarRealTime® is a vehicle dynamics simulation software; it represents the state of the art and is used professionally by major automakers. This software allows to create a virtual vehicle model; the architecture of the solver is based on a multibody model. A screenshot of the Vi-CarRealTime® GUI can be seen in the Figure 19. The company that produces this software, Vi-Grade is a world leader in building professional driving simulators. The University of Brescia bought one in 2019, specifically getting a compact simulator like the one shown in the Figure 19. With this simulator it is possible to drive in a simulated environment the vehicle model created in Vi-CarRealTime®.



a) b)

Figure 19. a) Vi-CarRealTime® graphic interface. b). Vi-Grade compact driving simulator.

Having developed suspension design software such as 3DSusp, the need arose to be able to integrate suspension kinematics easily and quickly into a virtual vehicle model in Vi-CarRealTime®.

Integration via Vi-Grade environment was done with two different mereologies in different steps. The first approach is the simplest from an implementation point of view but slower and less effective from a user experience point of view. A tool in the Vi-Grade package was exploited to do this: Vi-SuspensionGen®.

The second approach takes advantage of the fact that Vi-CarRealTime® allows its solver to be used in the Matlab-Simulink® environment so that external models can be utilized in co-simulation.

2.3.1 Integration through Vi-SuspensionGen®

A vehicle model in Vi-CarRealTime® is composed of several subsystems such as: body, powertrain, front and rear suspension, and so on. Each subsystem is characterized by a file with an .xml extension that describes all its properties, also including the curves that describe the kinematics of the suspension. The same goes for the file describing the subsystem "Steering."

Vi-CarRealTime® provides a tool called Vi-SuspensionGen® that allows the creation of .xml files needed for the vehicle model from a suspension defined by hardpoints, such as 3DSusp. Vi-SuspensionGen® allows the creation of different suspension architectures, such as those shown in the Figure 20.

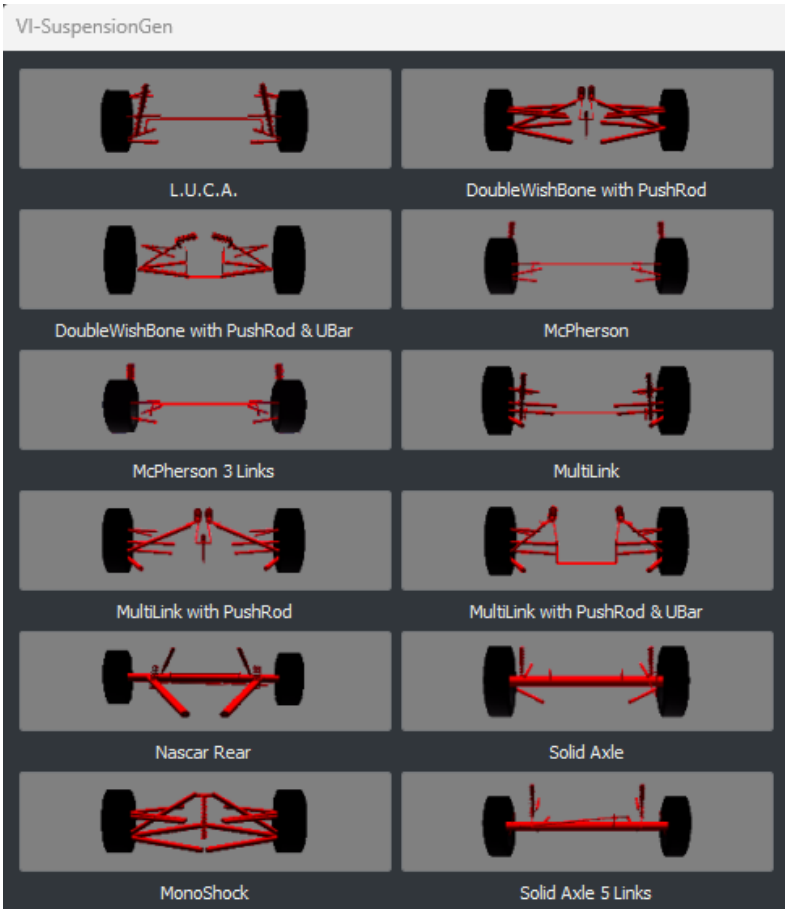


Figure 20. Available suspension model in Vi-SuspensionGen®.

Thus, in Vi-SuspensionGen® it is possible to import any type of suspension from 3DSusp except for the solid axle since it is a dependent wheel suspension and not provided for in 3DSusp. However, there are some configurations that can be modelled in 3DSusp that are not provided for in Vi-SuspensionGen®, such as the integral link discussed in the previous subchapter.

To get the two pieces of software talking, it was necessary to analyse the files with which Vi-SuspensionGen® manages and saves suspensions, which utilizes the .sgs extension. The necessary routines were then created in 3DSusp to be able to generate the .sgs files that can be read by Vi-SuspensionGen®, and through a simple button it is then possible to create the file, open it in Vi-SuspensionGen®, and generate the .xml files with the kinematic curve characteristics to be inserted into the vehicle model. As an example, a screenshot of how the .sgs file is formatted is shown in the Figure 21.

```
assemblyName     AssemblyDoubleWishBonePR
lonPosType                        front
unitsFile
                  length       mm
                   force       newton
end
leftSuspFile
                 suspType      DoubleWishBonePR
                 suspName      suspension1
               latPosType      left
               lonPosType      front
              wheelCenter                    0                -600               228.6
                  ucaFront                -179              -234.5                 324
                   ucaRear                 190                -251                 299
                  ucaOuter                   8                -495                 325
                  lcaFront                -248                 -70                 125
                   lcaRear                 210                 -70                 129
                  lcaOuter                  -8                -540                 139
               tierodOuter                  46                -484                 164
               tierodInner                 -81                -180                 155
              pushrodOuter                  11                -466              347.65
              pushrodInner                   4              -223.5                 562
               rockerPivot                 -45              -200.28              549.16
            rockerPivotAxis                -45                -233.4              586.62
                 rockerLink              -22.82                -120                 650
          springUpperMount                -206              -163.56              614.98
          springLowerMount                 -31              -163.56              614.98
          damperUpperMount                -206              -163.56              614.98
          damperLowerMount                 -31              -163.56              614.98
rockerLinkToCentralSpring                -82.3                -89.7                 485
rockerLinkToCentralDamper                -95.6                -73.9               490.1
   rockerLinkToRollDamper                -80.4               -101.6               437.2
      rockerLinkToInerter                -108.8                  -58               495.1
                  toeAngle                    0
               camberAngle                    0
                steerRatio               0.0974
              groundHeigth                   50
               wheelRadius                228.6
                wheelWidth                190.5
                  toeSetup                0.125
               camberSetup                -2.75
               casterSetup                    7
                 pinHeigth                  285
                rideHeigth                 25.4
                    kSpring                  330
             springPreload                 1550
                  cgHeight                  300
                 wheelBase                 3100
                driveRatio                    0
                brakeRatio                   70
            pRodAttachment                    1
                springType                    0
          springAttachment                    3
```

Figure 21. Screenshot showing the formatting of the readable .sgs file in Vi-SuspensionGen®.

2.3.2 Integration through Matlab-Simulink® Co-Simulation

Vi-CarRealTime® allows to use the solver in the Matlab-Simulink® environment thus being able to integrate logic or algorithms that work in co-simulation.

Vi-CarRealTime® precisely provides the possibility of using the so-called "External Suspension" and "External Steering" so that we can manage externally, in this in Simulink®, a suspension model with its steering mechanism. At each simulation step, the Vi-CarRealTime® solver needs to have information about the position and orientation of the wheel to correctly calculate the forces applied to the suspension and then to the chassis. It is necessary to provide information on the change in wheelbase, track, camber, toe and SVA (Side View Angle), which is the rotation of the suspension about its y-axis. Obviously, this data must be provided for each of the 4 wheels of the vehicle.

Vi-Grade's software provides detailed guidance on the inputs to be provided in order to work with an external suspension model. To be able to calculate how forces are transmitted to the chassis, thus taking into account anti-effects, jacking-force and so on, the first and second derivatives with respect to bump and steering must

be provided, for each suspension parameter. The first and second derivatives with respect to time are also needed.

The following is then a list of input variables to be provided, for each parameter (Variation of wheelbase and track, Toe, Camber and SVA) to the Vi-CarRealTime® solver to externally manage the suspension of a single wheel:

- InternalFlag
- Displacement
- Velocity
- Acceleration
- DerivativeWrtSwa
- DerivativeWrtJside
- DerivativeWrtJopp
- SecondDerivativeWrtSwaSwa
- SecondDerivativeWrtSwaJside
- SecondDerivativeWrtSwaJopp
- SecondDerivativeWrtJsideJside
- SecondDerivativeWrtJsideJopp
- SecondDerivativeWrtJoppJopp

Where InternalFlag is simply a boolean value to turn internal kinematics computation on or off, the value "0" must be provided to work with an external model.

Displacement is the value of the parameter at that instant, and Velocity and Acceleration represent the first and second derivatives of that value, respectively, with respect to time.

The other values represent first and second derivatives with respect to different values where Jside represents the jounce of the wheel itself, Jopp represents the jounce of the opposite wheel of the same axle, Swa represents the steering wheel angle. Wrt means "With respect to."

3DSusp models only independent-wheel suspensions, so that the value of a suspension parameter has no dependence on the jounce value of the opposite wheel, it follows that the derivatives with respect to Jopp will all be zero. Similar reasoning applies to the rear wheels; since they are not steered, the derivatives with respect to Swa will all be zero.

3DSusp's algorithm, natively written in VB.Net language has been rewritten in Matlab® language within a matlab-function that can be used within Simulink. This function receives as input the jounce of the individual suspension and the steering angle (for the front suspension) and returns, for each suspension parameter a vector containing the inputs to be provided to the Vi-CarRealTime® solver.

For the calculation of derivatives with respect to time, the Simulink® "Derivative" block has been used. Instead, the calculation of the other derivatives is simply done numerically by applying small jounce and steer values in the algorithm to then calculate the differential value for each individual parameter. The method is similar to that used for calculating the motion ratio of elements, described in previous chapters. For second derivatives, the small displacement is applied twice, and the differential is calculated on the values of the first derivative.

Shown in the Figure 22 is a screenshot of a corner of the Simulink® file where it is possible to see the matlab-function related to the left front wheel containing the 3DSusp resolution algorithm and its links to the Simulink block of the Vi-CarRealTime® solver.

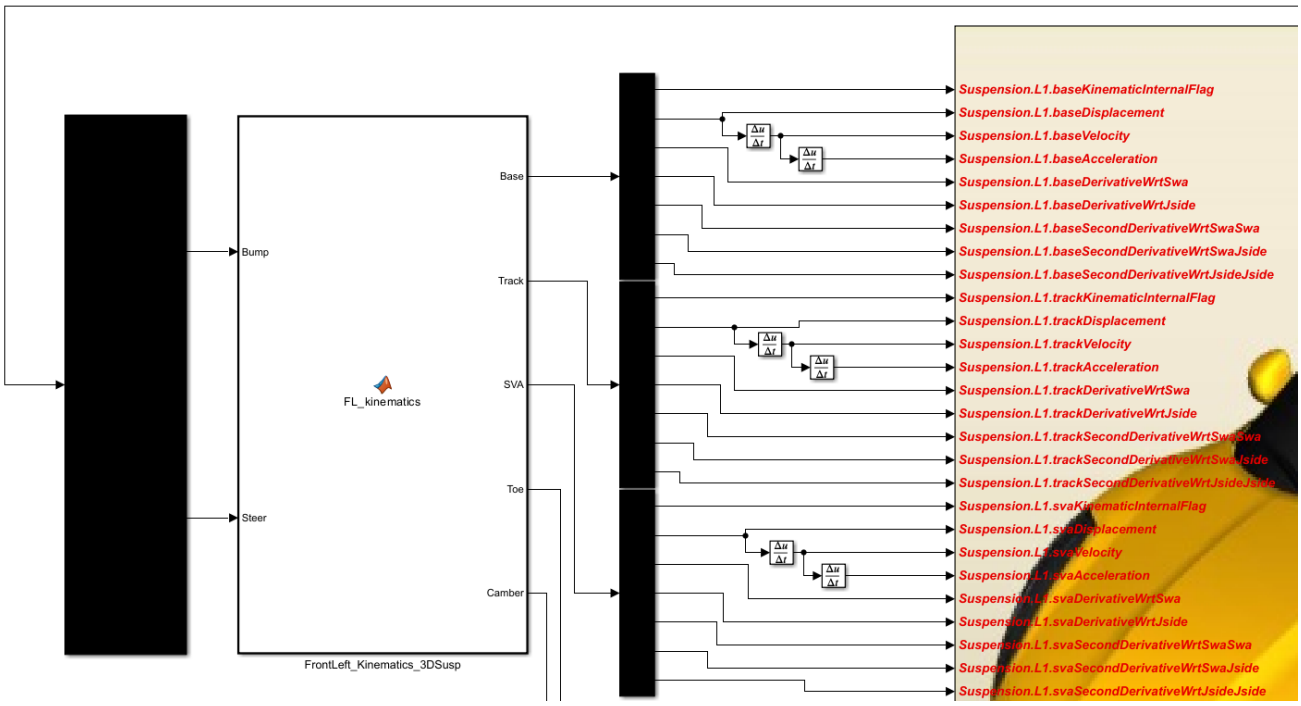


Figure 22. Screenshot of a corner of the Simulink® showing how the Matlab-function containing the 3DSusp algorithm is integrated into the Vi-CarRealTime® solver.

This method proved to be particularly powerful because the change of any point in the suspension can be evaluated instantaneously on the vehicle dynamics simulation software.

In addition, this method is particularly effective at the driving simulator because it even makes it possible to change any suspension parameter in real-time without having to stop the simulation and reload the model.

3 Suspension Elasto-kinematics and Loads

3.1 Suspension Loads

The algorithm for calculating the loads acting on the suspension is quite simple and is based on solving the equilibrium equations of the forces and moments. These equations are always linear, so it is only necessary to invert a matrix to solve.

By loads, more properly is meant the calculation of the constraint reactions acting on the suspension joints and eventually the axial force on the connecting rods, because of external forces applied on the wheel.

In general, two main cases can occur, depending on where the pushrod or spring is constrained:

- Pushrod/Spring attached to the upright
- Pushrod/Spring attached to a wishbone arm

The suspension model being considered is the most general one possible, namely the Multilink. The other suspension architectures are special configurations of the multilink where certain connecting rod joints are made to coincide with it. For example, a double wihsbone suspension is a multilink where the connecting rods converge two by two on the hub carrier.

Acronyms are defined below to facilitate understanding of the following subchapters. In particular, a name is given to each of the 6 connecting rods that make up the multilink:

- TLF = Top Link Front
- TLR = Top Link Rear
- BLF = Bottom Link Front
- BLF = Bottom Link Rear
- TR = TrackRod
- PR = PushRod/PullRod or Spring

To identify the points of each connecting rod, a letter is added to identify: wheel side (W) and frame side (C). For example, the frame side point of the front upper connecting rod will be $TLFC$. Its coordinates will be: $TLFC = (X_{TLFC}, Y_{TLFC}, Z_{TLFC})$.

To write the equations, a reference system with origin in the wheel centre and directions equal to the main reference system is considered. It is therefore necessary to change from an absolute to a relative coordinate system. This is done by using a transformation matrix that contains the direct cosines of the wheel reference system with respect to the global one.

The coordinates written in the equilibrium equations will always refer to the reference system integral with the wheel centre. The equilibrium at rotation will be written by choosing the wheel centre as the pole.

Next, the direct cosines are calculated for each connecting rod. Given two Points A and B, the direct cosines of segment AB are called: $c_{AB} = [cX_{AB}\ cY_{AB}\ Zc_{AB}]$

3.1.1 Pushrod or Spring attached to the upright

The first case is the simplest, because all the suspension arms are, in this model, considered to be connecting rods, thus subjected only to axial action. Thus, it is sufficient to calculate the equilibrium equations on the hub-carrier to be able to know all the constraint reactions.

A schematic of the model with PR constrained to the hub carrier can be seen in the Figure 23, where the yellow arrows represent the axial forces of each connecting rod.

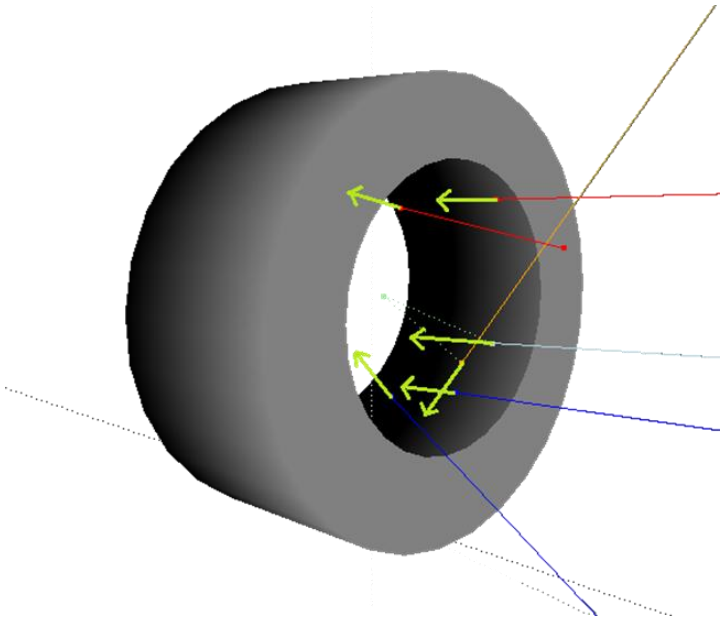


Figure 23. Schematic of the model with PR constrained to the hub carrier, the yellow arrows represent the axial forces of each connecting rod.

The direction of the axial force of each connecting rod is known, as it coincides with the vector of the directing cosines that identify the connecting rod in the reference system integral to the wheel.

The equations of equilibrium on the hub carrier are written in matrix in eq. 3.1.

$$[M] \cdot [R] = [F] \tag{3.1}$$

Where $[M]$ is described in eq. 3.2. Due to space requirements, subscripts have been omitted in the last three rows of the matrix, but they intuitively refer to those in the very column that identifies them.

$$\begin{bmatrix} cX_{TLF} & cX_{TLR} & cX_{BLF} & cX_{BLR} & cX_{TR} & cX_{PR} \\ cY_{TLF} & cY_{TLR} & cY_{BLF} & cY_{BLR} & cY_{TR} & cY_{PR} \\ cZ_{TLF} & cZ_{TLR} & cZ_{BLF} & cZ_{BLR} & cZ_{TR} & cZ_{PR} \\ cZ \cdot Y_W - cY \cdot Z_W & cZ \cdot Y_W - cY \cdot Z_W & cZ \cdot Y_W - cY \cdot Z_W & cZ \cdot Y_W - cY \cdot Z_W & cZ \cdot Y_W - cY \cdot Z_W & cZ \cdot Y_W - cY \cdot Z_W \\ cX \cdot Z_W - cZ \cdot X_W & cX \cdot Z_W - cZ \cdot X_W & cX \cdot Z_W - cZ \cdot X_W & cX \cdot Z_W - cZ \cdot X_W & cX \cdot Z_W - cZ \cdot X_W & cX \cdot Z_W - cZ \cdot X_W \\ cY \cdot X_W - cX \cdot Y_W & cY \cdot X_W - cX \cdot Y_W & cY \cdot X_W - cX \cdot Y_W & cY \cdot X_W - cX \cdot Y_W & cY \cdot X_W - cX \cdot Y_W & cY \cdot X_W - cX \cdot Y_W \end{bmatrix} \tag{3.2}$$

And the vector R is described in the eq. 3.3. This vector contains the axial actions of each of the connecting rods.

$$[R] = \begin{bmatrix} R_{TLFW} \\ R_{TLRW} \\ R_{BLFW} \\ R_{BLRW} \\ R_{TRW} \\ R_{PRW} \end{bmatrix} \tag{3.3}$$

And the vector F is described in the eq. 3.4. This vector represents the forces and moments eging in the wheel centre, thus in the origin of the reference system integral to the hub carrier. By calculating the carrier moments appropriately, it is possible to change the point of application of the forces.

$$[F] = -\begin{bmatrix} F_X \\ F_y \\ F_Z \\ M_X \\ M_Y \\ M_Z \end{bmatrix} \tag{3.4}$$

To calculate the unknown vector R, then the axial actions on each connecting rod, it is sufficient to solve eq. 3.5.

$$[R] = [M]^{-1} \cdot [F] \tag{3.5}$$

The axial actions on each connecting rod, because of the way the system is written, will be positive if the connecting rod is under tension and negative if it is compressed, in a manner concordant with convention.

Given the axial actions, to calculate the wheel-side constraint reactions $[RC]$ along the X, Y and Z directions, simply multiply the value of the action by the corresponding director cosine. The frame-side reactions will be equal but of opposite sign.

3.1.2 Pushrod or Spring attached to a wishbone arm

In this case, if the pushrod or spring is connected to a wishbone, the wishbone can no longer be considered to consist of two connecting rods subject to axial force alone. It will then be necessary to write another set of equations with respect to the previous subchapter in order to know the loads acting on the arm to which the pushrod is bound.

However, the system on the hub carrier turns out to be solvable. It turns out to be connected by 3 connecting rods plus the triangle (3 reactions in the 3 XYZ directions), the structure still remains isostatic and solvable.

Going to analyse the triangle to which the pushrod is connected, we see that the degrees of constraint are 7: 3 for each frame-side hinge and a constraint given by the Pushrod/Pullrod.

The system results kinematically hyperstatic. To solve it accurately, the stiffness of the materials must be considered.

An approximation is made to solve this problem. The concept is that the two joints constrained to the triangle frame, generate a redundant degree of constraint. In fact, the triangle during its movement will always rotate about an axis that is defined by the line passing through the two frame-side joints. The two spherical joints both provide a constraint on translation along this axis, when only one would suffice. A seventh equation is then written that binds the components along the axis of rotation of the triangle so that the system is solvable. This equation takes into account the fact that these two components are distributed according to the stiffness of the joints along the axis of rotation, like two springs in parallel. The equation will be described later; in the Figure 24 it is possible to see an example of a suspension with pushrod constrained to the lower arm and see the two components along the axis mentioned earlier.

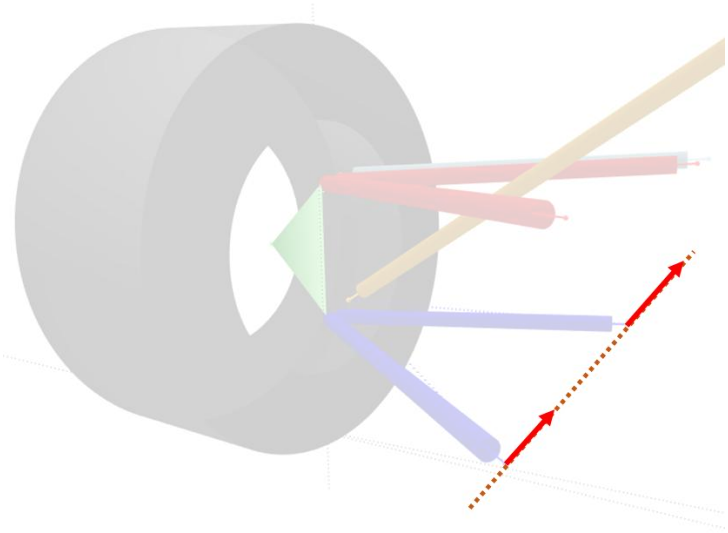


Figure 24. Components along the axis of rotation of the wishbone to which the hub carrier is attached.

Calculations for a suspension with a pushrod constrained to the lower arm are given below. For the upper wishbone, however, the same logic is applied. The first procedure is to calculate the constraint reactions on the hub carrier, which is now constrained with 3 connecting rods (2 for the wishbone and one for the trackrod) and a ball joint (representing the wishbone with the constrained pushrod). The Figure 25 represents the wheel-side constraint reactions.

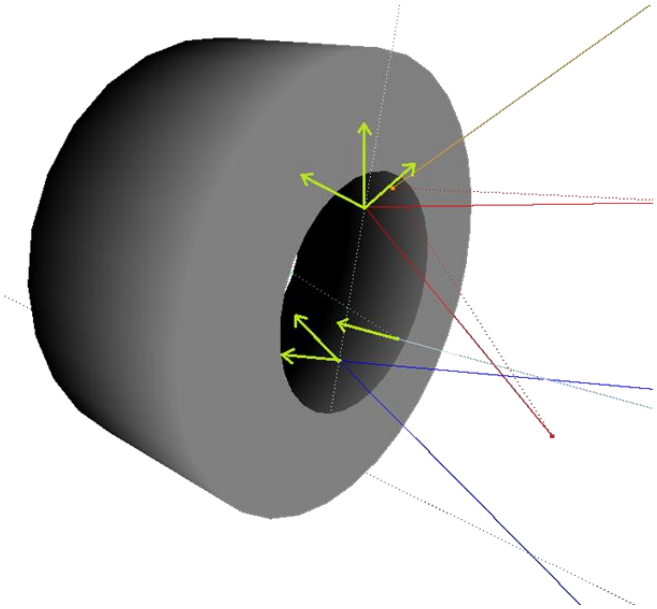


Figure 25. Schematic of the model with PR constrained to upper arm, the yellow arrows represent constraint reaction forces. In low the three direct axial actions like connecting rods. On top three orthogonal reactions on the connection with the upper wishbone.

As with the case seen in the previous subsection, the equilibrium reactions on the hub carrier are written in the eq. 3.6.

$$[M] \cdot [R] = [F] \tag{3.6}$$

Where $[M]$ is described in eq. 3.7. Due to space requirements, subscripts have been omitted in the last three rows of the matrix, but they intuitively refer to those in the very column that identifies them.

$$\begin{bmatrix} 1 & 0 & cX_{BLF} & cX_{BLR} & cX_{TR} & 0 \\ 0 & 1 & cY_{BLF} & cY_{BLR} & cY_{TR} & 0 \\ 0 & 0 & cZ_{BLF} & cZ_{BLR} & cZ_{TR} & 1 \\ 0 & -Z_{TLFW} & cZ \cdot Y_W - cY \cdot Z_W & cZ \cdot Y_W - cY \cdot Z_W & cZ \cdot Y_W - cY \cdot Z_W & Y_{TLFW} \\ Z_{TLFW} & 0 & cX \cdot Z_W - cZ \cdot X_W & cX \cdot Z_W - cZ \cdot X_W & cX \cdot Z_W - cZ \cdot X_W & -X_{TLFW} \\ -Y_{TLFW} & X_{TLFW} & cY \cdot X_W - cX \cdot Y_W & cY \cdot X_W - cX \cdot Y_W & cY \cdot X_W - cX \cdot Y_W & 0 \end{bmatrix} \tag{3.7}$$

And the vector R is described in the eq. 3.8. This vector contains the axial actions of each of the connecting rods.

$$[R] = \begin{bmatrix} R_X \\ R_Y \\ R_{BLFW} \\ R_{BLRW} \\ R_{TRW} \\ R_Z \end{bmatrix} \tag{3.8}$$

Where $R_X, R_Y \; e \; R_Z$ are the reactions given by the upper wishbone.

And the vector F is described in the eq. 3.9. This vector represents the forces and moments acting on the wheel centre, thus in the origin of the reference system integral to the hub carrier. By calculating the carrier moments appropriately, it is possible to change the point of application of the forces.

$$[F] = -\begin{bmatrix} F_X \\ F_y \\ F_Z \\ M_X \\ M_Y \\ M_Z \end{bmatrix} \tag{3.9}$$

To calculate the unknown vector R, then the axial actions on each connecting rod, it is sufficient to solve eq. 3.10.

$$[R] = [M]^{-1} \cdot [F] \tag{3.10}$$

Now the frame side reactions of the lower arm and steering tierod are known; to derive the frame side constraint reactions of the upper triangle and to calculate the axial action acting on the Pushrod we proceed to write the equilibrium equations on the upper wishbone. Plus, a final equation that binds the components along the axis of rotation of the wishbone, of the frame-side constraint reactions.

The triangle thus has 6 frame-side constraint reactions given by the ball joints plus the reaction given by the Pushrod, which is considered as a connecting rod. It is possible to see these reactions in the Figure 26, in light blue. In yellow the wheel-side constraint reactions, calculated just now.

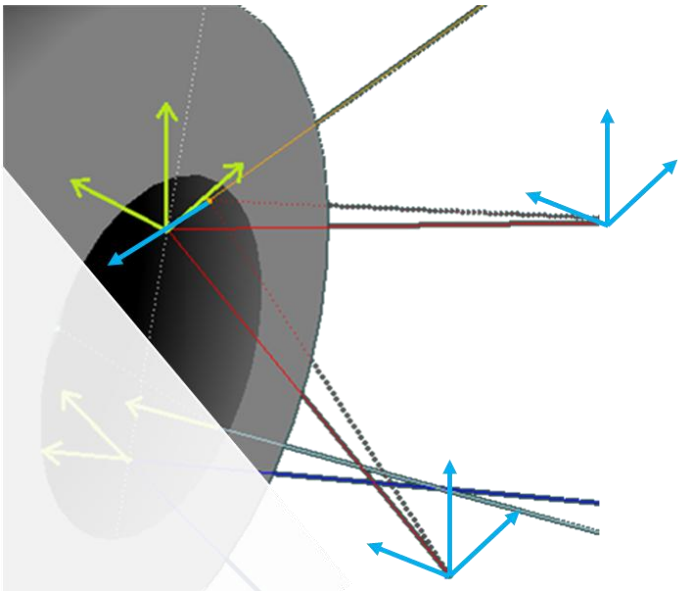


Figure 26. Diagram of the forces acting on the upper arm, in light blue the constraint reactions given by the frame-side joints and the pushrod. In yellow the wheel-side constraint reactions.

As mentioned earlier, in order to be able to solve the system, it is necessary to add a seventh equation to the 6 of equilibrium that binds the frame-side constraint reactions together. Specifically, an equation is written that distributes the components along the axis of rotation proportionally to the translational stiffness of the joints along this direction. In particular, the ratio kr is defined in eq. 3.11.

$$kr = \frac{kTLF_{ax}}{kTLF_{ax} + kTLR_{ax}} \tag{3.11}$$

Where $kTLF_{ax}$ and $kTLR_{ax}$ are the stiffness along the axis of rotation of the arm, which is denoted by ax. In the next equations there will also be written the direct cosines of this axis, denoted as cX_{ax}, cY_{ax} and cZ_{ax}.

In case in two frame-side joints are two equal uniballs, the stiffnesses will be the same, so the value of kr will be equal to 0.5 and the components along the axis will be equal.

Continuing, we then write the equation in matrix form to solve the system in eq. 3.12.

$$[R_{Arm}] = [M_{Arm}]^{-1} \cdot [F_{Arm}] \tag{3.12}$$

Where the matrix M_{Arm} is defined in equation 3.13.

$$\begin{bmatrix} cX_{PR} & 1 & 1 & 0 & 0 & 0 & 0 \\ cY_{PR} & 0 & 0 & 1 & 1 & 0 & 0 \\ cZ_{PR} & 0 & 1 & 0 & 1 & 1 & 1 \\ cZ \cdot Y_W - cY \cdot Z_W & 0 & 0 & -cZ_{TLF} & -cZ_{TLR} & cY_{TLF} & cY_{TLR} \\ cX \cdot Z_W - cZ \cdot X_W & cZ_{TLF} & cZ_{TLR} & 0 & 0 & -cX_{TLF} & -cX_{TLR} \\ cY \cdot X_W - cX \cdot Y_W & -cY_{TLF} & -cY_{TLR} & cX_{TLF} & cX_{TLR} & 0 & 0 \\ 0 & cX_{ax} \cdot 2(1-kr) & cY_{ax} \cdot 2(1-kr) & cZ_{ax} \cdot 2(1-kr) & -cX_{ax} \cdot 2kr & -cY_{ax} \cdot 2kr & -cZ_{ax} \cdot 2kr \end{bmatrix} \tag{3.13}$$

The matrix M_{Arm} is a 7x7. Where the first 6 rows represent the 6 equilibrium equations, the last equation instead binds the frame-side constraint reactions as explained above.

The Vector F_{Arm} is defined in the eq. 3.14.

$$[F_{Arm}\,] = \begin{bmatrix} R_X \\ R_y \\ R_z \\ R_Z \cdot Y_{TLFW} - R_Y \cdot Z_{TLFW} \\ R_X \cdot Z_{TLFW} - R_Z \cdot X_{TLFW} \\ R_Y \cdot X_{TLFW} - R_X \cdot Y_{TLFW} \\ 0 \end{bmatrix} \tag{3.14}$$

The first 3 elements of the vector represent the wheel-side constraint reactions calculated earlier, the other 3 values are the moments generated by these forces, calculated with respect to the origin of the reference system. The last value imposes equality to 0 for the seventh equation.

The Vector R_{Arm} is defined in Eq. 3.15.

$$[R_{Arm}] = \begin{bmatrix} R_{PRW} \\ X_{TLFC} \\ X_{TLRC} \\ Y_{TLFC} \\ Y_{TLRC} \\ Z_{TLFC} \\ Z_{TLRC} \end{bmatrix} \tag{3.15}$$

Solving the eq. 3.10 all frame-side constraint reactions are then known. The frame-side reactions of the pushrod are the same as the arm-side reactions, but of opposite sign since this is a connecting rod.

3.1.3 Example and Comparison with Lotus Shark®

This algorithm was implemented in Matlab®; Figure 27 show an example suspension subjected to loads. A simple graphical interface was created so that all constraint reactions can be visualized, and it can be understood which ones have a higher magnitude.

A double wishbone suspension was used in this example case. Each constraint reaction is displayed with 3 lines, each of a colour representing a component of the triad. The length of the lines is proportional to the magnitude of the force; finally, numerical values are also represented.

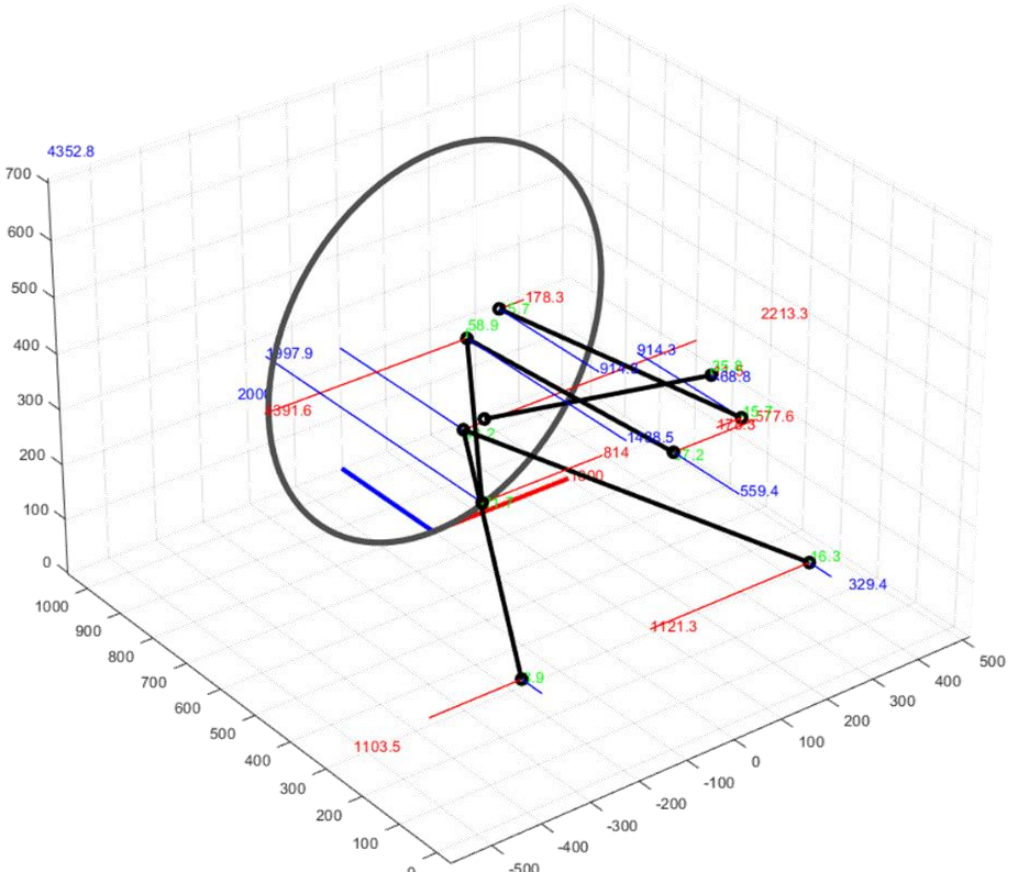


Figure 27. Graphical interface of the output of the calculation algorithm for loads on the suspension.

To validate the results, a comparison with a commercial software used in the automotive field, called Lotus Shark,® [51] was chosen. For any type of suspension and load case, the results were always identical. Two

screenshots showing the calculated loads on the suspension can be seen in Figure 28. The suspension chosen is a double wishbone with pushrod constrained to the lower triangle; a load case of $500N$ of Fx under braking and $2000N$ of Fz is shown in Figure 28. Looking closely at the numbers, the results are identical.

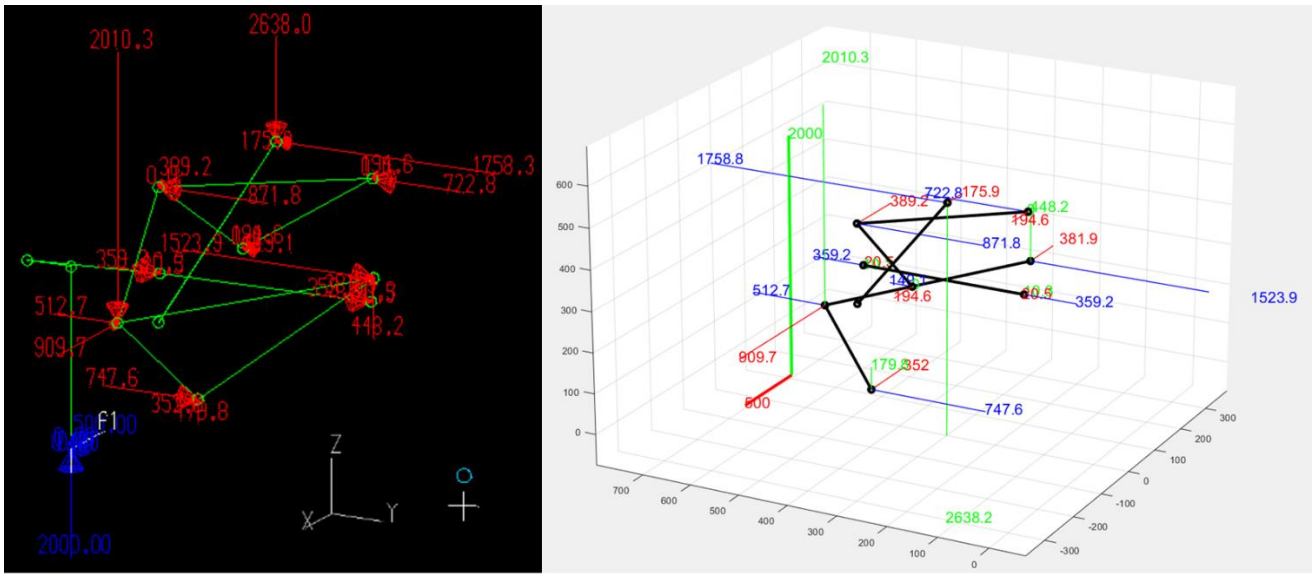


Figure 28. Screenshot representing the loads acting on the same suspension. a) Lotus Shark®. b) Algorithm presented visualized in Matlab®.

3.2 Translational Stiffness Elasto-kinematics

The model for calculating elasto-kinematics presented in this chapter was developed following the algorithm for calculating loads on the suspension. In fact, it is closely related to it. The goal of this algorithm is to define a suspension, giving as input the coordinates of the points, the stiffnesses of the joints and a vector with external forces and moments applied on the suspension and then having as output the deformed configuration of the suspension.

In this model, the ball joints are replaced with joints having translational compliance, that is, bushing deformable along the three Cartesian directions, as in the Figure 29.

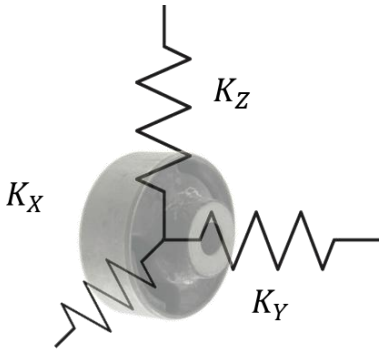


Figure 29. Model of the bushing in this algorithm. Characterized by three stiffnesses along the axes.

The suspension spring is considered infinitely stiff, and we focus only on the compliance provided by the bushings. Stiffnesses are considered linear, so each bushing is represented by 3 stiffness values expressed in N/mm. Furthermore, each bushing, in addition to being defined by its position with 3 coordinates, is also defined by the 3 Euler angles that define its orientation in the global reference system. In fact, the 3 rigidities are defined in the bushing local reference system (bLRS) using the vector $k^{bLRS} = [kx\ ky\ kz]$.

To calculate the rigidities in the local reference system, it is sufficient to calculate the rotation matrix Rb, a function of the 3 angles $\theta_x, \theta_y, \theta_z$ and calculate the new vector of rigidities as in the eq. 3.16.

$$k^{GRS} = R \cdot k^{bLRS} \tag{3.16}$$

The rotation matrix Rb is calculated with a *Tait-Bryan* transformation (4) $X_1 Z_2 Y_3$.

$$Rb = [rotx(\theta_x)] \cdot [rotz(\theta_z)] \cdot \left[roty(\theta_y)\right] \tag{3.17}$$

Where:

$$[rotx(\alpha)] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,\alpha & -sin\,\alpha \\ 0 & sin\,\alpha & cos\,\alpha \end{bmatrix} \tag{3.18}$$

$$[roty(\alpha)] = \begin{bmatrix} cos\,\alpha & -sin\,\alpha & 0 \\ sin\,\alpha & cos\,\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.19}$$

$$[rotz(\alpha)] = \begin{bmatrix} cos\,\alpha & 0 & sin\,\alpha \\ 0 & 1 & 0 \\ -sin\,\alpha & 0 & cos\,\alpha \end{bmatrix} \tag{3.20}$$

In this model, the rotational stiffnesses of the bushing are not considered, only the translational ones. In fact, in this algorithm the loads acting on the joints are first calculated using the algorithm explained in the previous subsection. After that the positions of the joints are slightly modified as a function of the constraint reaction and stiffness, the deformation vector u^{GRS} in the eq. 3.21 is thus defined for each joint in the GRS.

$$u^{GRS} = -L^{GRS} \oslash k^{GRS} \tag{3.21}$$

Where L^{GRS} is the 3x1 vector representing the calculated load on the specific bushing in the GRS and $\oslash$ is the operator that define a vector division element by element. The vector u^{wLRS} is also defined in eq. 3.22 which always defines a displacement vector, but in wheel local reference system (wLRS).

$$u^{wLRS} = Rw \cdot u^{GRS} \tag{3.22}$$

Where Rw is the rotation matrix that defines the orientation of the wheel in the GRS, a function of camber, toe and side view angle. Calculated similarly to that seen in Eq. 3.17. Specifically, the vector u^{GRS} will be calculated for each frame-side joint, while the vector u^{wLRS} will be calculated for each wheel-side joint, that

is, those bushing that connect the arms to the hub carrier. These strain vectors are always referred to the position of the joints in the initial configuration.

At this point, the position of the frame-side points is updated by translating them by the vector u^{GRS}. The coordinates relative to the wheel, then in the wLRS, are also updated by translating them by the vector u^{wLRS}.

Having shifted the points, it is now necessary to run the kinematics calculation, the same one used in the 3DSusp algorithm, remembering that as input there are just the positions of the frame-side joints in the GRS and the wheel-side joints in the wLRS. The calculation is done again using the Newton-Raphson algorithm by finding the new position of the hub carrier (thus also of the wheel-side joints) by imposing that the 6 connecting rods that make up the suspension have the same length as the one set at the beginning.

The u vectors are saved, and the calculation of the loads is now carried out again. New strain vectors u for each joint are then calculated. An $error$ variable, defined as the norm of the difference between the current and previous strain vectors, is calculated, as in the eq. 3.23.

$$error = \|u_{i-1} - u_i\| \tag{3.23}$$

In this way a loop can be constructed, where the suspension is deformed according to the loads, and at each iteration the loads are recalculated on the deformed configuration until the deformation vector changes little from that calculated in the previous iteration. That is, when the variable error is less than a small arbitrary value such as 0.001. Once the cycle exit condition is met, the suspension has reached an equilibrium configuration, and the new joint coordinates, wheel position, and all constraint reactions can be output. To better understand the algorithm explained, a flowchart is shown in the Figure 30.

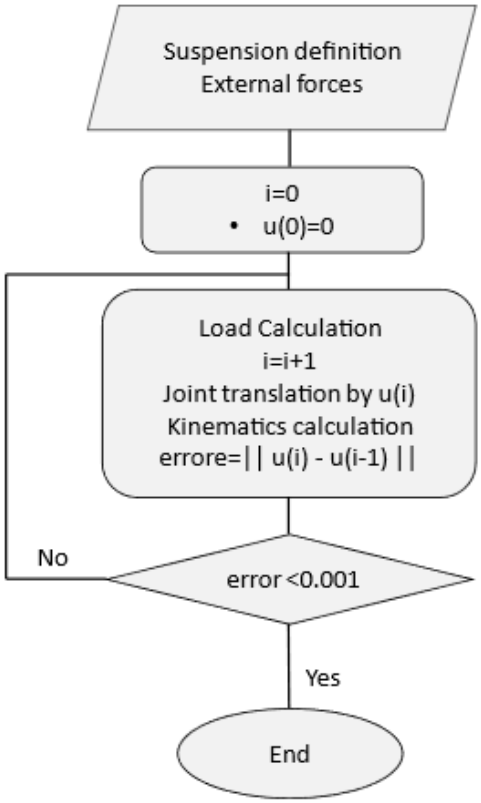


Figure 30. Simplified flowchart of the algorithm for calculating elastokinematics with translational stiffnesses.

The algorithm was implemented in Matlab® and a simple graphical interface was created to see the deformed configuration of the suspension after the load was applied to it. An example can be seen in the Figure 31 where loads of $F_X = 1000N$ and $F_Y = 2000N$ were applied on a generic suspension.

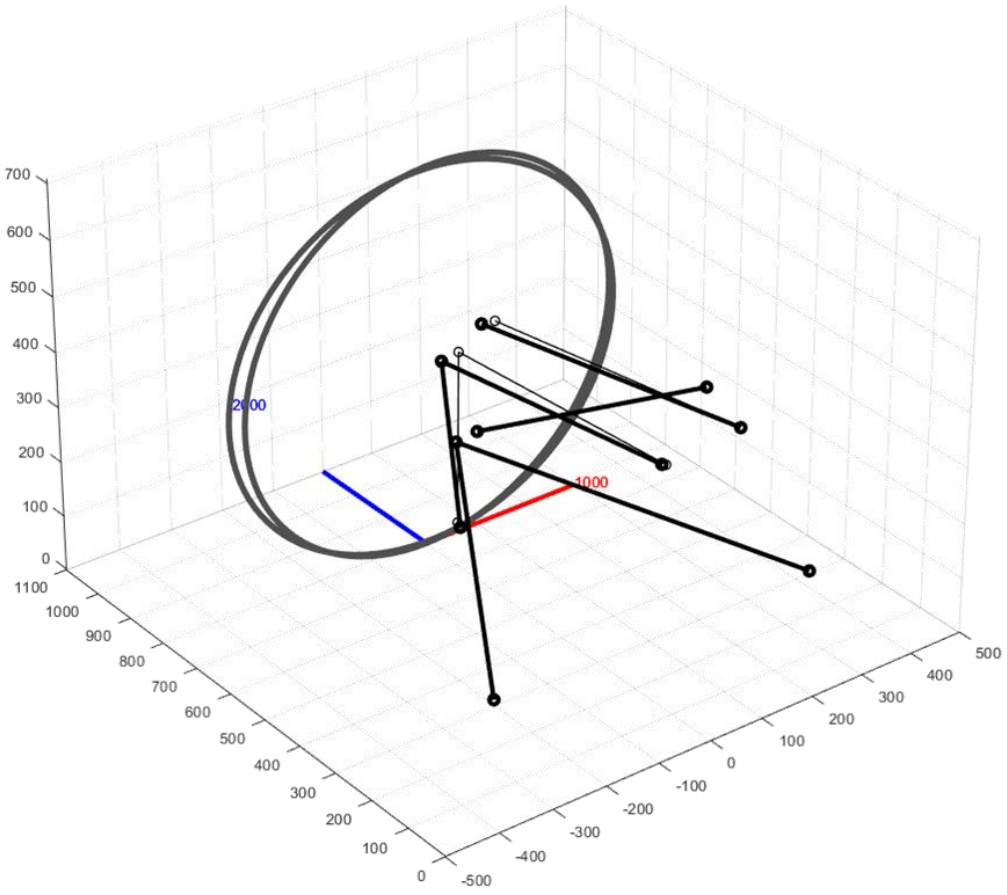


Figure 31. Matlab® interface screenshot showing the deformed configuration of the suspension, superimposed on the design configuration, after external forces are applied. In bold the deformed configuration

The development of this algorithm, which had some limitations, was prophetic for later ones. This model is very computationally fast and can be used for passenger cars where the bushings are very compliant.

Finally, this tool allows the calculation of suspension elasto-kinematics not only in the static suspension configuration; in fact, it is possible to use a suspension in steering and/or bump configuration as input.

3.3 Elasto-Kinematics Comprehensive Model

The development of this algorithm occupied most of the author's time during the PhD course. And it was probably the most significant one.

Purpose of this work is to provide a method for the solution of the suspension elastokinematic problem, by means of a general method enabling the design of most of the modern layouts, from simple semi-trailing arms to all kinds of suspensions with a virtual instantaneous axis of rotation in steering or under load, including the so-called integral link geometry. Bushings can be described with their stiffness properties for all six degrees of freedom and with real-world, non-linear curves. They can be located either side of each suspension arm i.e. on the chassis side and/or on the wheel side. The flexibility of trackrod can also be represented by means of equivalent, non-linear bushings. The wheel bearing stiffness can be considered as well. Wheel movement hence variations of vehicle dynamics-relevant parameters like camber, side view angle, toe, track, wheelbase

and vertical displacement can be computed under any combination of road loads: braking, traction, cornering and bump impacts. Any combination of jounce and steer motion can be simulated within the whole range allowed by bump and rebound stop. Suspension's joints and chassis loads can also be computed.

In this model, the suspension of a passenger car equipped with compliant bushings is considered. The compliance of the suspension arms is considered negligible, as it is much lower than that of the deformable joints.

The proposed algorithm has been presented in a general mathematical formulation; however, it has been implemented in the Matlab environment. This environment was chosen as it specialises in matrix calculation and allows the programmer to write vectors, matrices, and operations between them in an intuitive way, and is characterised by good computational speed despite being a high-level language.

The Matlab environment also allows interfacing with Simulink, enabling the user to implement other functions, e.g. implementing optimisation logic or sensitivity analysis.

However, Matlab is not freely accessible, and this may be the main limitation. An excellent alternative could be Python, using existing mathematical libraries. Another alternative, which would guarantee higher computation speeds than Matlab, could be to implement the code in C++, although this is less intuitive to code.

3.3.1 Model Description

In this model, two types of elements are considered. The first type of element is called "spring rod" and it is composed by two bushings at the ends and a rod with an axial stiffness. The second element is called "rigid element" and it is composed by a rigid body with any number of bushings. These elements can be seen in Figure 32.

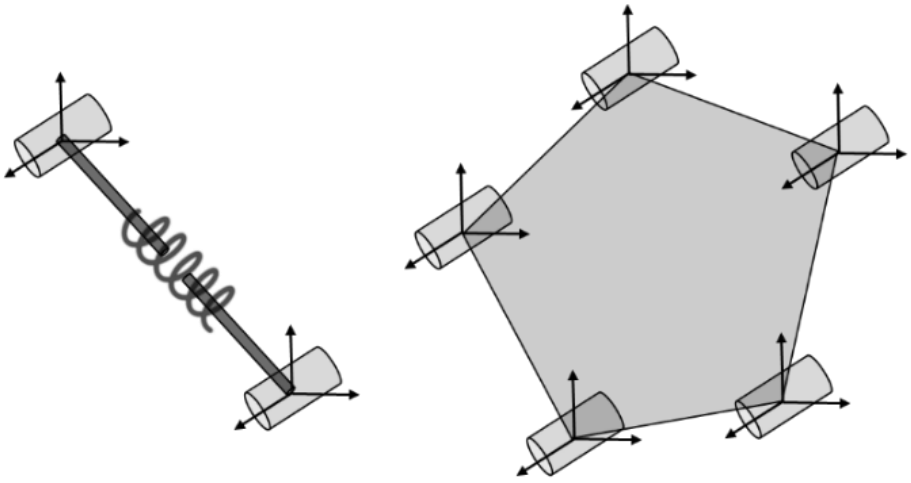


Figure 32. General "spring rod" and "rigid element" of the suspension model.

Any element can be attached to each other element or to the ground. The elements connected to each other share a bushing. Otherwise, the element is attached to the ground by linking its bushing to the global reference system (GRS) which represents the body chassis of the vehicle.

By combining these elements, it is possible to create any type of independent suspension. For example, in a double wishbone, the two wishbones will consist of a rigid element connected to the chassis by two bushings and to the upright by one bushing. The upright will also be a rigid element with three bushings, two connected to the wishbones and one to the steering tie rod; this one will instead be modelled using a spring rod element. A Multilink suspension can be easily modelled using five spring rods for the arms and a rigid element with five bushings for the hub carrier. A MacPherson strut can also be modelled by constraining a spring rod between the chassis and the hub carrier and making the bushing that connects them infinitely rigid. It is also possible to create an integral link suspension attaching a spring rod between the wishbone and the upright.

Finally, to calculate the elastokinematics configuration under lateral load more realistically, it is possible to steer the suspension by moving the attachment point on the chassis side of the steering rod.

By playing with bushing stiffnesses, it is also possible to exclude the elastic component. For example, by increasing the translation stiffness by a few orders of magnitude and decreasing the rotation stiffness, it is possible to turn the bushing into ball joints.

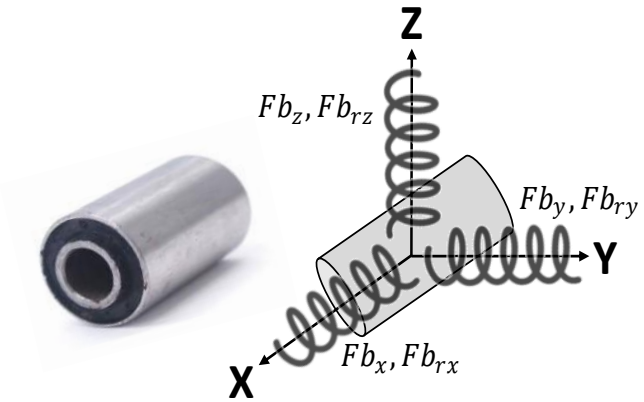


Figure 33. Bushing model with linear and rotational stiffnesses.

Each bushing must be defined with its position $Xb = [xb\ yb\ zb]$ in the GRS and with three angles $\theta b = [\theta b_x\ \theta b_y\ \theta b_z]$, with which it is therefore possible to define a rotation matrix $[R]$ to identify the orientation of the bushing in the GRS. The bushing is defined by a function "*Reaction Forces Vector*" in eq. 3.24 that contains six functions that correspond to the three reaction forces and the three reaction moments generated by the bushing as a function of the six deformations ub^{LRS} of the bushing itself.

$$Fb(ub^{LRS})^{LRS} = \begin{bmatrix} Fb_x(ub_x) \\ Fb_y(ub_y) \\ Fb_z(ub_z) \\ Fb_{rx}(ub_{rx}) \\ Fb_{ry}(ub_{ry}) \\ Fb_{rz}(ub_{rz}) \end{bmatrix} \tag{3.24}$$

Where "LRS" stays for *Local Reference System*. Both the deformations and the reaction forces vector are defined in the LRS in this case. Knowing the rotation matrix $Rb(\theta b)$ that identifies the orientation of the bushing, it is possible to calculate the reaction force in the *global reference system* GRS in eq.3.25.

$$Fb(ub)^{GRS} = \left[Rb_{exp}\right] \cdot Fb\left(\left[Rb_{exp}\right]^T \cdot (ub^{GRS})\right)^{LRS} \tag{3.25}$$

Where $[Rb_{exp}]$ is the *expanded* rotation matrix defined in eq. 3.26.

$$\left[Rb_{exp}(\theta b)\right] = \begin{bmatrix} [Rb(\theta b)] & 0 \\ 0 & [Rb(\theta b)] \end{bmatrix} \tag{3.26}$$

The rotation matrix Rb is calculated with a *Tait-Bryan* transformation $X_1 Z_2 Y_3$ in eq. 3.27.

$$Rb(\theta b) = [rotx(\theta b_x)] \cdot [rotz(\theta b_z)] \cdot \left[roty(\theta b_y)\right] \tag{3.27}$$

Where:

$$[rotx(\alpha)] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,\alpha & -sin\,\alpha \\ 0 & sin\,\alpha & cos\,\alpha \end{bmatrix} \tag{3.28}$$

$$[roty(\alpha)] = \begin{bmatrix} cos\,\alpha & -sin\,\alpha & 0 \\ sin\,\alpha & cos\,\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.29}$$

$$[rotz(\alpha)] = \begin{bmatrix} cos\,\alpha & 0 & sin\,\alpha \\ 0 & 1 & 0 \\ -sin\,\alpha & 0 & cos\,\alpha \end{bmatrix} \tag{3.30}$$

Each function of Fb can be any function of the deformation ub, even non-linear.

For the spring rod element, a function $Fa(ua)$ is also defined that represents the axial elastic force as a function of the axial deformation of the element. It can also be any function, just like for the bushings. The force-displacement curve that characterizes the spring element can be generated at will. It is also possible to insert a preload value as a displacement. Each *i-th* element of the suspension is described by a vector $u_i^{GRS} = [u_x\ u_y\ u_z\ u_{rx}\ u_{ry}\ u_{rz}]$ containing the displacement of one of its bushings from its design position and by its axial displacement ua_i if it is a *spring rod* element. The displacement vector of the other bushings of the element can be easily calculate trough transformation matrix because the element is a rigid body. To calculate

the deformed configuration of the suspension is necessary to know all the deformation vector and axial deformation of each element. The vector U containing these vectors is defined in eq. 3.31.

$$U = [u_1^{GRS}\ u_2^{GRS}\ \dots\ u_n^{GRS} ua_1\ ua_2 \dots ua_n] \tag{3.31}$$

To know the vector u, it is necessary to solve a system of N equations. Where N depends on the number of elements as in eq. 3.32.

$$N = 6 \cdot N_{rigid\ element} + 7 \cdot N_{spring\ rod} \tag{3.32}$$

On each rigid element is possible to apply an external force on any point of the body. It is necessary to define a vector $Fext_i$ as in eq. 3.33 and its application point $XFext_i$.

$$Fext_i = [Fext_x\ Fext_y\ Fext_z\ Mext_x\ Mext_y\ Mext_z] \tag{3.33}$$

To solve the equation $f(U) = 0$ a Newton-Raphson method (3.34) for multiple equation has been used to manage the non-linearity of the equations. This method is iterative, and the solution is accepted when $max(abs(f(U)))$ is less than ε which a small number.

$$U_{iter+1} = U_{iter} - [J(U_{iter})]^{-1} \cdot f(U_{iter}) \tag{3.34}$$

Where $U_0 = [0\ 0 \dots 0]$ and $[J(U_{iter})]$ is the Jacobian matrix, defined in eq. 3.35, of the function $f(U)$ calculated in U_{Iter}.

$$[J] = \begin{pmatrix} \frac{\partial f_1(U)}{\partial U_1} & \cdots & \frac{\partial f_1(U)}{\partial U_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_N(U)}{\partial U_1} & \cdots & \frac{\partial f_N(U)}{\partial U_N} \end{pmatrix} \tag{3.35}$$

The Newton-Raphson algorithm works well if the solution is sufficiently close to the initial condition. For this reason, it may happen that the solver does not converge, particularly in cases where there are high forces or low stiffnesses. One way to overcome this problem has been to break the resolution algorithm into multiple iterations, progressively increasing the vector of external forces and the *spring rod* element's preload if present.

The function $f(U)$ is a set of N functions, same dimensions of the displacement vector U. The first six functions (3.36) (3.37) represent the equilibrium equation of the system where the external forces and moments must be equal to the constraint reaction generated by the bushings which is attached to the ground and by the axial reaction of the spring rod element which has one of the bushings attached to the ground.

$$f = \sum Fext[1:3] - \sum Fb_{Chassis}[1:3]\ (ub) \tag{3.36}$$

$$f = \sum Fext[4:6] + \sum Fext[1:3] \times XFext - \sum Fb_{Chassis}[4:6] - \sum Fb_{Chassis}[1:3]\,(ub) \times Xb \tag{3.37}$$

The next functions (3.38) (3.39), on the other hand, describe the also kinematic constraints of the system, considering the stiffness of the bushings which is not attached to the ground and the stiffness of the springs.

A set of six functions is written for each bushing that connects two different elements, so a bushing not constrained to the frame. The functions calculate the difference between the reaction force of the *internal* bushing and the resultant of the forces and moments calculated around the bushing, of one of the two elements chosen arbitrarily.

$$f = \sum Fb[1:3](ub) - Fb_{Int}[1:3](ub) \tag{3.38}$$

$$f = \sum Fb[4:6](ub) + \sum Fb_{Int}[1:3](ub) \times (X - X_{Int)}) - Fb[4:6](ub) \tag{3.39}$$

Fb_{Int} and X_{Int} represents the reaction forces and position of the bushing that connects the two elements, Fb represents the forces vector of the other bushings of the chose element. The deformation vector ub and the force vector fb are in the local reference system of the bushing. It is therefore necessary to first calculate the appropriate relative deformations of the bushings in the global reference system and then operate using the rotation matrices as described in (3.25).

Finally, the lasts functions, described in the eq. 3.40, must be written to consider the axial stiffness of the spring rod elements. For each spring rod element, the function is written to calculate the difference between the axial force generated by the spring and the modulus of the reaction force vector of one of the two bushings of the element. For convenience, it is best to choose the bushing whose displacement vector ub also describes the displacement u of the element.

$$f = \|Fb(ub[1{:}3])\| - Fa(ua) \tag{3.40}$$

To consider the steering of the suspension by, for example, moving the frame side point of an element representing the trackrod, it is necessary to introduce a displacement vector $dRack$ that is used to translate the X vector that describes the position of the bushing in the equations.

At this point, having written the equations as a function of the displacements u that describe the position and rotation of the elements in space and the axial deformations ua, it is possible to apply the solving algorithm to find the solution that represents the deformed configuration of the suspension.

3.3.2 Double Wishbone Suspension case study

The equations described earlier were written in a very general way on purpose. For a better understanding of the method, it was therefore decided to build an example suspension and the equations for solving the elastokinematics of this configuration will be described below.

A double wishbone front suspension was chosen. The two wishbones were modelled with *rigid elements*, as was the hub carrier. The trackrod was modelled with a *spring rod* element to also provide axial compliance. The spring was also modelled with a *spring rod* that connects the frame to the lower arm. The suspension scheme can be seen in Figure 34.

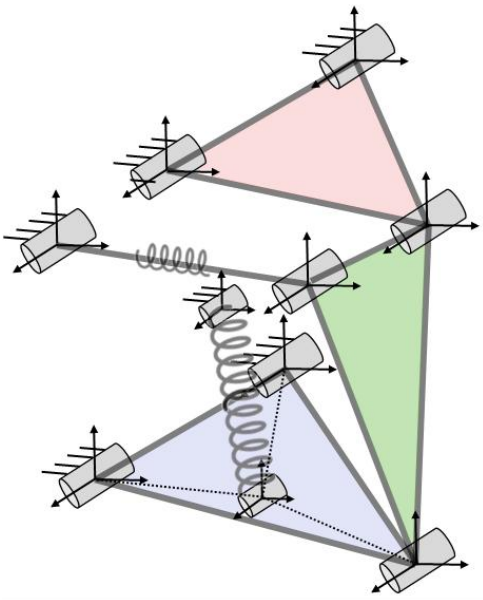


Figure 34. Double Wishbone suspension scheme used in this work. “Spring rod” for trackrod and spring. “Rigid element” for upper arm, lower arm and upright.

The elements are connected to each other and to the frame through deformable bushings. For compactness, we will call the names of the bushings that make up the suspension with acronyms: The front and rear bushings on chassis side of the upper arm are $UAFc$ and $UARc$ respectively. UAw is the bushing of the upper arm on wheel side. For the lower arm the bushing’s names are $LAFc$, $LARc$ and LAw. For the trackrod the names are TRc and TRw chassis side and wheel side respectively and for the spring bushing the names are Sc and Sw. The external forces and moments are applied only in $xFext$ which is a point belonging to the rigid element that describes the upright. For the elements, the upper and lower arm is called UA and LA respectively, TR, S and UP for trackrod, spring and upright. WC is the wheel centre.

This suspension is modelled with three *rigid element*s and two *spring rods* for a total of $3 \cdot 6 + 2 \cdot 7 = 32$ degrees of freedom. U is a 32x1 vector and the number of equations to solve is 32. These equations are reported below creating the function $f(U)$ which will subsequently be set equal to zero to find the solution of the system.

For simplicity of writing, we will establish that in the subsequent equations each ub and Fb vector is referred to in the global reference system (GRS). However, we know that the Fb function is defined in its local reference system as a function of the displacement vector ub, also defined in LRS. So, only in the writing of the equations below, the change of reference system described in (2) is omitted to make them more readable.

The first six functions represent the total equilibrium of the system. The first three functions in eq. 3.41, calculate the difference between the external forces and the reaction forces of the bushings which is attached to the chassis.

$$f[1:3] = Fext[1:3] - \sum_j Fb_j[1:3](ub_j[1:3]) \tag{3.41}$$

The fourth to sixth function (3.42) calculate the difference between the moments generated by the external loads and the moments generated by the bushings on the chassis side.

$$f[4:6] = Fext[4:6] - Fext[1:3] \times XFext - \sum_j Fb_j[4:6]\big(ub_j[4:6]\big) + \sum_j Fb[1:3]\big(ub_j[1:3]\big) \times Xb_j \tag{3.42}$$

Where $j = \{UAFc, UARc, LAFc, LARc, TRc, Sc\}$ both for (3.41) and (3.42).

As mentioned before, f is a function of U which is the vector which contains displacement of the elements u and not function of the bushing's deformation ub. So, it is necessary to redefine ub vectors in function of u. For each element attached to the chassis we set that the associated u vector is equal to one of its bushing's vectors. The chosen one is called *main* bushing (3.43-46). If the element is not constrained to the chassis, this equality cannot be made, and the displacement of the element will be given by the deformation of the bushing added to the displacement and rotation in space of the bushing itself (3.47) (3.48).

$$u_{UA} = ub_{UAFc} \tag{3.43}$$

$$u_{LA} = ub_{LAFc} \tag{3.44}$$

$$u_{TR} = ub_{TRc} \tag{3.45}$$

$$u_S = ub_{Sc} \tag{3.46}$$

$$u_{UP}[1:3] = ub_{UAw}[1:3] + Xb_{UAw}^{UA} - Xb_{UAw} \tag{3.47}$$

$$u_{UP}[4:6] = ub_{UAw}[4:6] + u_{UA}[4:6] \tag{3.48}$$

The upright is not attached to the chassis and Xb_{UAw}^{UA} represents the new point of the Xb_{UAw} bushing associated with the movement of the upper triangle. The understanding of this term will become clearer in later chapters.

The functions (3.41) (3.42) refer to the vectors in the global reference system. It is now necessary to express the ub vectors of the other bushings of the element in terms of the u vector of the element. For each element,

a reference system attached to the element is then created, oriented like the GRS, and centred on the main bushing previously chosen in (3.43-48) to describe the rotation and translation of the element itself. This reference system is called Element Reference System ERS. M is therefore the transformation matrix, described in eq. 3.49, associated with ERS and attached to the element, and GRS.

$$[M(Xb,u)] = \begin{bmatrix} & [R(u[4:6])] & & \begin{matrix} Xb[1]+u[1] \\ Xb[2]+u[2] \\ Xb[3]+u[3] \end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.49}$$

Where Xb is the position vector of the *main* bushing and $R(u[4:6])$ is the rotation matrix and describe the rotation of the element (3.50). In the design configuration it is an identity matrix. In general, the rotation matrix is defined in a similar way to what is described in (3.26) with a *Tait-Bryan* transformation $X_1Z_2Y_3$.

$$[R(u[4:6])] = rotx(u[4]) \cdot rotz(u[6]) \cdot roty(u[5]) \tag{3.50}$$

First, it is necessary to calculate the relative coordinates of the positions of the other bushings of the element in the ERS reference system. The transformation matrix $M0(Xb,0)$ associated with ERS is then created, centred in Xb but with a null u vector, and it is described in eq. 3.51

$$[M0(Xb,0)] = \begin{bmatrix} 1 & 0 & 0 & Xb[1] \\ 0 & 1 & 0 & Xb[2] \\ 0 & 0 & 1 & Xb[3] \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.51}$$

It is now possible to calculate the relative coordinate Xb^{REL} of each other bushing of the element (3.52).

$$\begin{bmatrix} Xb[1]^{REL,Element} \\ Xb[2]^{REL,Element} \\ Xb[3]^{REL,Element} \\ 1 \end{bmatrix} = [M0]^{-1} \cdot \begin{bmatrix} Xb[1] \\ Xb[2] \\ Xb[3] \\ 1 \end{bmatrix} \tag{3.52}$$

For the rigid element the ub bushing chassis side deformations are then expressed as a function of u of the element.

$$ub[1:3] = Xb^{Element}(u) - Xb \tag{3.53}$$

$$ub[4:6] = u[4:6] \tag{3.54}$$

(3.53) and (3.54) are true only for chassis side bushings. the displacement of the part constrained to the element is equal to the deformation of the bushing since the bushing is fixed to the chassis.

The rotational deformation are equals for each bushing in the element. The translational deformations are calculated with the differences with the between the new position of the bushing $Xb^{Element}(u)$ attached on the element side and the position of the bushing Xb on the fixed side where:

$$\begin{bmatrix} Xb^{Element}(u)[1] \\ Xb^{Element}(u)[2] \\ Xb^{Element}(u)[3] \\ 1 \end{bmatrix} = [M(Xb,u)] \cdot \begin{bmatrix} Xb[1]^{REL,Element} \\ Xb[2]^{REL,Element} \\ Xb[3]^{REL,Element} \\ 1 \end{bmatrix} \tag{3.55}$$

It is then now possible to write the equations for linking bushing deformations on the chassis with element deformations. For the upper arm:

$$[M0_{UA}] = \begin{bmatrix} 1 & 0 & 0 & Xb_{UAFc}[1] \\ 0 & 1 & 0 & Xb_{UAFc}[2] \\ 0 & 0 & 1 & Xb_{UAFc}[3] \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.56}$$

$$\begin{bmatrix} [Xb_{UARc}^{REL,UA}] \\ 1 \end{bmatrix} = [M0_{UA}]^{-1} \cdot \begin{bmatrix} [Xb_{UARc}] \\ 1 \end{bmatrix} \tag{3.57}$$

$$[M_{UA}] = \begin{bmatrix} [R(u_{UA})] & [Xb_{UAFc} + u_{UA}] \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \tag{3.58}$$

$$\begin{bmatrix} [Xb_{UARc}^{UA}] \\ 1 \end{bmatrix} = [M_{UA}] \cdot \begin{bmatrix} [Xb_{UARc}^{REL,UA}] \\ 1 \end{bmatrix} \tag{3.59}$$

It is now possible to calculate the bushing deformation on the chassis side.

$$ub_{UARc}[1:3] = Xb_{UARc}^{UA}(u_{UA}) - Xb_{UARc} \tag{3.60}$$

$$ub_{UARc}[4:6] = u_{UA}[4:6] = u_{UAFc}[4:6] \tag{3.61}$$

This procedure is better illustrated in Figure 35.

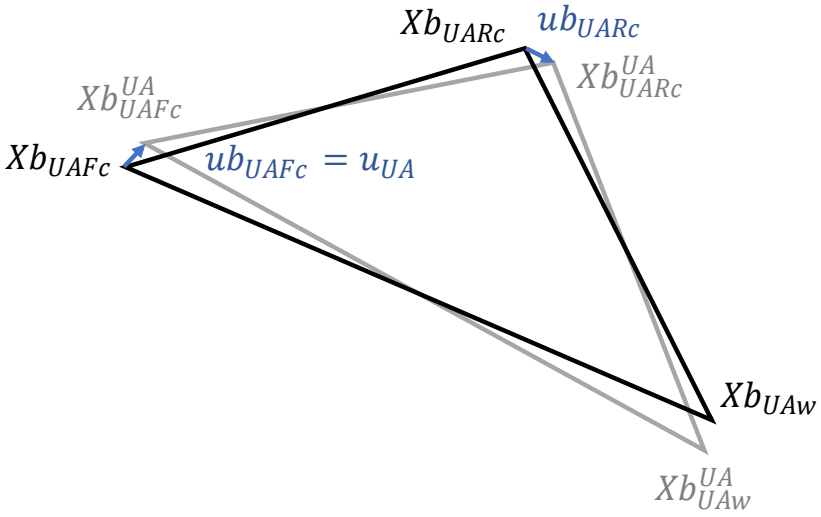


Figure 35. Upper Arm, scheme to identify the chassis side bushing deformations.

The same procedure is carried out for the lower arm.

$$\begin{bmatrix} Xb_{LARc}^{REL,LA} \\ 1 \end{bmatrix} = [M0_{LA}]^{-1} \cdot \begin{bmatrix} Xb_{LARc} \\ 1 \end{bmatrix} \tag{3.62}$$

$$\begin{bmatrix} Xb_{LARc}^{LA} \\ 1 \end{bmatrix} = [M_{LA}] \cdot \begin{bmatrix} Xb_{LARc}^{REL,LA} \\ 1 \end{bmatrix} \tag{3.63}$$

$$ub_{LARc}[1:3] = Xb_{LARc}^{LA}(u_{LA}) - Xb_{LARc} \tag{3.64}$$

$$ub_{LARc}[4:6] = u_{LA}[4:6] = u_{LAFc}[4:6] \tag{3.65}$$

For *spring rod* elements, this step is not necessary as there is always only one bushing constrained to the chassis.

The first set of equations is now consistent, and we can move on to the second set. In these equations we will instead consider the "internal" bushing that connects the elements together as described in (3.38) and (3.39). Regarding the bushing UAw that connects the upper arm to the upright the functions (3.66) are:

$$f[7:12] = [R_{UA}] \cdot Fb_{UAw}([R_{UA}]^T \cdot ub_{UAw}) - TF_{UA} \tag{3.66}$$

Where TF_{UA} is the vector that contains the total forces and moments calculated in the bushings and generated by one of the elements. In this specific case, the arm was chosen as the element.

$$TF_{UA}[1:3] = Fb_{UAFc}(ub_{UAFc}[1:3]) + Fb_{UARc}(ub_{UARc}[1:3]) \tag{3.67}$$

$$\begin{aligned} TF_{UA}[4:6] = {} & Fb_{UAFc}(ub_{UAFc}[4:6]) + Fb_{UARc}(ub_{UARc}[4:6]) \\ & + Fb_{UAFc}(ub_{UAFc}[1:3]) \times \left(Xb_{UAFc} - Xb_{UAw}^{UA}\right) \\ & + Fb_{UARc}(ub_{UARc}[1:3]) \times \left(Xb_{UARc} - Xb_{UAw}^{UA}\right) \end{aligned} \tag{3.68}$$

Xb_{UAw}^{UA} is the new position of the bushing due to the deformation of the upper arm element. It is calculated in the same way of the other bushing in the rigid element (3.69) (3.70).

$$\begin{bmatrix} Xb_{UAw}^{UA} \\ 1 \end{bmatrix} = [M_{UA}] \cdot \begin{bmatrix} Xb_{UAw}^{REL,UA} \\ 1 \end{bmatrix} \tag{3.69}$$

$$\begin{bmatrix} Xb_{UAw}^{REL,UA} \\ 1 \end{bmatrix} = [M0_{UA}]^{-1} \cdot \begin{bmatrix} Xb_{UAw} \\ 1 \end{bmatrix} \tag{3.70}$$

R_{UA} is the rotation matrix associated with the upper arm, eq. 3.71. Since the ub vectors are referential to the GRS, it is necessary to return them to the local reference system since this bushing can rotate in space since it is constrained to the element and not to the frame.

$$[R_{UA}] = rotx(u_{UA}[4]) \cdot rotz(u_{UA}[6]) \cdot roty(u_{UA}[5]) \tag{3.71}$$

The bushing deformation ub_{UAw}, is:

$$ub_{UAw}[1:3] = u_{UP}[1:3] - (Xb_{UAw}^{UA} - Xb_{UAw}) \tag{3.72}$$

$$ub_{UAw}[4:6] = u_{UP}[4:6] - u_{UA}[4:6] \tag{3.73}$$

Other elements of the function f regarding the lower triangle can be written in a similar way in eq. 3.74.

$$f[13:18] = [R_{LA}] \cdot Fb_{LAw}([R_{LA}]^T \cdot ub_{LAw}) - TF_{LA} \tag{3.74}$$

$$[R_{LA}] = rotx(u_{LA}[4]) \cdot rotz(u_{LA}[6]) \cdot roty(u_{LA}[5]) \tag{3.75}$$

In this case, the lower triangle is also constrained to the spring through a bushing, so we enter the TF_S vector that accounts for forces and moments generated by the spring.

$$TF_{LA}[1:3] = Fb_{LAFc}(ub_{LAFc}[1:3]) + Fb_{LARc}(ub_{LARc}[1:3]) + TF_S[1:3] \tag{3.76}$$

$$\begin{aligned} TF_{LA}[4:6] = {} & Fb_{LAFc}(ub_{LAFc}[4:6]) + Fb_{LARc}(ub_{LARc}[4:6]) + TF_S[4:6] \\ & + Fb_{LAFc}(ub_{LAFc}[1:3]) \times \left(Xb_{LAFc} - Xb_{UAw}^{LA}\right) \\ & + Fb_{LARc}(ub_{LARc}[1:3]) \times \left(Xb_{LARc} - Xb_{UAw}^{LA}\right) \\ & + TF_S[1:3] \times \left(Xb_{Sw}^{LA} - Xb_{UAw}^{LA}\right) \end{aligned} \tag{3.77}$$

$$\begin{bmatrix} [Xb_{LAw}^{REL,LA}] \\ 1 \end{bmatrix} = [M0_{LA}]^{-1} \cdot \begin{bmatrix} [Xb_{LAw}] \\ 1 \end{bmatrix} \tag{3.78}$$

$$\begin{bmatrix} [Xb_{LAw}^{LA}] \\ 1 \end{bmatrix} = [M_{LA}] \cdot \begin{bmatrix} [Xb_{LAw}^{REL,LA}] \\ 1 \end{bmatrix} \tag{3.79}$$

$$\begin{bmatrix} [Xb_{Sw}^{REL,LA}] \\ 1 \end{bmatrix} = [M0_{LA}]^{-1} \cdot \begin{bmatrix} [Xb_{Sw}] \\ 1 \end{bmatrix} \tag{3.80}$$

$$\begin{bmatrix} [Xb_{Sw}^{LA}] \\ 1 \end{bmatrix} = [M_{LA}] \cdot \begin{bmatrix} [Xb_{Sw}^{REL,LA}] \\ 1 \end{bmatrix} \tag{3.81}$$

Everything in (3.76) and (3.77) is very similar to what we saw in the previous set of equations (3.67) and (3.68). The only significative difference is the presence of the term ub_{LAw} representing the deformation of the bushing connecting hub carrier and lower arm.

Again, as always, it is necessary to relate the bushing deformation back to the displacement vector of the element. In this case, to calculate the bushing deformation vector, it will be necessary to know the new position of the bushing constrained on the arm and constrained on the upright. the difference of these two vectors is the translational bushing deformation, eq. 3.82.

$$ub_{LAw}[1:3] = Xb_{LAw}^{UP}(u_{UP}) - Xb_{LAw}^{LA}(u_{LA}) \tag{3.82}$$

Regarding rotations, the bushing deformation is given by the difference between the rotation of the upright and the rotation of the lower arm (3.83) because the bushing connects these two elements

$$ub_{LAw}[4:6] = u_{UP}[4:6] - u_{LA}[4:6] \tag{3.83}$$

Since the upright is a rigid element, the procedure is analogous to that seen for the two wishbones.

$$\begin{bmatrix} [Xb_{LAw}^{REL,UP}] \\ 1 \end{bmatrix} = [M0_{UP}]^{-1} \cdot \begin{bmatrix} [Xb_{LAw}] \\ 1 \end{bmatrix} \tag{3.84}$$

$$\begin{bmatrix} [Xb_{LAw}^{UP}] \\ 1 \end{bmatrix} = [M_{UP}] \cdot \begin{bmatrix} [Xb_{LAw}^{REL,UP}] \\ 1 \end{bmatrix} \tag{3.85}$$

This procedure to identify the bushing deformation is also explained in the Figure 36.

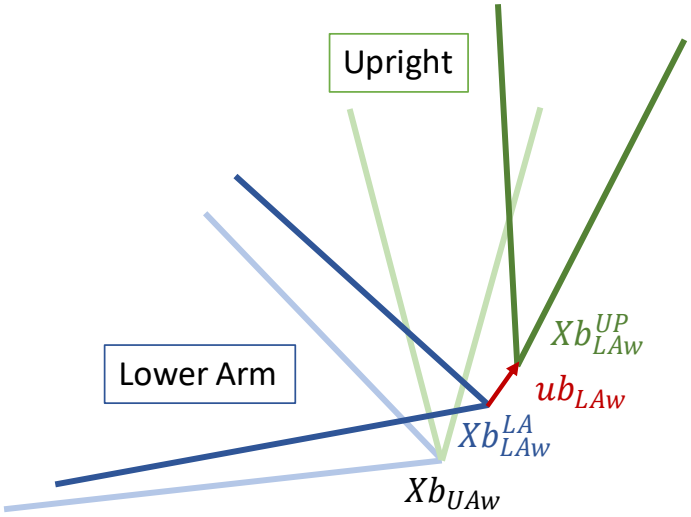


Figure 36. Lower Arm and Upright, scheme to identify wheel side bushing deformations

For spring rod elements, the axial deformation ua of the element must also be considered. Once $\widehat{Xb}_2^{Element}$ has been calculated through the transformation matrix and the relative coordinates as with what we have already seen for the rigid element. An additional step is needed to calculate the new point $Xb_2^{Element}$ while also considering axial deformation.

$$Xb_2^{Element} = \widehat{Xb}_2^{Element} + ua \circ cosdir \tag{3.86}$$

Where $\circ$ is the element-by-element product and $cosdir$ is the vector with the cosine directors (64) associated with the axial direction of the *spring rod* element.

$$cosdir_{element} = \frac{(Xb_1^{Element} - \widehat{Xb}_2^{Element})}{\left\| Xb_1^{Element} - \widehat{Xb}_2^{Element} \right\|} \tag{3.87}$$

$Xb_1^{Element}$ and $Xb_2^{Element}$ are the new position vector associated with two bushing of the *spring rod* element. This procedure is well described in Figure 37.

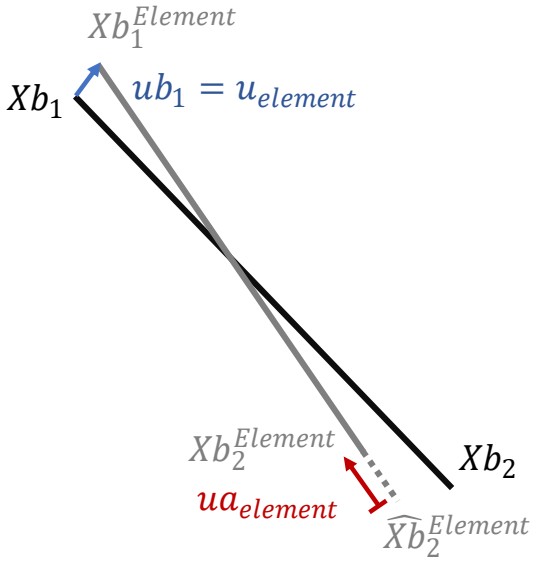


Figure 37. Spring Rod element, scheme to identify bushing deformations with axial displacement

After explaining this procedure, we can then proceed to the calculation of Xb_{Sw}^{S} in eq. 3.92.

$$Xb_{Sc}^{S} = Xb_{Sc} + u_S \tag{3.88}$$

$$\begin{bmatrix} [Xb_{Sw}^{REL,S}] \\ 1 \end{bmatrix} = [M0_S]^{-1} \cdot \begin{bmatrix} [Xb_{Sw}] \\ 1 \end{bmatrix} \tag{3.89}$$

$$\begin{bmatrix} [\widehat{Xb}_{Sw}^{S}] \\ 1 \end{bmatrix} = [M_S] \cdot \begin{bmatrix} [Xb_{Sw}^{REL,S}] \\ 1 \end{bmatrix} \tag{3.90}$$

$$cosdir_S = \frac{(Xb_{Sc}^{S} - \widehat{Xb}_{Sw}^{S})}{\left\| Xb_{Sc}^{S} - \widehat{Xb}_{Sw}^{S} \right\|} \tag{3.91}$$

$$Xb_{Sw}^{S} = \widehat{Xb}_{Sw}^{S} + ua_S \circ cosdir_S \tag{3.92}$$

To complete the equations describing the balance of the lower arm, we calculate the vector TF_S (3.93) (3.94).

$$TF_S[1:3] = Fb_{Sc}(ub_{Sc}[1:3]) \tag{3.93}$$

$$TF_S[4:6] = Fb_{Sc}(ub_{Sc}[4:6]) + Fb_{Sc}(ub_{Sc}[1:3]) \times \left(Xb_{Sc}^{S} - Xb_{Sw}^{S}\right) \tag{3.94}$$

It is now possible to write the new set of functions of f to describe the equilibrium on the bushing connecting the spring with the lower arm, eq. 3.95.

$$f[19:24] = [R_{LA}] \cdot Fb_{Sw}([R_{LA}]^T \cdot ub_{Sw}) - TF_S \tag{3.95}$$

Where:

$$ub_{Sw} = Xb_{Sw}^{S}(u_S, ua_S) - Xb_{Sw}^{LA}(u_{LA}) \tag{3.96}$$

The steering tie rod is now considered. The procedure is quite similar to what has already been seen. However, the possibility of moving the chassis-side bushing along the y-axis to simulate the translation of the rack is considered. So, a new vector $dTRc$ is considered, eq. 3.97.

$$dTRc = \begin{bmatrix} 0 \\ dRack \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.97}$$

Where $dRack$ is the rack displacement.

The new set of functions associated to the trackrod attached to the upright is reported below in eq. 3.98

$$f[25:30] = [R_{TR}] \cdot Fb_{TRw}([R_{TR}]^T \cdot ub_{TRw}) - TF_{TR} \tag{3.98}$$

Where R_{TR} is the rotation matrix associated with the trackrod element displacement u_{TR}. And ub_{TRw} is expressed in eq. 3.99 and eq. 3.100.

$$ub_{TRw}[1:3] = Xb_{TRw}^{UP}(u_{UP}) - Xb_{TRw}^{TR}(u_{TR}, ua_{TR}) \tag{3.99}$$

$$ub_{TRw}[4:6] = u_{UP}[4:6] - u_{TR}[4:6] \tag{3.100}$$

$$\begin{bmatrix} [Xb_{TRw}^{REL,UP}] \\ 1 \end{bmatrix} = [M0_{UP}]^{-1} \cdot \begin{bmatrix} [Xb_{TRw}] \\ 1 \end{bmatrix} \tag{3.101}$$

$$\begin{bmatrix} [Xb_{TRw}^{UP}] \\ 1 \end{bmatrix} = [M_{UP}] \cdot \begin{bmatrix} [Xb_{TRw}^{REL,UP}] \\ 1 \end{bmatrix} \tag{3.102}$$

And, like the spring, but also considering the rack displacement:

$$Xb_{TRc}^{TR} = Xb_{TRc} + u_{TR} + dTRc \tag{3.103}$$

$$\begin{bmatrix} [Xb_{TRw}^{REL,TR}] \\ 1 \end{bmatrix} = [M0_{TR}]^{-1} \cdot \begin{bmatrix} [Xb_{TRw}] \\ 1 \end{bmatrix} \tag{3.104}$$

$$\begin{bmatrix} [Xb_{TRw}^{TR}] \\ 1 \end{bmatrix} = [M_{TR}] \cdot \begin{bmatrix} [Xb_{TRw}^{REL,TR}] \\ 1 \end{bmatrix} \tag{3.105}$$

$$cosdir_{TR} = \frac{(Xb_{TRc}^{TR} - \widehat{Xb}_{TRw}^{TR})}{\left\| Xb_{TRc}^{S} - \widehat{Xb}_{TRw}^{S} \right\|} \tag{3.106}$$

$$Xb_{TRw}^{TR} = \widehat{Xb}_{TRw}^{TR} + ua_{TR} \circ cosdir_{TR} \tag{3.107}$$

$M0_{TR}$ is calculated similarly to all other elements. the matrix M_{TR}, on the other hand, is calculated by also considering the frame-side bushing translation given by the steering (3.108).

$$[M_{TR}] = \begin{bmatrix} [R(u_{TR})] & [Xb_{TRc} + u_{TR} + dTRc] \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \tag{3.108}$$

To conclude this set of equations, we also define the force vector TF_{TR} (3.109) (3.110).

$$TF_{TR}[1:3] = Fb_{TRc}(ub_{TRc}[1:3]) \tag{3.109}$$

$$TF_{TR}[4:6] = Fb_{TRc}(ub_{TRc}[4:6]) + Fb_{TRc}(ub_{TRc}[1:3]) \times \left(Xb^{S}_{TRc} + dTRc[1:3] - Xb^{S}_{TRw}\right) \tag{3.110}$$

Finally, to complete the function f we need to define the last two functions (3.111) (3.112). These take into account the axial stiffness of the spring and the trackrod.

$$f(31) = \|Fb_{Sc}(ub_{Sc}[1:3])\| - Fa_{S}(ua_{S}) \tag{3.111}$$

$$f(32) = \|Fb_{TRc}(ub_{TRc}[1:3])\| - Fa_{TR}(ua_{TR}) \tag{3.112}$$

Where $ub_{Sc} = u_S$ and $ub_{TRc} = u_{TR}$.

Now that the function f is totally defined, it is possible to calculate the displacement vector U describing the deformed configuration of the suspension by means of the Newton-Raphson algorithm described earlier.

All equations, including the function and solving algorithm were implemented in Matlab®. The suspension pickup points are reported in Table 1. It is specified that this is not a real suspension but only a model created for demonstration purposes for this work.

Table 1. Suspension pickup points coordinates

	$X\ [mm]$	$Y\ [mm]$	$Z\ [mm]$
UAFc	50	400	480
UARc	-180	420	470
UAw	-30	660	450
LAFc	0	300	210
LARc	-300	350	220
LAw	30	700	200
TRc	120	400	460
TRw	100	660	450
Sc	20	400	700
Sw	0	600	180
WC	0	750	320

The reference system in question is a right-handed system, originating in the centre of the front axle. The X-axis faces in the forward direction of the vehicle and the Z-axis faces upward.

For the comparison, the same suspension has been designed using MathWorks Simscape® Multibody with the same pickup points, same bushing, and same forces. A screenshot of the suspension created in Simscape® Multibody can be seen in the Figure 38 and Figure 39.

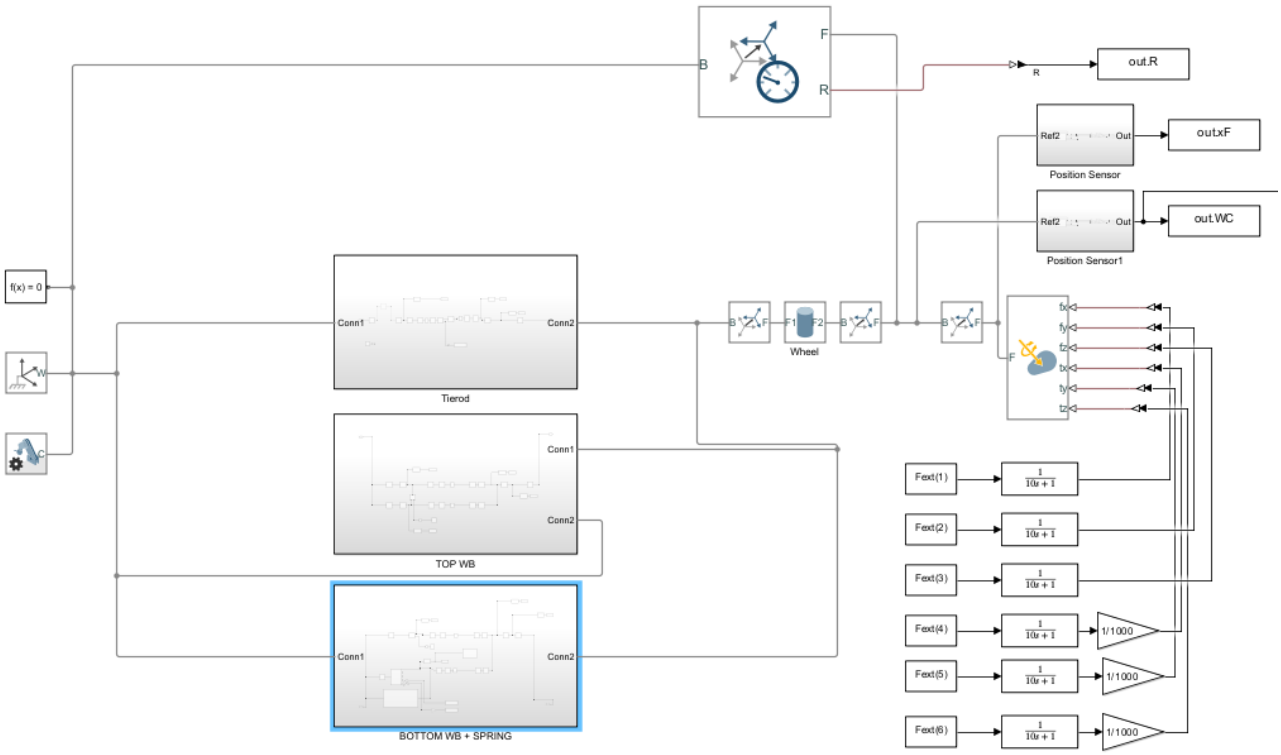


Figure 38. Suspension model designed in Simscape® Multibody.

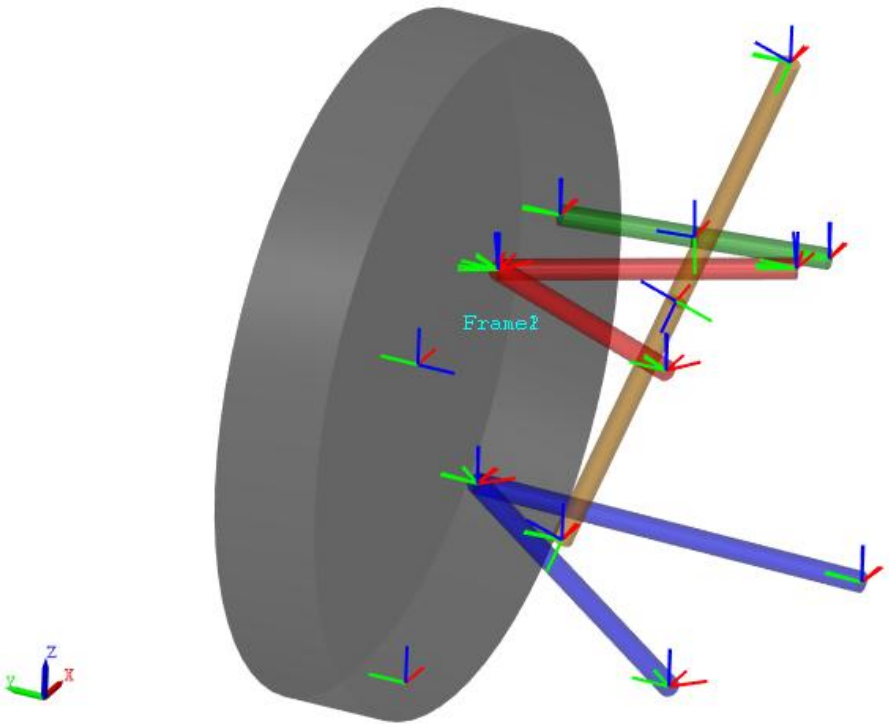


Figure 39. Suspension model designed in Simscape® Multibody's 'Mechanics Explorer'

Being a multibody software, the simulation also takes inertial and damping effects into account. For this reason, only the last values are taken as useful data for comparison, once the transient is exhausted and the whole system is at steady state, equilibrium is given only by the stiffnesses. For simplicity, in the design configuration

all bushings are oriented the same way and in particular the local axes of each bushing are oriented as the axes of the GRS.

Unfortunately, Simscape® does not allow modelling of bushing with nonlinear stiffnesses. it is only possible to assign six stiffness values for each bushing and one axial stiffness. Therefore, linear stiffnesses are used for both bushing and axial compliance for models comparison purposes.

All frame-side bushings have the same stiffnesses, and their values are given in the Table 2. Same for wheel-side bushing, which are all the same and have the same translational stiffnesses as frame-side bushing but one-tenth the rotational stiffness. This is to avoid making the suspension too stiff during steering. It should be noted that again the stiffness values do not represent actual bushing, although an order of magnitude consistent with commercially available bushing is chosen.

Table 2. Bushings translational and rotational stiffness

	$K_x \left[\frac{N}{mm}\right]$	$K_y \left[\frac{N}{mm}\right]$	$K_z \left[\frac{N}{mm}\right]$	$Kr_x \left[\frac{Nm}{deg}\right]$	$Kr_y \left[\frac{Nm}{deg}\right]$	$Kr_z \left[\frac{Nm}{deg}\right]$
UAFc	1000	2000	3000	1500	30000	30000
UARc	1000	2000	3000	1500	30000	30000
UAw	1000	2000	3000	150	3000	3000
LAFc	1000	2000	3000	1500	30000	30000
LARc	1000	2000	3000	1500	30000	30000
LAw	1000	2000	3000	150	3000	3000
TRc	1000	2000	3000	1500	30000	30000
TRw	1000	2000	3000	150	3000	3000
Sc	1000	2000	3000	1500	30000	30000
Sw	1000	2000	3000	150	3000	3000

Table 3. shows the general characteristics of the vehicle on which this idealized front suspension is mounted. The vehicle data are used to calculate the load cases that will be used later for the comparison.

Table 3. Vehicle characteristics used for load cases calculation

Wheelbase [mm]	2750
Front track [mm]	1500
Rear track [mm]	1500
CG height [mm]	500
Traction	RWD
Total mass [kg]	1600
Total weight distribution [%]	60
Non-suspended mass front [kg]	80
Non-suspended mass rear [kg]	80
Roll stiffness distribution [%]	60
Wheel radius [mm]	320
Braking forces distribution [%]	65
Pneumatic trail [mm]	20

Table 4. shows the load cases used to compare the model with the reference. Ten cases were chosen, containing braking, acceleration, lateral and combined loads. The last two load cases, on the other hand, want to recreate the suspension hitting a pothole while braking and the wheel hitting a curb sideways.

Table 4. List of load cases used for the comparison

N	Load case	F_x [N]	F_y [N]	F_z [N]	M_x [Nm]	M_y [Nm]	M_z [Nm]
1	Static Weight	0	0	4709	0	0	0
2	Static Weight * 1.5 g	0	0	7063.5	0	0	0
3	Braking 0.5 g	-2550.6	0	5422	0	0	0
4	Braking 0.8 g	-4080.96	0	5850	0	0	0
5	Acceleration 0.4 g	3139.2	0	4138	0	0	0
6	Cornering 0.7 g	0	-4630	6867	0	0	92.6
7	Corn 0.7 g+ Acc 0.4 g	3139.2	-4630	4345	0	1004.544	92.6
8	Corn 0.7 g+ Brk 0.8 g	-4080.96	-4630	8009	0	0	92.6
9	Pothole braking	-8161.92	0	7563	0	0	0
10	Curb sideways	0	-7500	4709	0	0	0

The load cases are defined as three forces and three moments applied on the hub carrier. Loads were calculated by considering vehicle data and load transfers given by accelerations. It is not the purpose of this work to report the equations by which the loads were obtained. The points of load application vary according to the case and are given in the Table 5. together with the translation of the rack.

Table 5. Forces application points and rack displacement for each load cases

N	Load case	X [mm]	Y [mm]	Z [mm]	$dRack$ [mm]
1	Static Weight	0	750	0	0
2	Static Weight * 1.5 g	0	750	0	0
3	Braking 0.5 g	0	750	0	0
4	Braking 0.8 g	0	750	0	0
5	Acceleration 0.4 g	0	750	320	0
6	Cornering 0.7 g	0	750	0	-20
7	Corn 0.7 g+ Acc 0.4 g	0	750	0	-20
8	Corn 0.7 g+ Brk 0.8 g	0	750	0	-20
9	Pothole braking	0	750	0	0
10	Curb sideways	0	750	200	0

Figure 40 and Figure 41 show the schematic of the suspension in the design configuration and the deformed configuration after applying load case number 4 and 7, as an example. Two views of the suspension are shown in the Figures, and the deformed configuration calculated through the model and that calculated through Simscape® is shown.

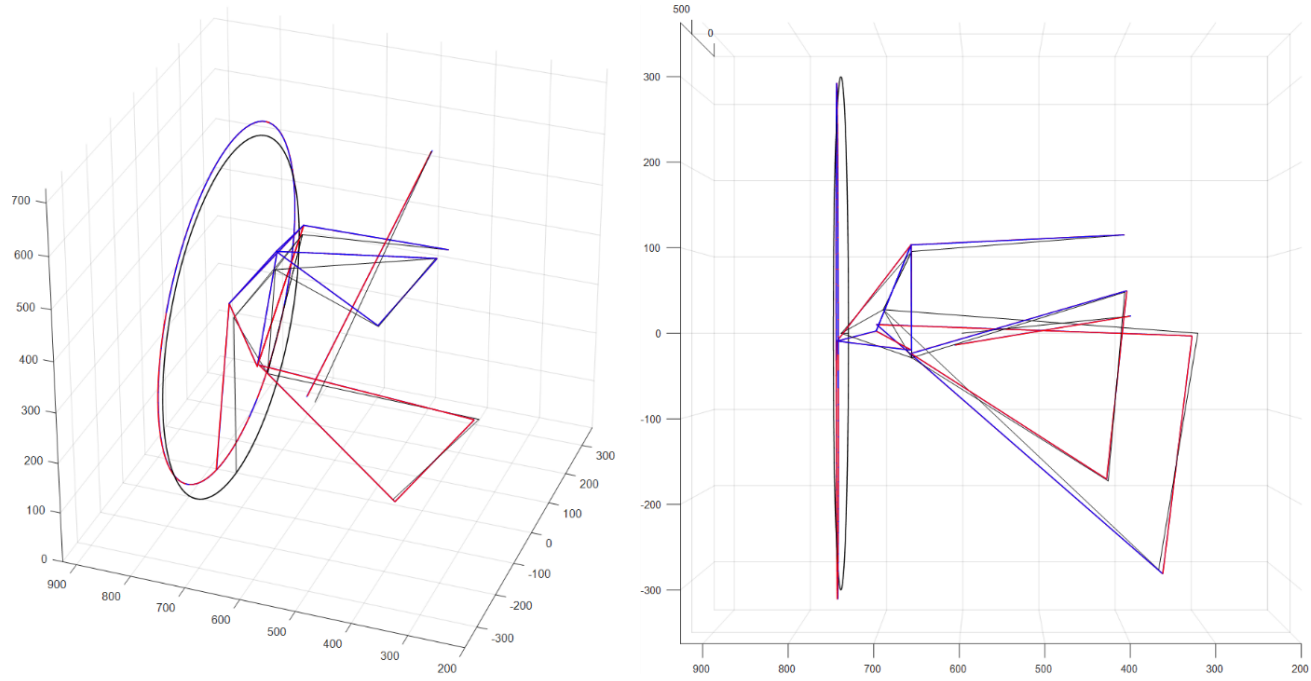


Figure 40. Graphic view of the deformed suspension under load number 4. Design configuration (black), Model deformed configuration (red), Simscape® (blue)

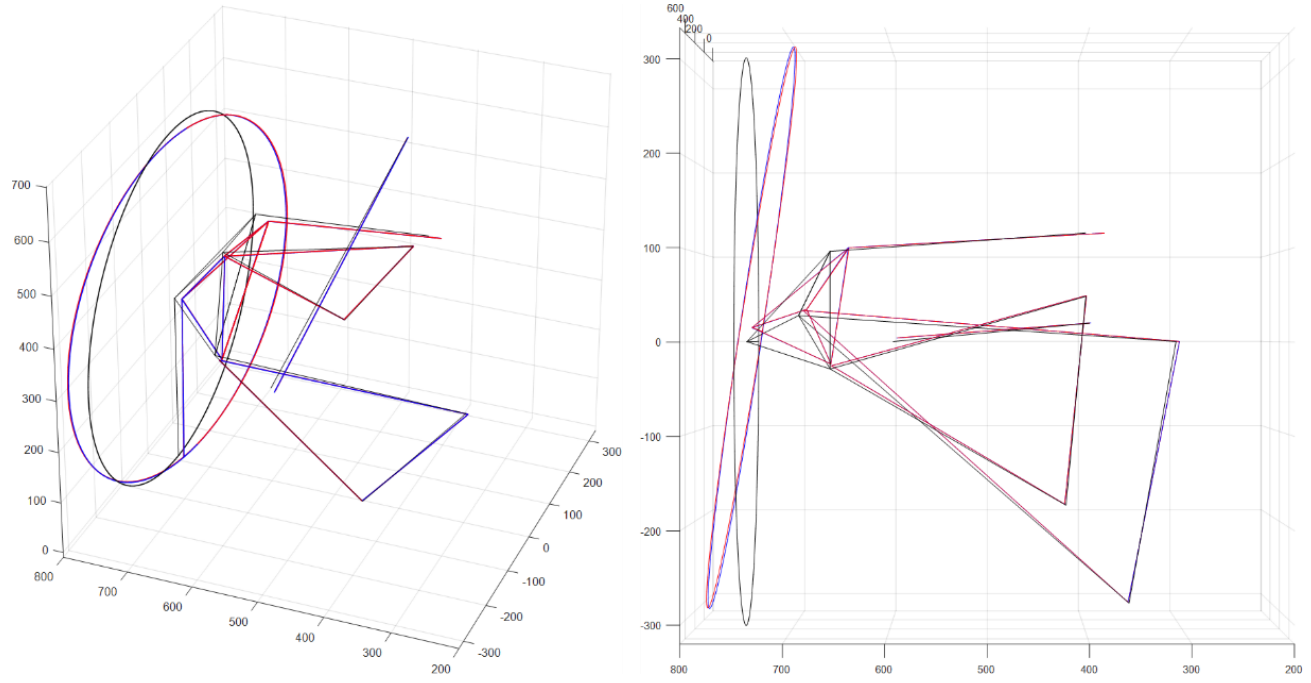


Figure 41. Graphic view of the deformed suspension under load number 7. Design configuration (black), Model deformed configuration (red), Simscape® (blue)

To do the comparison, we chose to compare the values describing the position is the orientation of the wheel in space in the deformed configuration. Specifically,

Table 6. Calculation results of the model for each load case

N	$dCamber$ [°]	$dSWA$ [°]	$dToe$ [°]	dX [mm]	dY [mm]	dZ [mm]
1	-0.75	-1.33	0.28	-1.28	1.57	4.69
2	-1.21	-1.16	1.82	-0.52	-0.95	69.41
3	-1.49	-5.72	0.15	-6.42	3.54	22.07
4	-1.98	-8.33	-0.05	-9.73	4.23	33.04
5	-0.63	-1.26	1.26	4.02	0.40	-9.18
6	-0.91	-1.04	9.00	11.76	-6.71	70.52
7	-0.17	-0.77	8.16	16.10	-5.11	-9.38
8	-2.14	-8.50	6.33	-1.16	-11.86	109.51
9	-3.80	-15.32	-1.39	-20.33	1.84	80.44
10	0.99	0.00	-1.68	-0.57	-5.27	-17.45

Table 7. Calculation results of Simscape® model for each load case

N	$dCamber$ [°]	$dSWA$ [°]	$dToe$ [°]	dX [mm]	dY [mm]	dZ [mm]
1	-0.75	-1.32	0.28	-1.27	1.59	4.97
2	-1.22	-1.14	1.83	-0.57	-1.00	69.84
3	-1.48	-5.62	0.11	-6.43	3.50	22.27
4	-1.96	-8.17	-0.12	-9.75	4.16	33.09
5	-0.63	-1.27	1.27	4.04	0.44	-8.90
6	-0.80	-0.95	8.49	11.05	-6.57	70.27
7	-0.11	-0.75	7.84	15.68	-5.10	-9.88
8	-2.02	-8.17	5.90	-1.70	-11.25	107.61
9	-3.74	-14.91	-1.52	-20.27	1.98	78.52
10	1.02	0.03	-1.89	-0.85	-5.36	-18.23

Table 6. And Table 7. show the changes from the design configuration of the wheel centre coordinates and rotations about their axes, namely camber, toe and side view angle (SWA). Table 6. shows the calculation result of the model, Table 7. the of Simscape®.

Table 8. shows the normalized root mean squared error (nRMSE) and The Pearson correlation coefficient (R) values calculated by the model compared with those calculated with Simscape®.

Table 8. nRMSE and R between model and Simscape® results

	$dCamber$ [°]	$dSWA$ [°]	$dToe$ [°]	dX [mm]	dY [mm]	dZ [mm]
nRMSE [%]	3.7%	2.9%	5.9%	3.3%	4.0%	1.7%
R	0.999	1.000	0.999	1.000	1.000	1.000

The results shown in the Table 8. are also shown in the graphs in the Figure 42. for better visualization.

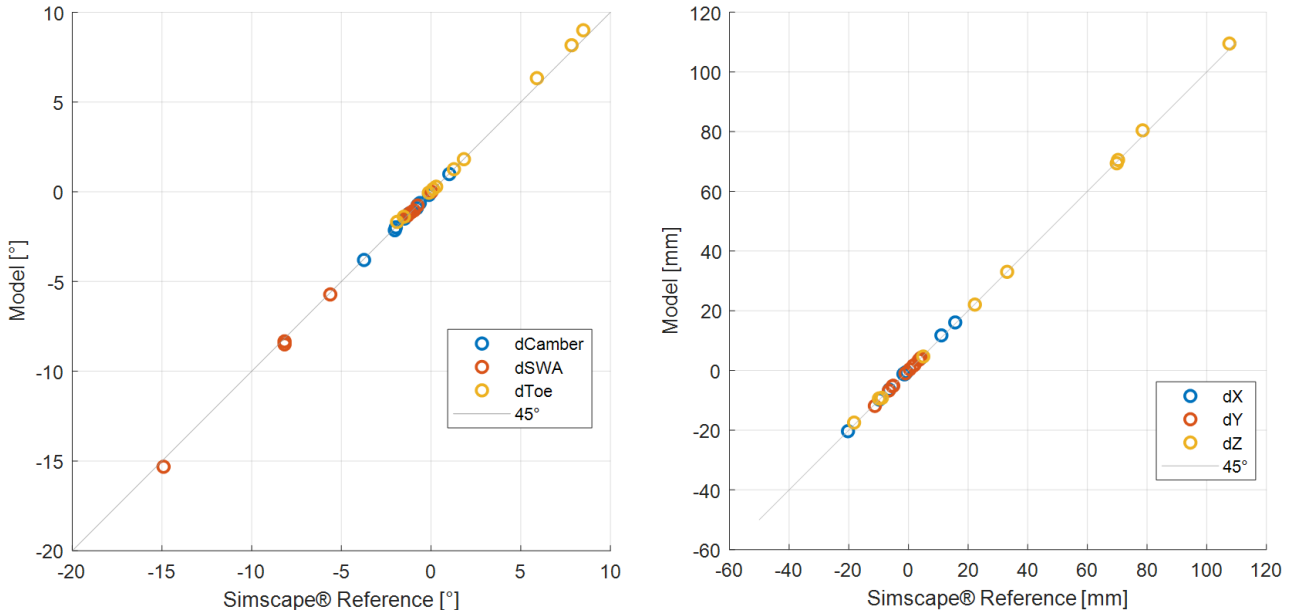


Figure 42. Model Results VS Simscape® results. wheel orientation (dCamber, dSWA, dToe) on the left side and position (dX,dY,dZ) on the right side

Looking at the comparison with the multibody software, the model seems to compute the solution correctly with low errors. the correlation coefficients appear to be essentially unitary. Qualitatively, by graphing the deformed configuration for different loads these are always practically overlapping, as can be seen in the Figure 40 and Figure 41.

This section reports the results of elastokinematics computations using nonlinear bushings, spring and trackrod. The same suspension scheme seen in the previous chapter is used, with the same pickup points. However, nonlinear stiffnesses are used to better simulate the true behaviour of bushing. Specifically, each of the six force-displacement functions that characterize bushing are described by the function $Fb_{NL}(ub)$. Each of these functions is described by a fifth-degree polynomial with a first-degree term inside to describe the linear behaviour for small displacements. The shape of the function is graphed in Figure 43. Displacements and rotations are defined in millimetres and radians.

$$Fb_{NL}(ub) = (ub \circ cx)^{\circ 5} + 1000 \cdot cx \circ ub) \circ cy \tag{3.113}$$

Where, for the bushings, the coefficients cx and cy are two 6x1 vectors associated with each of the six bushing stiffnesses. All bushings are oriented as described in the previous chapter, and the rotational stiffnesses of the wheel-side bushings are one-tenth of those on the chassis-side as can be seen from in (3.114-116).

$$cx_c = cx_W = [1 \quad 1 \quad 1 \quad 180/\pi \quad 180/\pi \quad 180/\pi] \tag{3.114}$$

$$cy_c = [1N \quad 2N \quad 3N \quad 1.5Nmm \quad 30Nmm \quad 30Nmm] \tag{3.115}$$

$$cy_w = [1N \quad 2N \quad 3N \quad 1.5Nmm \quad 30Nmm \quad 30Nmm] \tag{3.116}$$

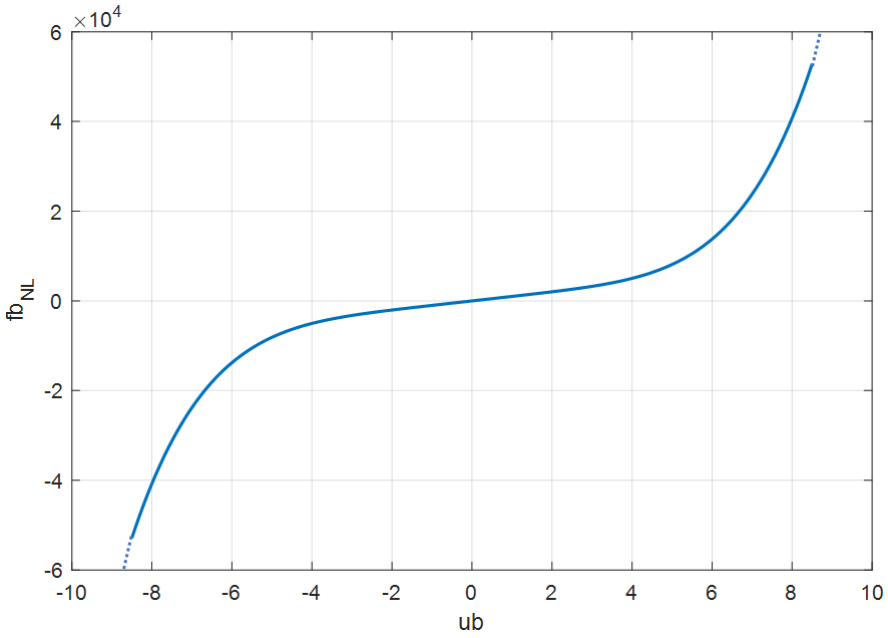


Figure 43. Shape of the function (90), on X-axis the bushing deformation, on Y-axis per bushing reaction

The axial stiffness of the trackrod (3.117) is also defined by a similar function of (3.113) where $cx_{TR} = 1$ and $cy_{TR} = 5$.

$$Fa_{TR}(ua_{TR}) = (ua_{TR})^5 + 1000 \cdot ua_{TR}) \cdot 5\ [N] \tag{3.117}$$

Instead, the spring is described in (3.118) with a linear function and preload. However, the presence of a bumpstop with a clearance. The curve force-displacement of the bumpstop is analogous to (3.113).

$$Fa_S(ua_S) = \begin{cases} k_s \cdot ua_s\ N, & ua_s < clr \\ k_s \cdot ua_s + ((ua_S - clr)^5 + 1000 \cdot (ua_s - clr)) \cdot 0.2\ N, & ua_s \geq clr \end{cases} \tag{3.118}$$

Where $k_s = 60\ N/mm$ is the spring stiffness and $clr = 40\ mm$ is the bumpstop clearance.

These displacement force curves were chosen to make sure that for small displacements the suspension has stiffnesses quite similar to the one analysed in the previous chapter. Next, in fact, a comparison is made between the model with linear and nonlinear stiffnesses to understand how much the nonlinearity can affect the suspension deformation. We chose to compare them using probably what is the heaviest load case, number 8 expressed in the Table 4. The schematic of the deformed suspension is shown in the Figure 44. and forces and moments generated by one of the bushings are shown instead in Figure 45. The rear bushing of the lower arm was chosen as an example.

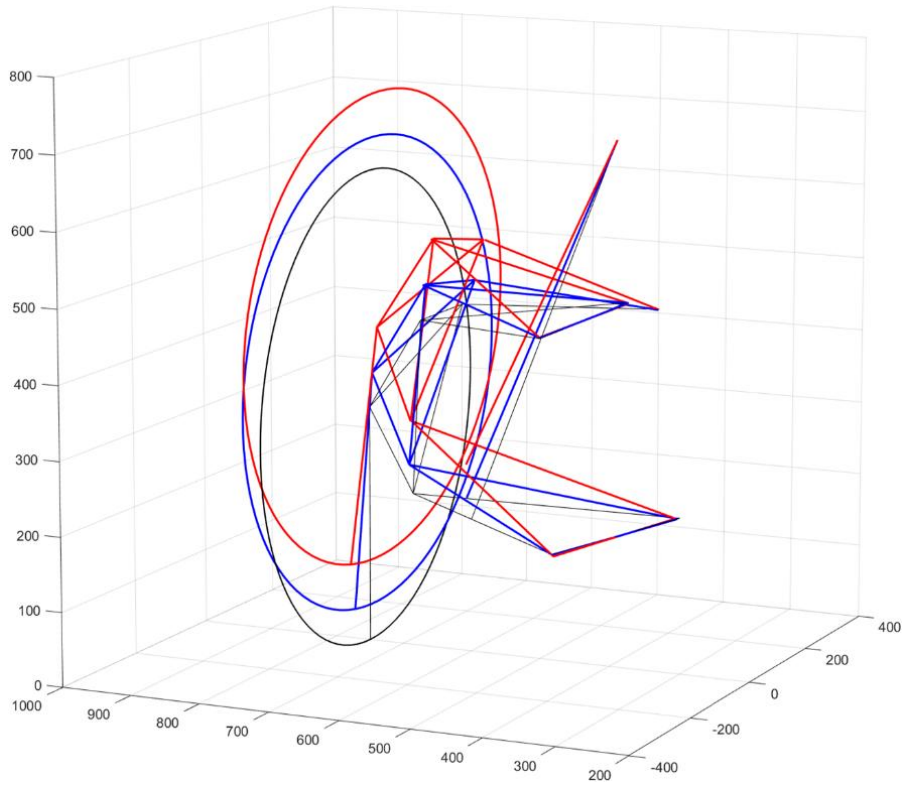


Figure 44. Graphic view of the deformed suspension both for linear and non-linear model under the load N 8. Design configuration (black), Linear (red) and Non-linear(blue)

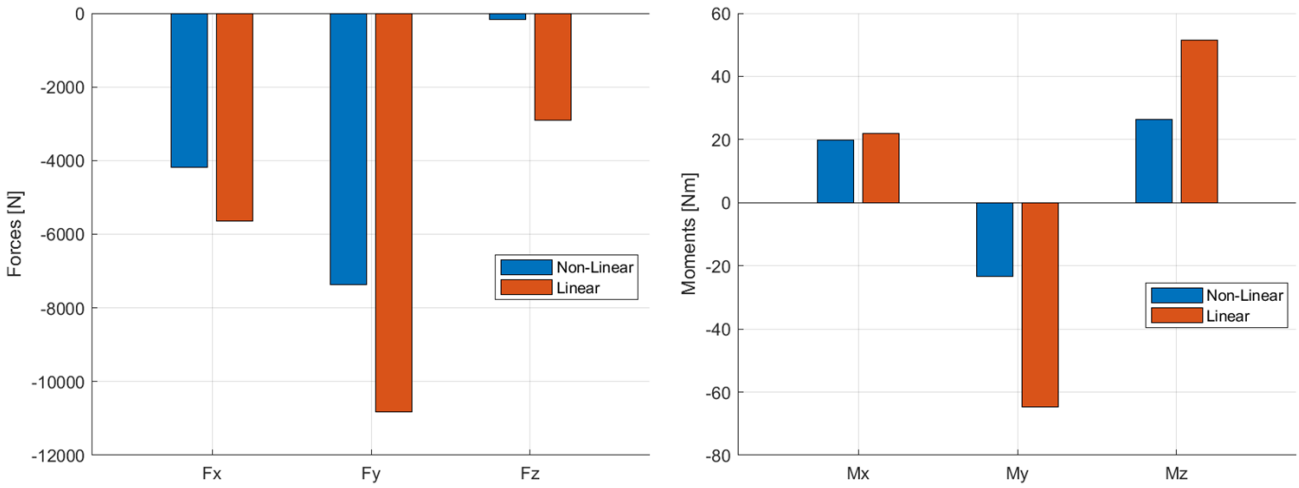


Figure 45. Forces (left) and Moments (right) of the lower arm rear bushing under the load N 8

Finally, the wheel motion was chosen to be observed to see the evolution of elastokinematics at various fractions of the load case. Always using load number 8 defined in Table 4., the calculation is carried out ten times starting first from 10% of the load, then 20% and so on until it reaches 100%. In addition to loads, rack displacement and spring preload are also made to vary progressively. The translations and rotations of the wheel centre are shown in Figure 46. comparing the suspension with linear and nonlinear stiffnesses.

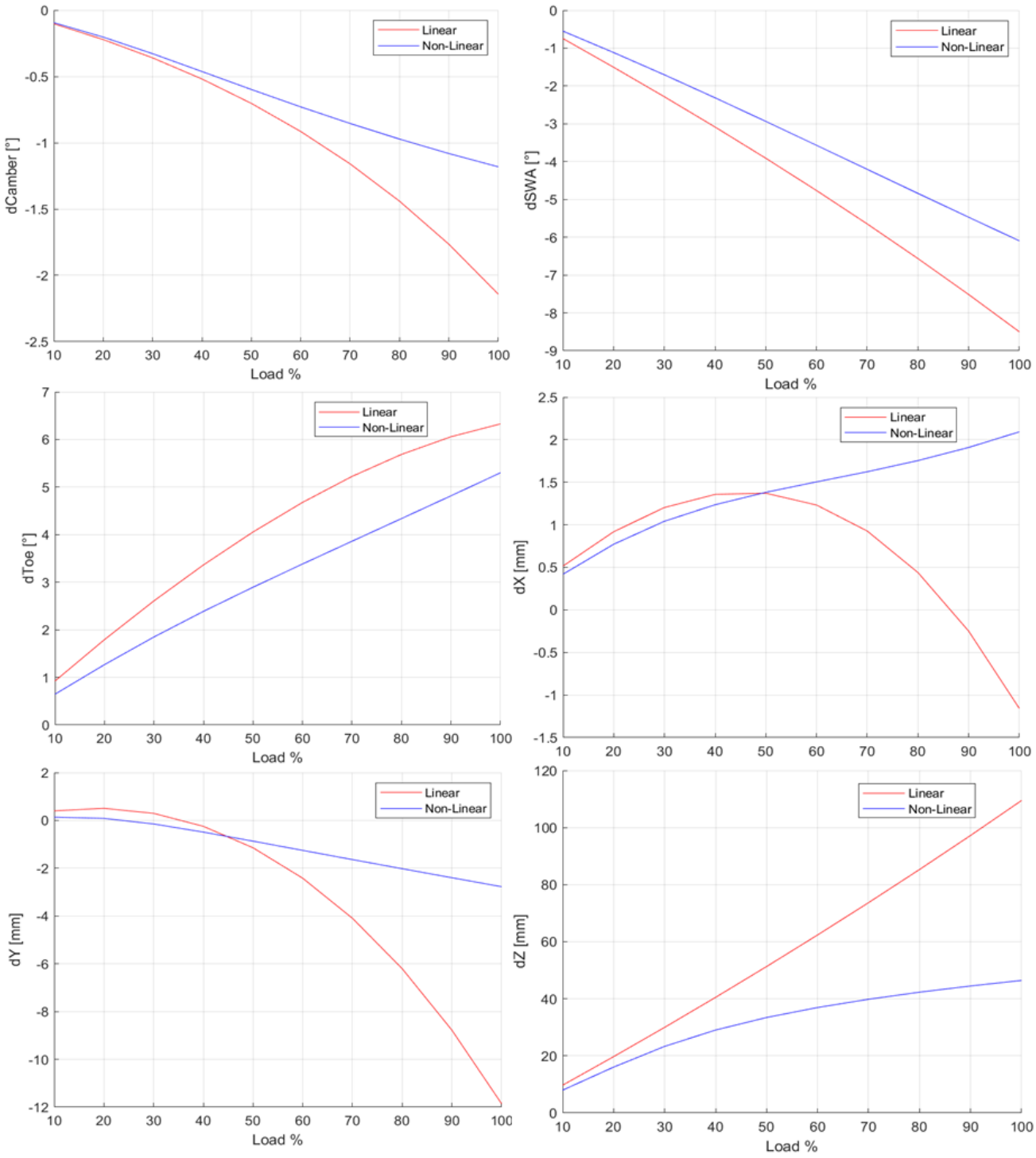


Figure 46. Variation of wheel orientation (dCamber, dSWA, dToe) and position (dX,dY,dZ) at various percentages of the load case number 8

3.4 Axial Stiffness Elasto-Kinematics

This model for calculating suspension compliance, is somewhat complementary to the one explained in the chapter 3.2. In fact, in this model the joints are considered infinitely rigid, the suspension arms on the contrary are considered with compliance. In particular, axial stiffness is considered. This algorithm, unlike the previous two, is dedicated to racecar suspensions.

This model is also based on the load calculation algorithm explained in previous chapters. The wheel is constrained to the frame by 6 connecting rods subjected to axial force. As mentioned in previous chapters, connecting rods can be conFigured in different ways, matching some of their ends to create different suspension architectures.

3.4.1 Model Description

In this model, there are some cases that need to be simplified. Particularly those cases where the sixth connecting rod, that is, the pushrod is tied to one of the arms. In that case, as was explained in the previous chapter, the arm can no longer be considered as two connecting rods.

To overcome this problem, an approximation is made, that is, the wheel-side point of the strut, if constrained to the arm, is made to coincide with the end of the arm, wheel side. This procedure is shown in the Figure 47 for better understanding.

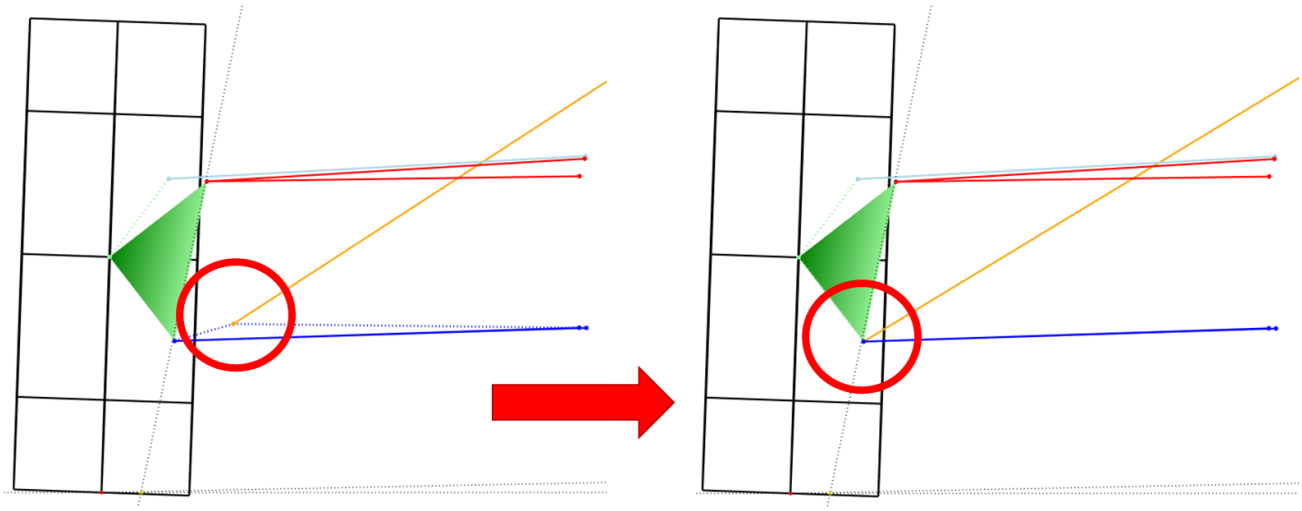


Figure 47. Translation of the pushrod attachment point on the joint connecting the arm to the hub carrier.

In this way, the arm is no longer subject to bending or shear actions and can be subjected only to axial forces. This simplification can be considered valid because generally, the pushrod attachment point is close enough to the arm joint precisely so as not to create bending in it. In this way, all suspension arms can be traced back to connecting rods, and it is then possible to calculate their axial load with the load calculation algorithm by simply solving the balance equations on the hub carrier.

In addition, race cars often have the pushrod tied directly to the hub carrier, because this configuration allows for an extra parameter for vehicle setup by going to create steering-induced vertical wheel displacements, changing the load transfers and thus the balance of the car [52]. In this case, there is no approximation to be made.

Solving this elasto-kinematics requires solving 12 equations. The first 6 equations represent the calculation of forces, then the balance of forces and moments on the hub carrier. The equation to be written are then the same

as those used in the eq. 3.1. The other 6 equations, on the other hand, represent the kinematics of the system, so the same functions used in 3DSusp's algorithm are used that impose a given length of the 6 connecting rods that make up the suspension, the difference being that in this case the length of the connecting rod will not have to be equal to the initial length but will be longer/shorter due to the deformation associated with the axial load.

The following are the acronyms used to name the connecting rods:

- TLF = Top Link Front
- TLR = Top Link Rear
- BLF = Bottom Link Front
- BLF = Bottom Link Rear
- TR = TrackRod
- PR = PushRod/PullRod

$|PR|$ represents the length of the pushrod, while L_{PR} represents the axial force on the pushrod, positive in tension and negative in compression.

The suspension is then defined via the coordinates of the joints, and the six axial stiffnesses for each connecting rod are defined. k_{PR} is, for example, the axial stiffness of the pushrod, expressed in N/mm.

The equation to be solved is in the form $f(X) = 0$ where f is a vectorial function containing 12 scalar functions, described in the eq. 3.119.

$$f(X) = \begin{cases} \sum F_x(X) \\ \sum F_y(X) \\ \sum F_z(X) \\ \sum M_x(X) \\ \sum M_y(X) \\ \sum M_z(X) \\ |TLF(X)| - |TLF_{initial}| - \dfrac{L_{TLF}}{k_{TLF}} \\ |TLR(X)| - |TLR_{initial}| - \dfrac{L_{TLR}}{k_{TLR}} \\ |BLF(X)| - |BLF_{initial}| - \dfrac{L_{BLF}}{k_{BLF}} \\ |BLR(X)| - |BLR_{initial}| - \dfrac{L_{BLR}}{k_{BLR}} \\ |TR(X)| - |TR_{initial}| - \dfrac{L_{TR}}{k_{TR}} \\ |PR(X)| - |PR_{initial}| - \dfrac{L_{PR}}{k_{PR}} \end{cases} \tag{3.119}$$

X is the vector of unknowns, defined in the eq. 3.120, which contains:

- The 3 angles that define the orientation of the wheel in space.
- The 3 coordinates defining the position of the wheel.
- The 6 axial forces each connecting.

$$[X\,] = \begin{bmatrix} Camber \\ Toe \\ SVA \\ X_{Wheel} \\ Y_{Wheel} \\ Z_{Wheel} \\ L_{TLF} \\ L_{TLR} \\ L_{BLF} \\ L_{BLR} \\ L_{TR} \\ L_{PR} \end{bmatrix} \tag{3.120}$$

The Newton-Rapson algorithm (3.121) is used to solve this system of equations.

$$X_{iter+1} = X_{iter} - [J(X_{iter})]^{-1} \cdot f(X_{iter}) \tag{3.121}$$

Where the function is linearised by creating a 12x12 Jacobian matrix J , eq. 3.122.

$$[J(X)] = \begin{pmatrix} \frac{\partial f_1(\mathrm{X})}{\partial X_1} & \cdots & \frac{\partial f_1(\mathrm{X})}{\partial X_{12}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_N(\mathrm{X})}{\partial X_1} & \cdots & \frac{\partial f_{12}(\mathrm{X})}{\partial X_{12}} \end{pmatrix} \tag{3.122}$$

3.4.2 Results and Comparison with Simulink® Simscape Multibody

The algorithm was implemented in a Matlab® script. An example double wishbone suspension was created, with the arms having random axial stiffnesses in the order of 100kN/mm. The following forces were applied at the ground point of the tire: $Fx = -8000N\ Fy = 5000N\ FZ = 10000N$.

The Figure 48 shows the output of the script, in particular a graphical display of the deformed configuration compared to the design configuration and then a histogram representing the axial forces on each connecting rod. Being a racecar suspension, compliance is very low, the deformation is not clearly visible to the naked eye, but even small variations can influence vehicle dynamics.

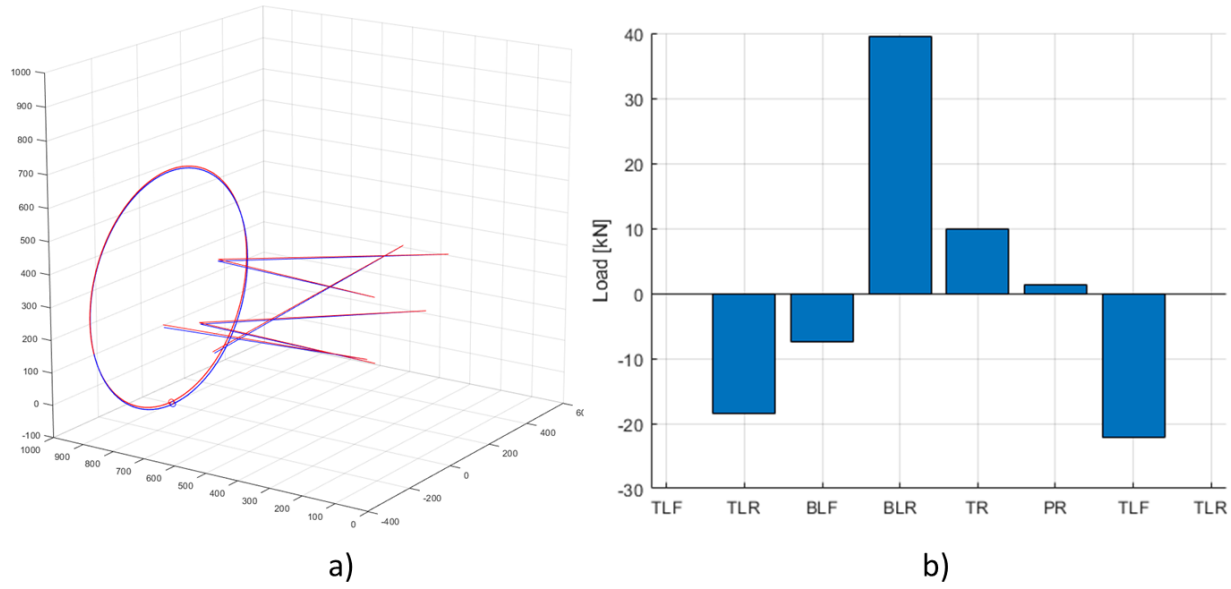


Figure 48. a) Suspension in design configuration (blue) and deformed (red). b) Histogram for axial loads on each connecting rod.

To do some kind of validation, the suspension itself was modelled using Simulink® Simscape Multibody, a software included in the Mathworks® package for carrying out multibody simulations. The suspension itself was then created. A screenshot of Simulink can be seen in the Figure 49.

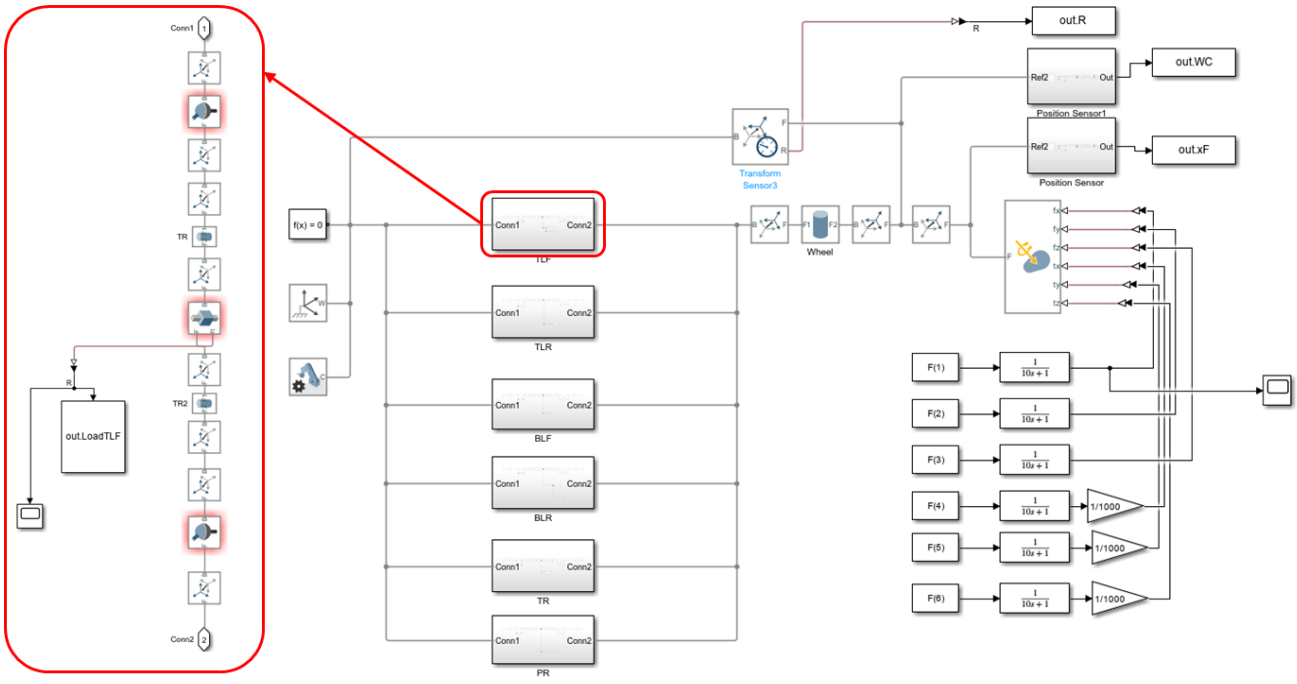


Figure 49. Simulink screenshot showing the Simscape model of the suspension.

The model consists of a rigid body, representing the wheel, to which the forces are applied, and bound to the frame by six connecting rods with axial rigidity. The connecting rods are connected to the frame and the wheel via ideal ball joints. Each connecting rod is modelled by 2 rigid bodies connected to each other via a prismatic

joint with a stiffness along its axis of translation, to model the axial compliance of the connecting rods, as can be seen on the left-hand side of the Figure 49.

In addition, the Figure 50 shows a screenshot of the 'Mechanics Explorer', i.e. Simscape® Multibody's graphical viewer, where the modelled suspension is represented.

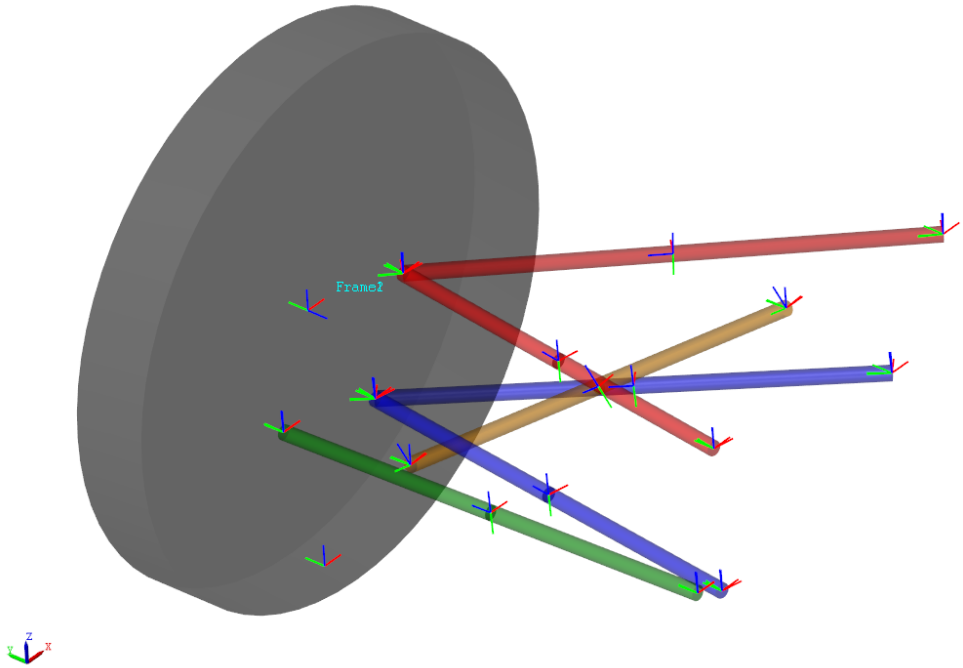


Figure 50. Screenshot of Simscape®'s 'Mechanics Explorer', showing the modelled suspension.

The same load cases mentioned above were then applied to both models. The results are shown in the Table 9 and Table 10.

Table 9. Comparison of results between the proposed model and Simscape® Multibody, wheel position and orientation.

	$Camber[°]$	$Toe[°]$	$SVA[°]$	$X_{Wheel}[mm]$	$Y_{Wheel}[mm]$	$Z_{Wheel}[mm]$
This Model	-0.1051	-0.0395	0.2331	-0.4103	895.1044	266.3452
Simscape® Multibody	-0.1054	-0.0397	-0.2320	-0.4096	895.1044	266.3474
Error %	-0.284	-0.503	0.474	0.170	0	-0.001

Table 10. Comparison of results between the proposed model and Simscape® Multibody, axial loads.

	$L_{TLF}[N]$	$L_{TLR}[N]$	$L_{BLF}[N]$	$L_{BLR}[N]$	$L_{TR}[N]$	$L_{PR}[N]$
This Model	-18505	-7463	39577	9967	1295	-22192
Simscape® Multibody	-18460	-7513	39490	10092	1294	-22262
Error %	0.243	-0.665	0.220	-1.238	0.077	-0.314

The results appear very similar to each other, and the percentage error was small.

3.5 Integration in ViCarRealTime®

As described in chapter 2.3.1, ViCarRealTime offers the possibility of using its own solver in Simulink, with so-called co-simulation. It is possible using the Simulink® block of ViCarRealTime® and setting the input variables correctly. The Figure 51 shows a screenshot with the Simulink block of ViCarRealTime and some of the input variables used for integration.

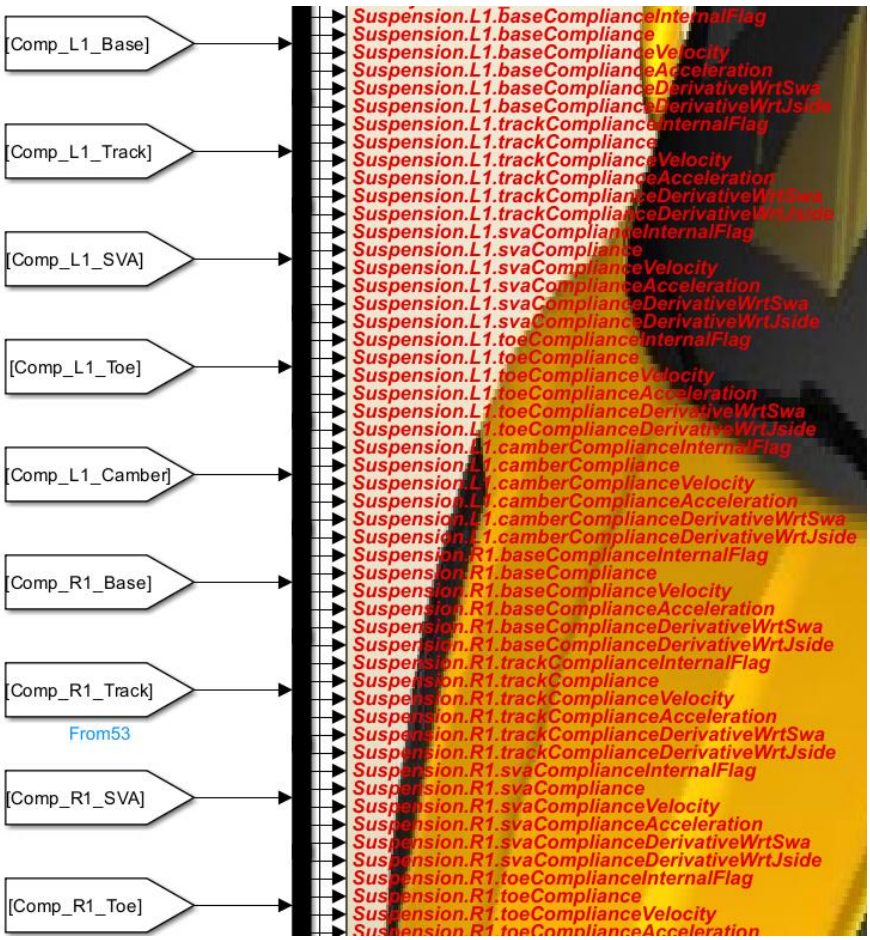


Figure 51. Screenshot with the Simulink block of ViCarRealTime and some of the input variables used for integration.

The following is then a list of input variables to be provided, for each parameter (Variation of wheelbase and track, Toe, Camber and SVA) to the Vi-CarRealTime® solver to externally manage the compliance of a single wheel:

- ComplianceInternalFlag
- ComplianceDisplacement
- ComplianceVelocity
- ComplianceAcceleration
- ComplianceDerivativeWrtSwa
- ComplianceDerivativeWrtJside
- ComplianceDerivativeWrtJopp

Where ComplianceInternalFlag is simply a boolean value to turn internal computation on or off.

ComplianceDisplacement is the value of the parameter at that instant, and Velocity and Acceleration represent the first and second derivatives of that value, respectively, with respect to time. The other values represent first derivative with respect to different values where Jside represents the jounce of the wheel itself, Jopp represents the jounce of the opposite wheel of the same axle, Swa represents the steering wheel angle. Wrt means "With respect to." Since it is still independent wheel suspension, there will be no dependence on the jounce of the opposite wheel, so the value ComplianceDerivativeWrtJopp will be set equal to 0.

As seen in the chapter for the integration of kinematics Integration with Vi-CarRealTime®2.3, it is sufficient to create, from the script of the elasto-kinematics algorithm, a Matlab® function to be inserted into Simulink®. In this way, it is possible to manage compliance using one of the developed algorithms, having more control over the suspension parameters.

Vi-CarRealTime in fact offers the possibility, in one's own vehicle model, in the subsystem 'Suspension' to enter the elasto-kinematics of the suspension, in the tab called 'Compliance'.

In Vi-CarRealTime the wheel orientation is implemented using the effect superposition principle. Given that the kinematic position and orientation being defined by the dependency on suspension jounce and eventually steering wheel angle the effect of external loads on a given suspension property (track or base variation, toe, camber, side view angle) is computed as the sum of the effect given by all active loads taken singularly.

The aim of the work explained below is to compare the model based on the superposition of effects with the proposed model by means of some vehicle dynamics simulations. Trying to point out that, the method based on the superposition of effects has its limitations when dealing with non-linear systems, especially because of rubber bushing.

3.5.1 Comparison with superposition principle model

A generic model of a double wishbone suspension was created, featuring nonlinear, force-displacement bushing characteristics defined with a fifth-degree polynomial to recreate their typically progressive, stiffening behaviour. The model used is the one described in Chapter 3.3.

The suspension model was implemented as a Matlab® function and transferred to Simulink®. This second function was then linked to ViCarRealTime®. A mix of dynamic simulations such as Step Steer, DLC and Braking in Turn, were then carried out as a case study, their purpose being the comparison of the above model with a simpler one based on the superposition principle, where the effect of each force or moment on the suspension is considered as orthogonally decomposed.

This kind of approach is typically used for K&C rig testing results: the suspension deflection is calculated as a simple summation of all effects of each external action. Indeed, one of the goals of this work is to show that the proposed model can deal with strong nonlinearities in the system more accurately. A basic vehicle model

was therefore used. To creating the “superposition” model, using the proposed algorithm, a series of 3-dimension look-up tables were generated in as a function of wheel travel, steer and external force, each describing the effect that each external force has on the suspension's 6 degrees of freedom.

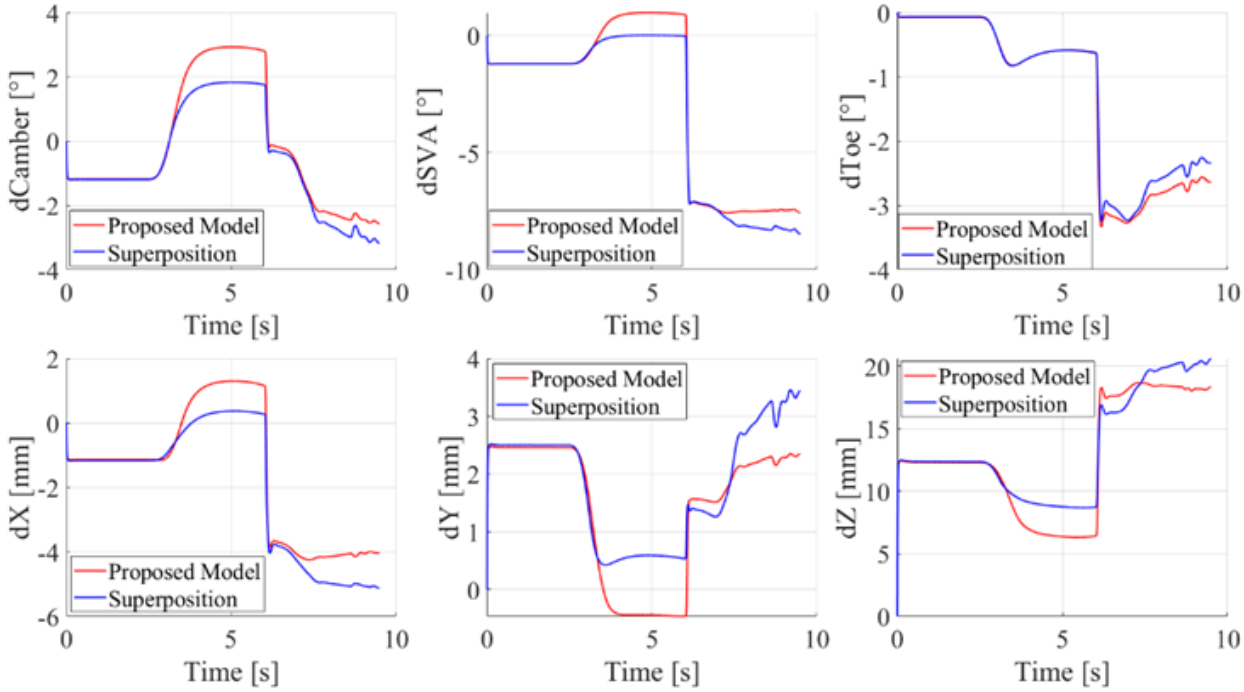


Figure 52. Time history of front wheel assembly motion during a simulated cornering maneuver followed by heavy braking. Variations are shown with respect to the design position. Proposed model (red) vs model based on the superposition principle (blue).

Figure 52 reports wheel displacement data obtained with the two models during the simulation of a complex maneuver with a passenger car: the vehicle initially travels in a straight line. It is then prompted into a constant radius curve. Finally heavy braking is applied, achieving deceleration levels up to 1g. To decouple kinematics and compliance, wheel forces were applied to a suspension model equipped with a very stiff spring. In this first comparison, the two methods are compared "offline" by providing the two models with the forces and moments acting on the wheels as input.

In the second comparison, the two models are directly implemented in the dynamic simulation. This simulation is a corner-braking test where the vehicle starts from a speed of 108 km/h and enters in a cornering with a radius of 75m. A virtual driver model controls the steering. After 2 seconds the driver brakes following a target deceleration of 10 $\frac{m}{s^2}$.The purpose is the comparison of the above model with a simpler one based on the superposition principle, where the effect of each force or moment on the suspension is considered as orthogonally decomposed. The suspension deflection is calculated as a simple summation of all effects of each external action.

The simulation has been carried out in the Vi-CarRealTime® environment with Simulink® co-simulation, with an integration step of 0.001s. Only the compliance of the front axle was considered. To ensure that the simulation worked in real time, the algorithm calculating the elastokinematics has been compiled in C and has been run at 500Hz, assigning the calculation of each wheel in parallel to one core of an Intel® Xeon Gold 6134 CPU.

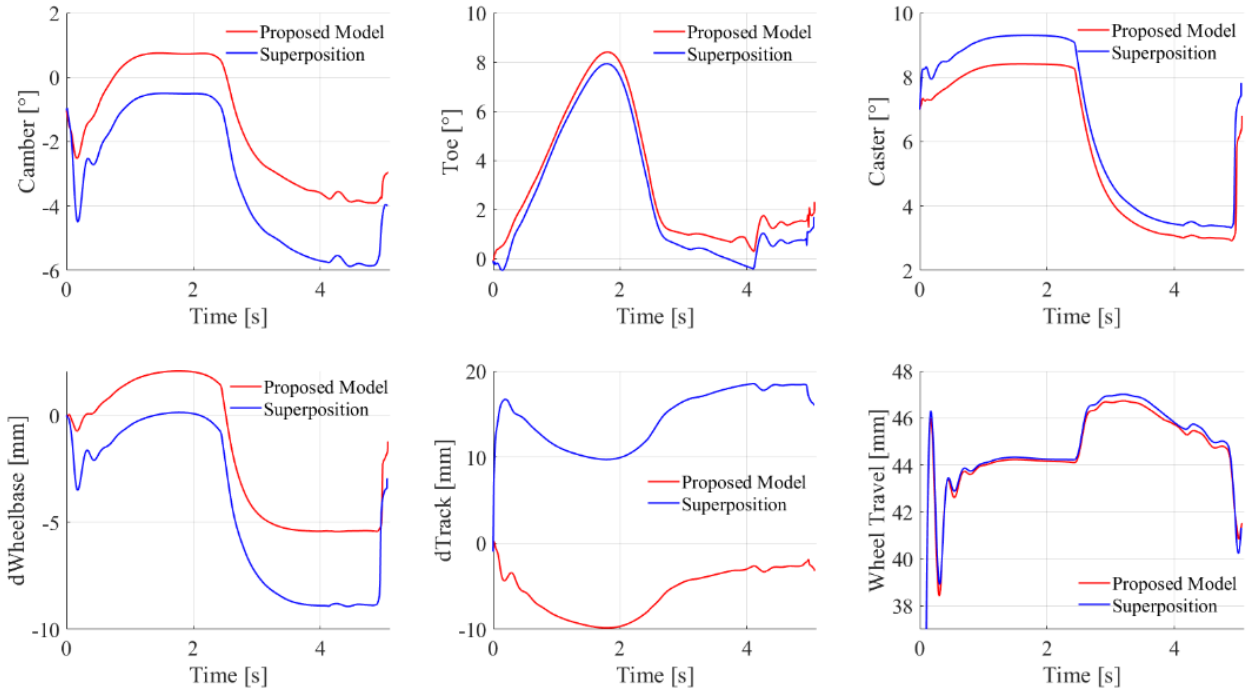


Figure 53. Simulation results for left-hand wheel movements: camber, toe, caster, wheelbase variation and wheel travel. In red the comprehensive proposed model, in blue the model based on lookup-table and superposition.

Figure 53 shows the simulation results for the movements of the left wheel (i.e. the most loaded as it is a right-hand bend): camber, toe, caster, wheelbase variation and wheel travel, comparing the two models.

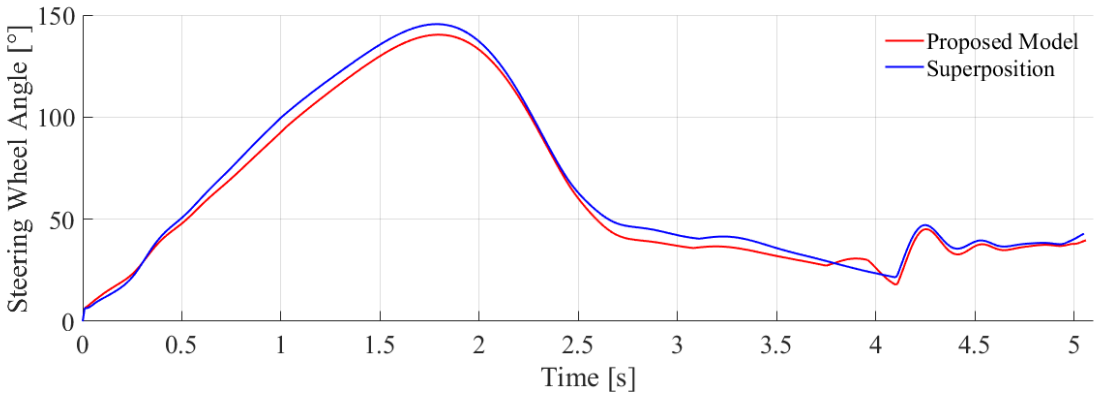


Figure 54. Steering wheel angle over time in the simulation. In red the comprehensive proposed model, in blue the model based on lookup-table and superposition.

Figure 54 represents the steering wheel angle over time. This is different between the two simulations as the test is in closed loop, i.e. virtual driver maintains an imposed trajectory and target deceleration.

By using typical, strongly nonlinear force-displacement curves for the bushings, the difference between the two models becomes particularly apparent, as previously seen in Figure 52 and Figure 53, where large differences in track variation can be observed, even with opposite signs. The differences seen in camber, caster, and toe can significantly influence the handling and performance of the vehicle. This is also reflected in the varying values of steering angle, which alters the feedback for the driver.

Separating the effects of the various actions and combining them through the superposition principle does not lead to an accurate solution when large deformations occur, and large forces act on the wheel. In normal driving, the differences between the two models tend to be negligible. Even when examining the inner wheel during a turn, the differences are less significant.

For performance applications, the need arises for an appropriate solver to accurately calculate compliance effects, enabling an effective design and simulation process without limitations. Elasto-kinematic properties can be computed for any combination of wheel jounce and steering rack position, with the flexibility to change hardpoints or bushing stiffness curves at any time.

Unlike a multi-body model, the proposed model is a quasi-static solver that is well-suited for real-time applications, such as in a driving simulator. In this context, each solution of the problem is close to that of the previous instant, allowing the Newton-Raphson algorithm to converge in negligible time.

However, compared to a multi-body solver, dynamic features of rubber bushings, such as damping and hysteresis effects, are not accounted for. Another limitation of this model is the lack of an anti-roll bar model, which can transfer forces onto the suspension system and affect its elastokinematics.

In this work, the proposed model has been implemented in the simulation with its calculation frequency limited to 500Hz, so that the vehicle model could operate at 1000Hz, which is considered the minimum frequency for real-time. Lowering the update frequency of the elastokinematics does not significantly influence the vehicle model, as the dynamic effects of the suspension are not accounted for here. It is expected that with more powerful hardware, the suspension model frequency can also be increased to 1000Hz.

In the author's opinion, an analytical approach suitable for both design and real-time operation is the best solution in the initial design phase of a vehicle with its suspension and steering systems. Additionally, with the proposed model, it is possible to calculate the constraint reactions on each individual bushing, which is useful for FEM analyses, for example something that cannot be done with a model based on look-up tables or an artificial neural network.

4 Vehicle Side Slip Angle estimation

One of the topics the author worked on during his PhD was precisely the comparison of the two main methods for Vehicle Side Slip Angle (VSA) estimation found in the literature, usually classified as Kalman Filter-based and Neural Network-based.

The Vehicle Side Slip Angle (VSA), often identified with the letter β, is a key quantity for evaluating vehicle and tire behaviour. VSA is defined as the angle between the vehicle's longitudinal axis and its velocity, the vector tangent to the vehicle's trajectory, Figure 55.

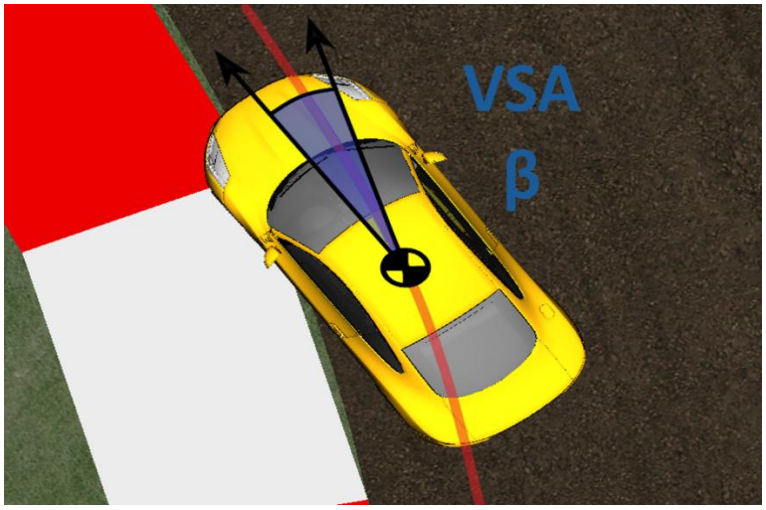


Figure 55. Vehicle Side Slip Angle, the angle in blue between the longitudinal axis of the vehicle and its velocity vector, tangent to the trajectory (red).

However, devices required to obtain a direct measure of the vehicle sideslip angle are not robust to adverse environmental conditions and very expensive as well, therefore they are suitable for R&D rather than for production vehicles. These sensors are usually optical sensors, like the one shown in Figure 56.

Figure 56. Optical sensor for Vehicle Side Slip Angle measurement, Kistler® S-Motion mounted on Fiat 124 by Megaride Applied Vehicle Research®.

The VSA hence requires estimation on the basis of conventional measurements such as lateral/longitudinal acceleration, yaw rate, and steer angle. An estimate of vehicle speed is also required, to be based in turn on wheel angular speeds.

All the above calls for an innovative approach where traditional mechanical engineering skills for chassis design are integrated with digital knowledge, and the use of innovative tools makes for a research framework where subjective factors are also taken into account, thus making it also suitable for the development of Advanced Driver Assistance Systems (ADAS). On top of that, such an approach recalls what is usually defined as model-based design, where digital models are the key to product definition. In this case the models related to VSA estimation strategies can be used for functional simulations at the same time.

4.1 Comparison of The Two Most Used Vehicle Sideslip Angle Estimation Methods

Vehicle sideslip angle estimation is still one of the most challenging research topics in the automotive industry. Many papers can be found on this topic, where authors propose varied methods to reach the goal. After an extensive literature review, two very different methods have been identified as the most used: Extended Kalman Filter with dynamic model and Artificial Neural Network. In this chapter a comparison among these methods is presented. A fully instrumented car has been used to gather typical vehicle dynamics data and feed the models required for a model-based design approach. Results showed that each method has either positive aspects or drawbacks.

4.1.1 Kalman Filter-based estimation

It is well known that the Kalman filter (KF) provides optimal state estimation of linear systems. When dealing with nonlinear systems, as it is often the case in practice, a possible solution to the state estimation problem is offered by the Extended Kalman Filter (EKF). The EKF is based on a recursive linearization of

the system model around the estimated state so that the KF equations can be applied. A complete dissertation about the EKF is beyond the scope of this work and the reader may conveniently refer to [7]. Nevertheless, in order to make the present work more self-contained, the EKF equations required to estimate the state of a given nonlinear system are reported. Generally speaking, KF (and its variants) addresses the general problem of trying to estimate the state vector $x \in \Re^n$ of a discrete-time controlled process that is governed by the generic set of equations [6]. The following system can be considered:

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}) \tag{4.1}$$

$$z_k = h(x_k, v_k) \tag{4.2}$$

where x is the state vector, u the system input, z the system output (the measured variables) and v, w given Gaussian white noise. Q and R are defined as the system noise **w** covariance matrix and the measurement noise **v** covariance matrix, respectively. Now if the system is linear, equations 4.1 and 4.2 can be written in the form:

$$x_k = Ax_{k-1} + Fu_{k-1} + w_{k-1} \tag{4.3}$$

$$z_k = Hx_k + v_k \tag{4.4}$$

Denoting with $\widehat{x_k}^-$ the state estimate at step *k* according to the dynamic evolution of the system (4.3), the KF estimation is:

$$\widehat{x_k} = \widehat{x_k}^- + K(z_k - H\widehat{x_k}^-) \tag{4.5}$$

One form of *K* is given in [6]: it depends upon **Q** and **R**. For example, if **R** approaches zero i.e. for an extremely reliable measurement, then $K \approx H^{-1}$ hence $\widehat{\boldsymbol{x_k}} \approx H^{-1}\boldsymbol{z_k}$. If the system is not linear i.e. it can be described only in the general form (4.1 and 4.2), it can be linearized and written in a form similar to (4.3 and 4.4). Such an approach is known as an EKF. The vehicle model provides a detailed description of its dynamic behaviour, as it is based on the equilibrium equations. Considering the vehicle as a rigid body, the following is a generic set of equilibrium equations:

$$ma_x = m(\dot{u} - vr) = \sum_{i=1}^{p_x} F_{x,i} \tag{4.6}$$

$$ma_y = m(\dot{v} + ur) = \sum_{i=1}^{p_y} F_{y,i} \tag{4.7}$$

$$J_z\dot{r} = \sum_{i=1}^{p_z} M_i \tag{4.8}$$

where m is the vehicle mass, J_z is the polar moment of inertia, $F_{x,i}$ and $F_{y,i}$ are the generic force contributions along the longitudinal and lateral direction respectively, and M_i is the generic yaw moment contribution. A dynamic model can feature different levels of detail and complexity and adopt various assumptions, all these factors affecting the estimation accuracy. For the purpose of this work, a standard four-wheel model has been used (see Figure 57), while the three equilibrium equations can be found in [50].

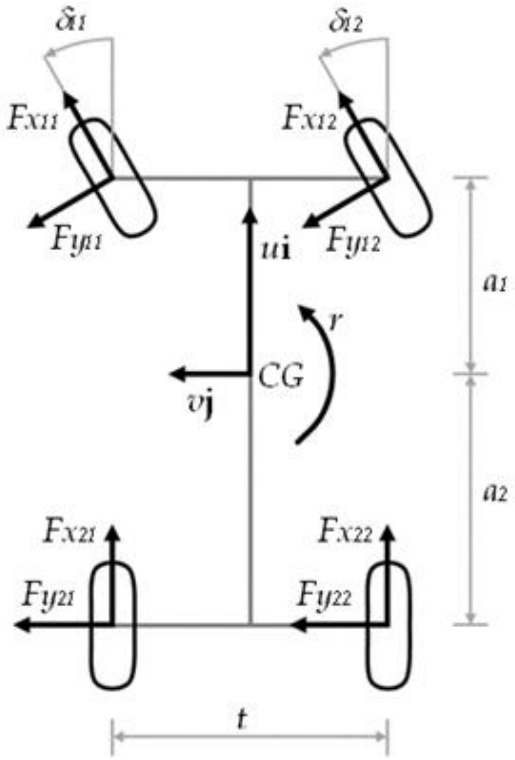


Figure 57. Dynamic model of a vehicle with road-tire interaction force.

4.1.2 Neural Network based estimation

The Artificial Neural Network (ANN) approach is the same adopted in [53], using the Levenberg-Marquardt back-propagation (BP) algorithm along with the mean squared error (MSE) performance index, as seen in [54]. This BP optimization algorithm updates weights and biases of the network as presented in [55]. See Figure 58 for the high-level schematics.

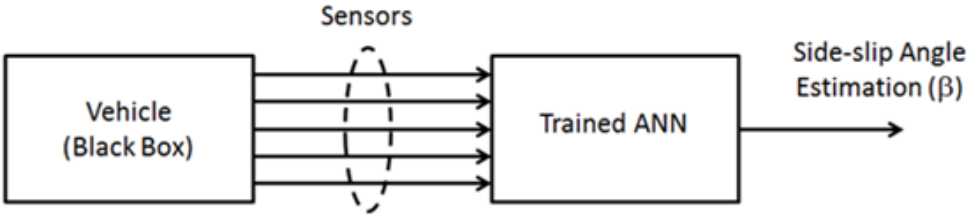


Figure 58. ANN estimation high-level schematics.

The ANN performance is then determined by the MSE which is:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - yt_i)^2 \tag{4.9}$$

where y_i and yk_i are the predicted and target value of the *i-th* pattern respectively and N is the number of patterns. The real-world dataset required to train and test the ANN is composed of input data patterns together with corresponding targets. In this case 90% of the dataset has been used to train the network while the remaining 10% has been used to test the capability of the network. This is a common procedure as reported in [56]. The mathematical background along with testing and training procedures can be found in [55], [56]. For this application inputs (which determine the ANN's input layer neuron number) are vehicle speed (Vx), steering angle (δ), lateral acceleration (ay), longitudinal acceleration (ax) and yaw rate (r). The output is the VSA (β), hence the ANN has only one neuron in its output layer. Real-time estimation capabilities of ANNs strictly depend on the number of neurons in the hidden layer, to be carefully chosen: 10 in this case.

The most challenging part of this work was the identification of an appropriate set of maneuvers to be used for training. In order to be effective, the training data set should be able to represent the system and its dynamics in any possible conditions. When this criterion is met, the neuron number of the hidden layer can be minimized, thus resulting in a simple and light ANN structure [50], [57].

A comprehensive set of demanding maneuvers was designed according to the authors' experience and also with an eye on ISO vehicle test standards, and performed as reported earlier: Step steer from 30° to 120° steering wheel angle with speed ranging from 60 to 150 kph; 2Hz constant frequency sine steering (amplitude 60° and 90°) at 60 and 90 kph; 60m and 90m radius steering pad, with acceleration rate of 5 kph/s. Such data set was employed to train/feed the ANN thus meeting all the above-mentioned criteria. Therefore, the network can be considered ready for testing with a generic data set.

4.1.3 Results and discussion

The car selected for this test was a high-performance road car. All the tests were performed with a help of a professional test driver at the Nardò proving ground (Figure 59) on a flat and dry tarmac surface with a consistent friction coefficient.

Figure 59. Nardò (LE) proving ground.

Every maneuver was repeated at least three times for statistical purposes. Both EKF- and ANN-based methods are fed with the basic but critical data set related to vehicle dynamics. In order to prove the effectiveness of each method, a measurement of the VSA (β) is required as well, to be used as a reference and for the purpose of ANN training. Therefore, a custom set of sensors was designed (see Table 11) and used in conjunction with a motorsport data logger.

Table 11. List of sensors.

Sensor	Type	Range	Accuracy
GPS	MoTeC L10	n.a.	0.1 m/s
Accelerometer	Bosch AM600	±5 G	0,01 G
Yaw rate sensor	YRS3	±210 deg/s	0,1 deg/s
Steering (rotary pot.)	Gefran PZR20	±720 deg	0,1 deg
VSA sensor	Correvit S-400	±60 deg	0,1 deg

This Section reports the comparison between the two VSA estimation methods. Each Figure refers to a different maneuver and includes the time histories of real-world VSA measured with the optical sensor (red) and VSA estimation with ANN (blue) and with EKF (green). Figure 60 and Figure 61 represent two of the so-called steering pad maneuvers in steady state and with different final speed, while Figure 62, Figure 63, Figure 64 and Figure 65 represent four different step-steer maneuvers with different combinations of speed and target steering wheel angle.

First of all, as far as the EKF estimation method is concerned, the overall VSA estimation is considerably accurate. Potential drawbacks are related to the fact that tire/road force interactions should be modelled through

accurate experimental data processed with Pacejka's well-known magic formula [50], but unfortunately data available from tire manufacturers might not be accurate enough quite often. Moreover, the computational effort required by the EKF is fairly high.

Furthermore, it is worth to point out that EKF estimation is quite good in lateral steady state conditions, while the error in transient conditions is larger. This is related to a state-space vehicle model composed of four state variables only, in order to limit the computational burden required: yaw rate, lateral acceleration, lateral forces on front and rear axles [50]. Hence transient maneuvers where combined lateral and longitudinal forces act on the vehicle at the same time are not fully modelled.

Finally, the ANN method appears to be the most accurate on both steady-state and transient maneuvers. This method is also lighter in computational terms since it is based on simple math operations only. However, ANN estimation performances are strictly related to training quality, which in turn might be highly time and cost demanding. Moreover, the ANN is not able to consider heavy environmental and system changes such as tires wear and abrupt transition from dry to wet road surface if they are not explicitly included in the training.

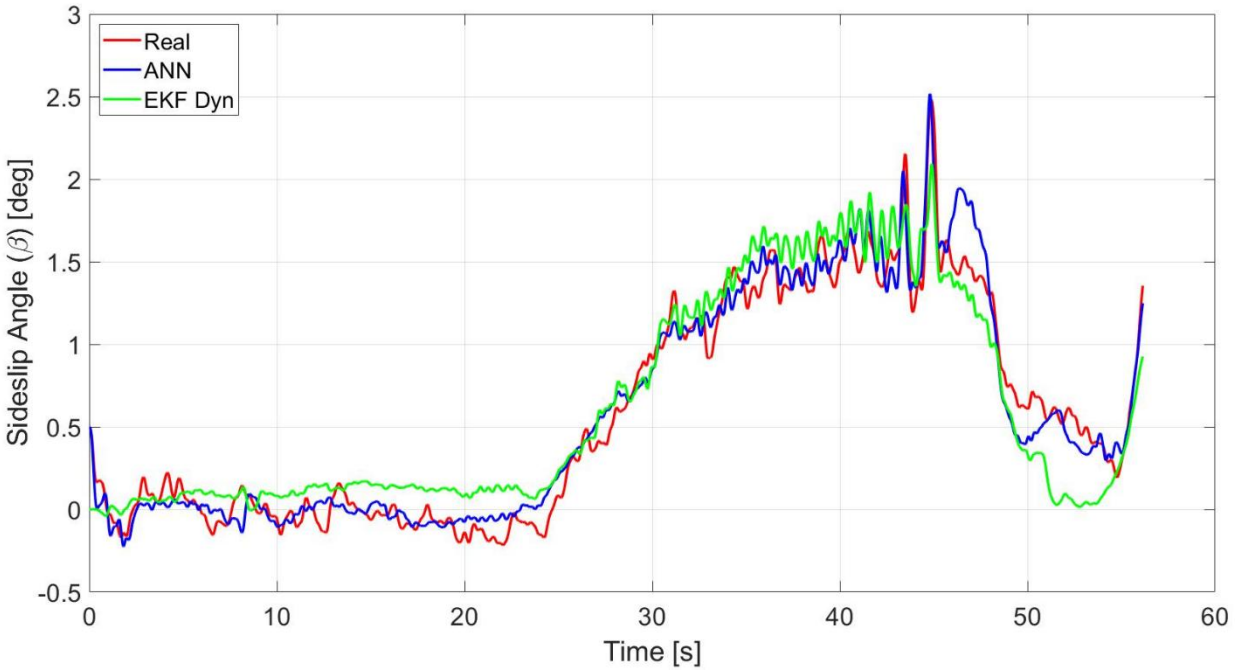


Figure 60. 90m radius steering pad manoeuvre, speed increasing from 60 to 120 kph.

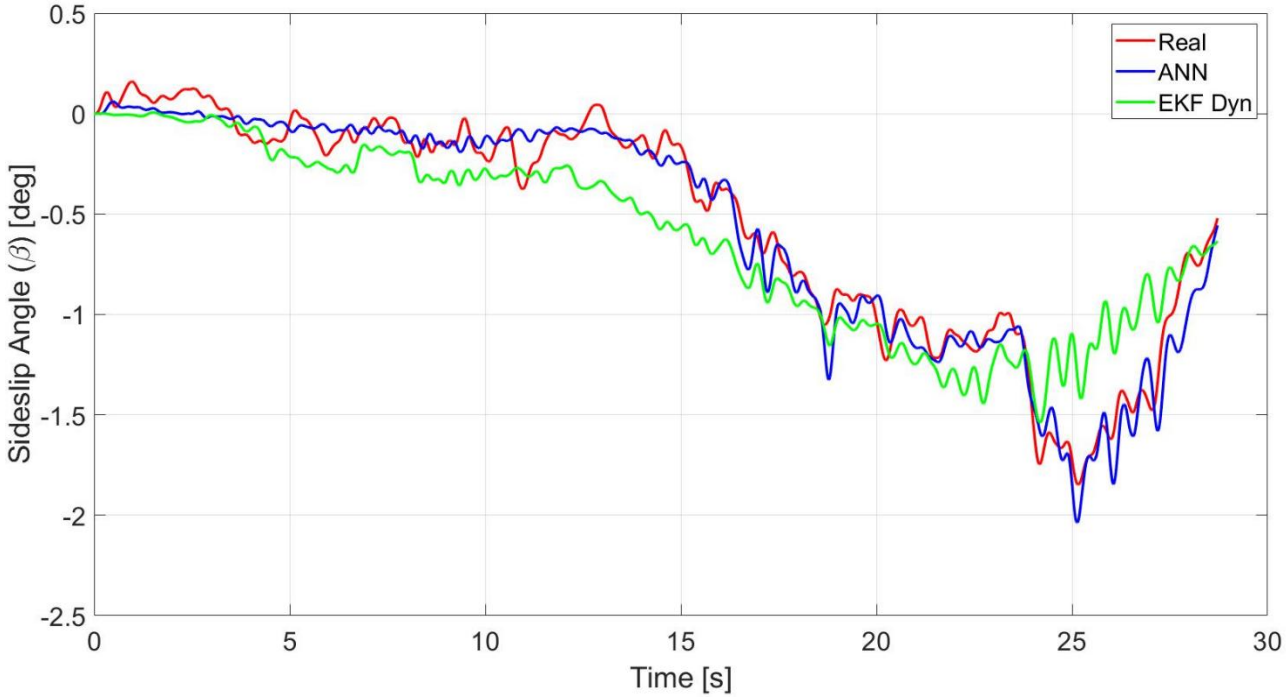


Figure 61. 90m radius steering pad manoeuvre, speed increasing from 60 to 100 kph.

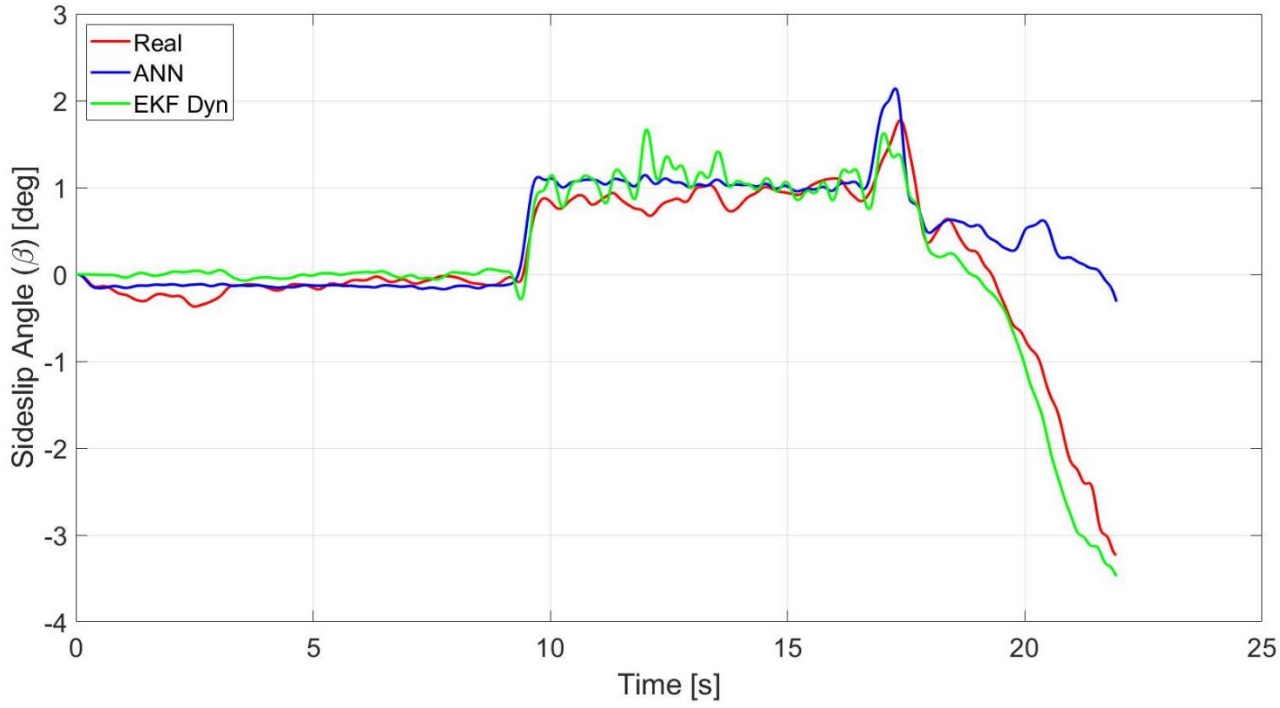


Figure 62. 70° step steer manoeuvre @ 125 kph.

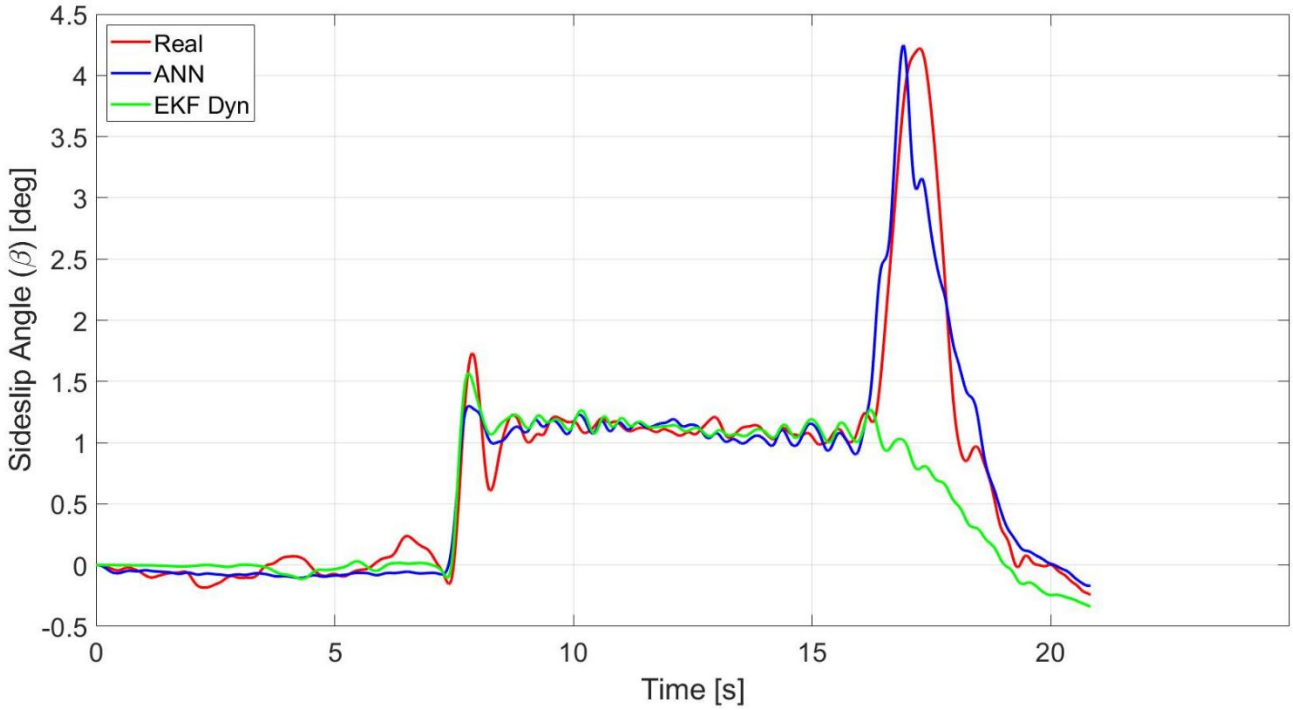


Figure 63. 90° step steer manoeuvre @ 125 kph.

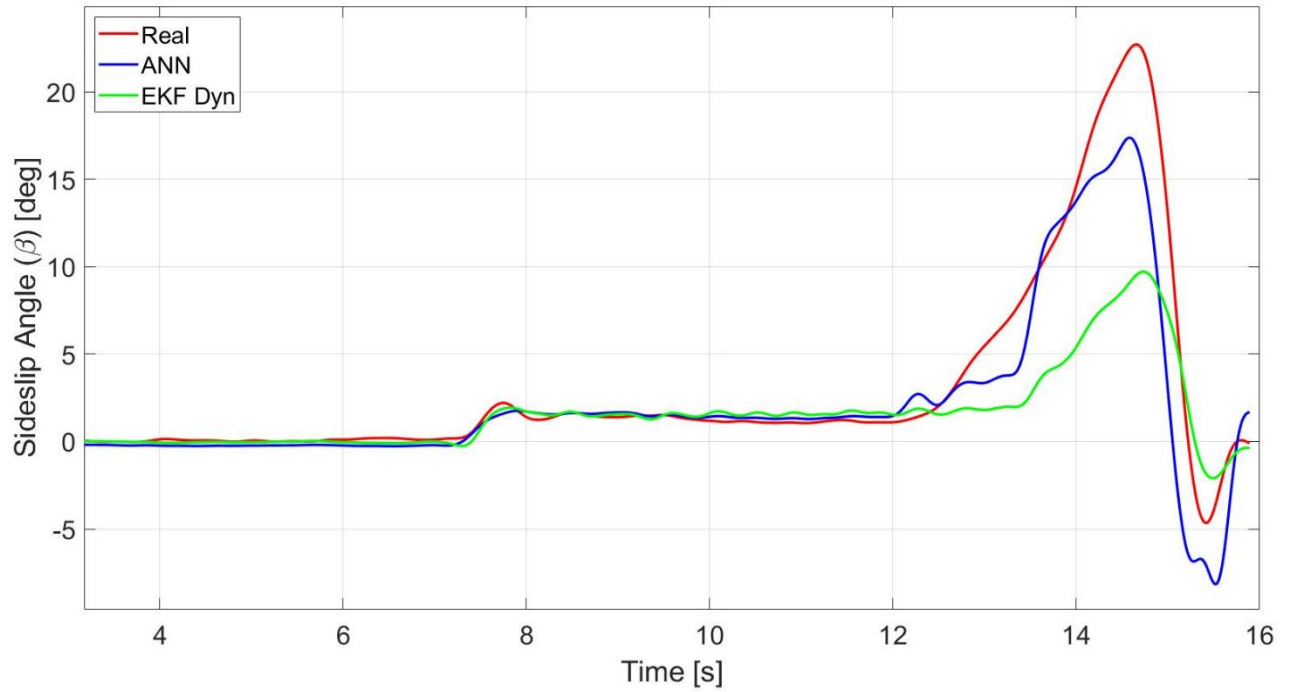


Figure 64. 70° step steer manoeuvre @ 125 kph.

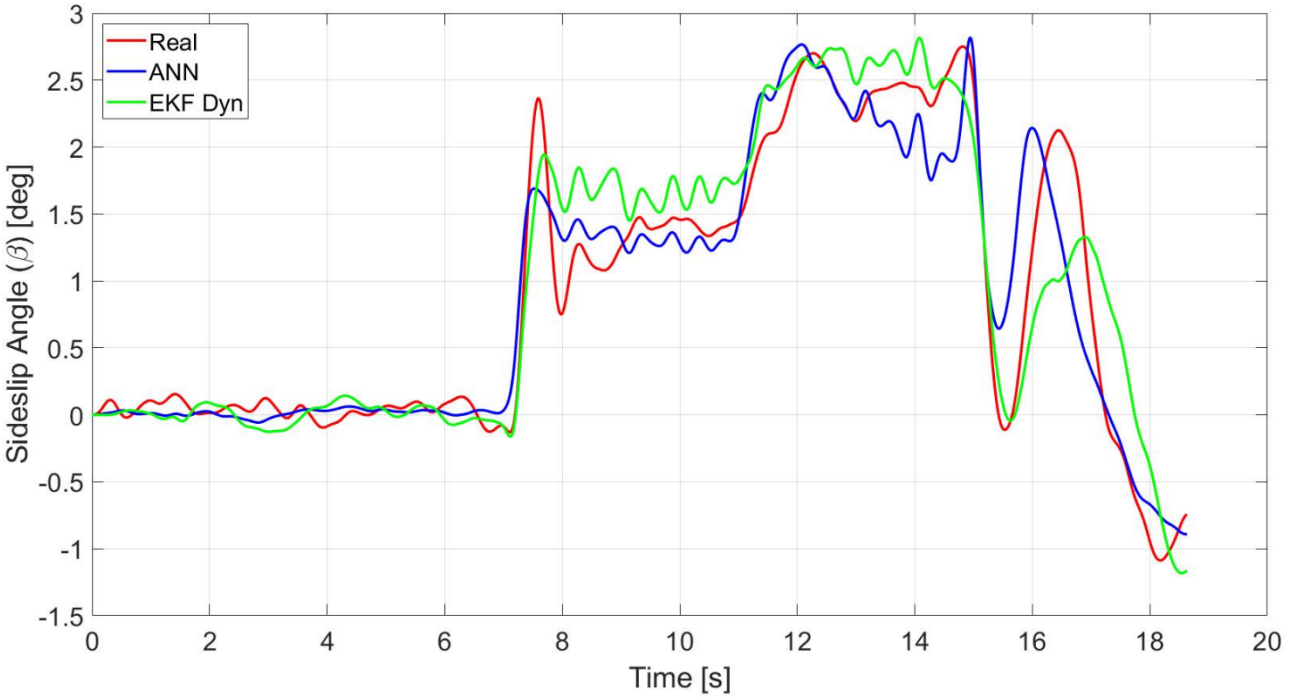


Figure 65. 90° step steer manoeuvre @ 150 kph.

4.2 Integration of ANN in Wintax®

The next step in the work was the integration of the neural network into Wintax®, a data analysis software used in motorsport produced by Marelli, Figure 66.

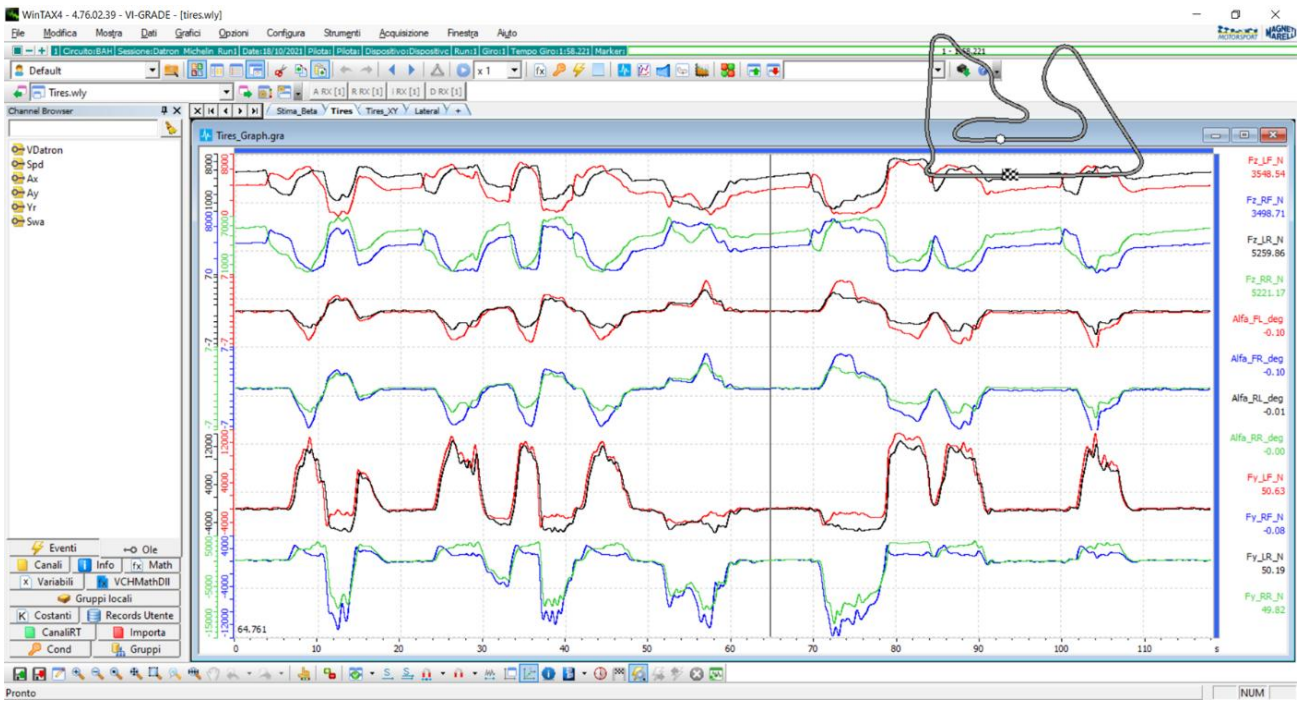


Figure 66. Screenshot of Wintax® data analysis software.

This software makes it possible to visualise data both in real time and in post-processing and allows the creation of different mathematical channels. The aim of this work is to report the mathematical function describing the neural network, created in Matlab®, within Wintax®.

The ANN is the same as described in the previous chapter, i.e. a simple network characterised by a middle layer with 10 neurons; a graphic representation of the neural network can be seen in the Figure 67.

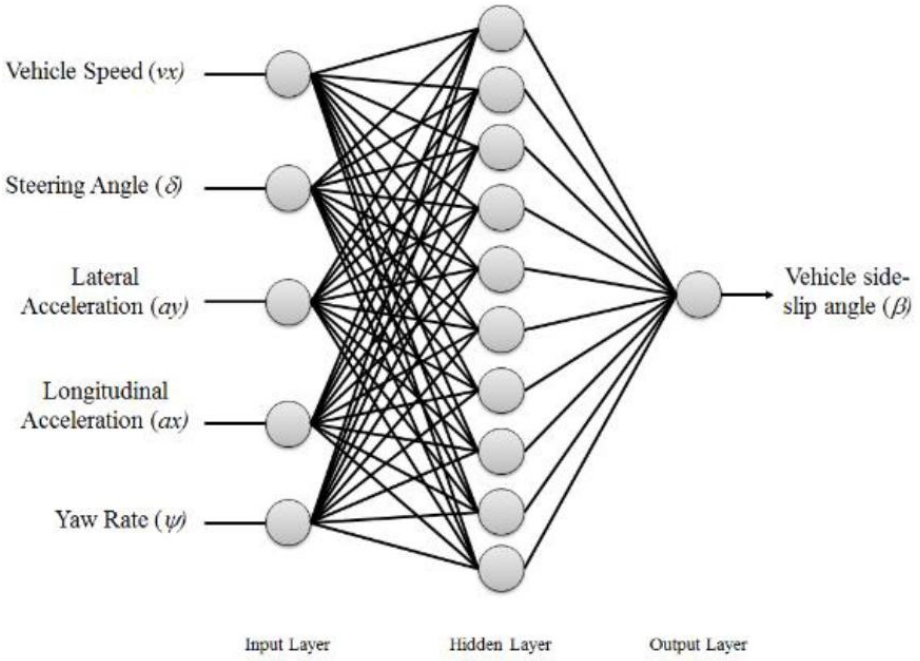


Figure 67. Graphical representation of the artificial neural network (ANN).

The network as input has 5 inputs:

- Vehicle Speed
- Steering Angle
- Lateral Acceleration
- Longitudinal Acceleration
- Yaw Rate

The output is the side slip angle (beta).

In order to be able to recreate the network on Wintax®, it was necessary to write the mathematical functions describing it and then transfer them into a VBscript. The equation 4.10 represents the mathematical function describing the ANN.

$$Y = \frac{W2 \cdot \left(\frac{2}{1 + e^{-2\left(b1 + W1 \cdot \left((X - x_o)x_g + x_b\right)\right)}} - 1 \right) + b2 - y_b}{y_g} + y_0 \tag{4.10}$$

Where Y is the output, in this case the slide slip angle. X is instead a 5x1 vector containing the input variables of the network.

x_o, x_g, x_b and y_o, y_g, y_b are the offset, gain and bias vectors referring to input X and output Y respectively.

$W1$ and $W2$ are the weights of each neuron associated with each input and output respectively. Specifically, $W1$ is a 10x5 matrix, representing 10 weights for each of the 5 inputs. $W2$, on the other hand, is a 10x1 vector.

The part of the equation in brackets represents the so-called activation function and is represented by a function described in equation 4.11 assuming $n = b1 + W1 \cdot \left((X - x_o)x_g + x_b\right)$.

$$\frac{2}{1 + e^{-2(n)}} - 1 \tag{4.11}$$

Equation 4.11 represents a sigmoid, its characteristic graph is shown in the Figure 68.

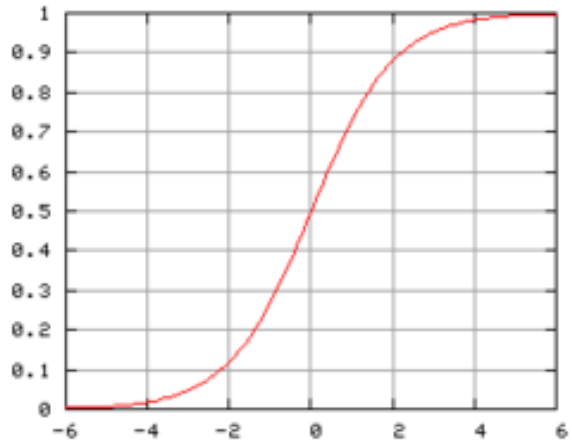


Figure 68. Graph of the sigmoid function used as an activation function in ANN.

As mentioned above, the mathematical model of the neural network was implemented in a VisualBasic script (VBs) in order to be handled as it works by Wintax®. The neural network was previously created using a proprietary Matlab® toolbox, which contains the algorithms required for training the network.

To make the translation between the 2 languages easier, a script was created in Matlab® that creates a text file containing all the network parameters (weights, bias, gain, offset) that can be easily pasted into VBscript.

Two neural networks were created to do a little validation. The first network was trained with real data from a GT-type car, having a sensor (Datron) for reading the side slip angle. The second network was instead trained by creating a virtual model of the vehicle with the available data. With this model, a series of manoeuvres and

laps were carried out in order to extend the domain as far as possible. The data collected was used to train the second network as the slide slip angle is a known state of the vehicle during the simulation.

Shown below are the slide slip angle values for one lap of the Barcelona circuit, superimposing the side slip angle calculated with the Datron sensor and those estimated with the two neural networks. (Figure 69 and Figure 70)

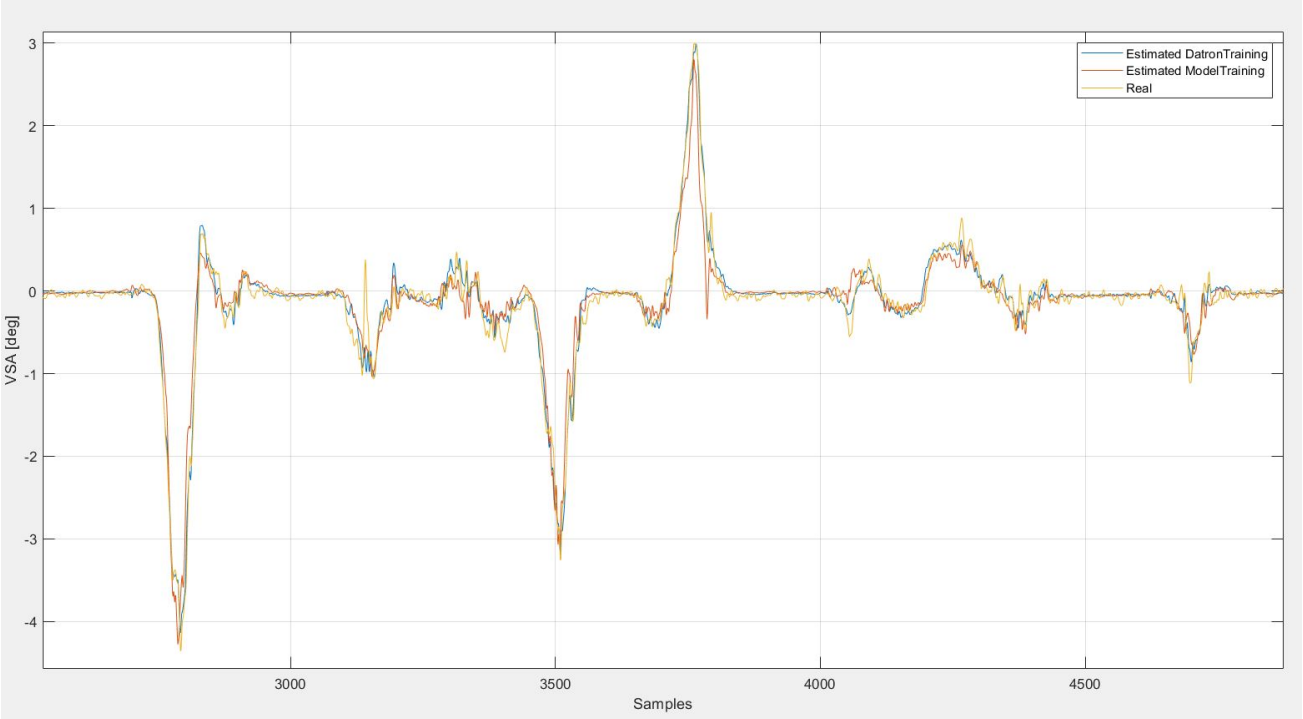


Figure 69. Vehicle Side Slip Angle measured by sensor (yellow), estimated with ANN trained with model (orange), estimated with ANN trained with real data (blue) during a qualifying lap.

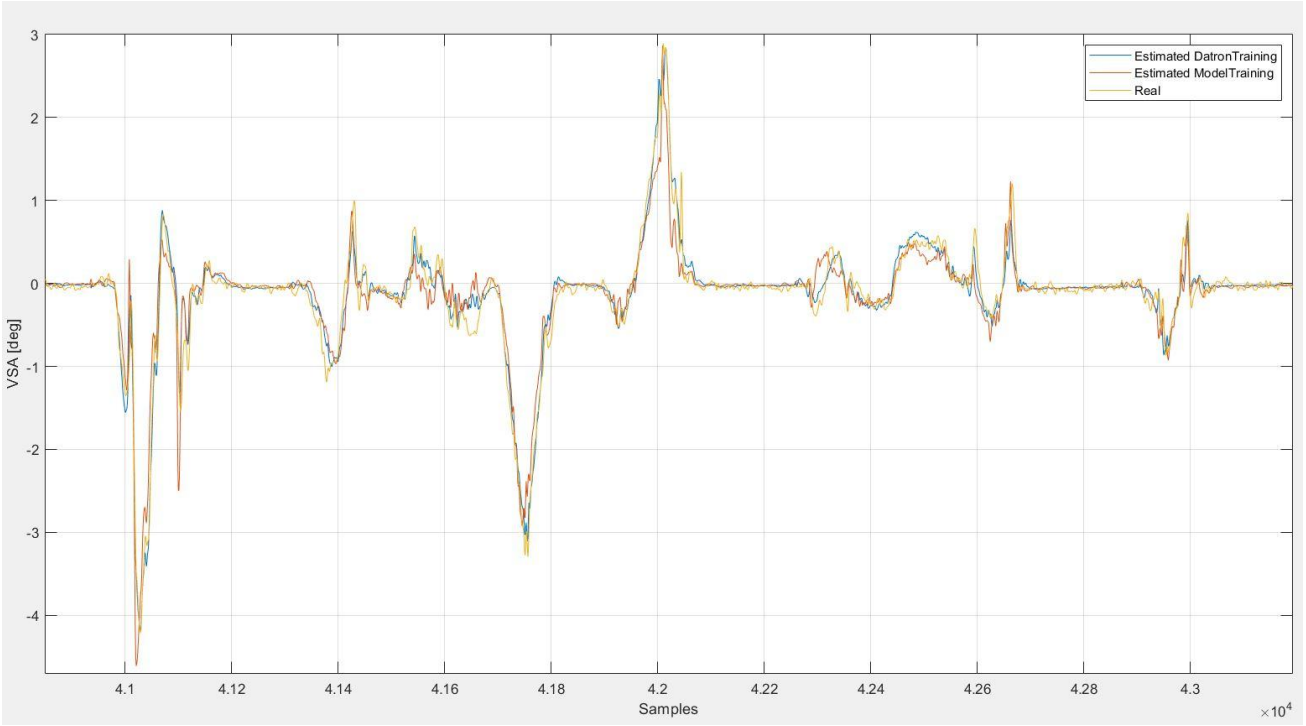


Figure 70. Vehicle Side Slip Angle measured by sensor (yellow), estimated with ANN trained with model (orange), estimated with ANN trained with real data (blue) during a race lap.

The results are also quite satisfactory in this case, and it was found that the neural network was correctly implemented in Wintax®.

Finally, the Figure 71 also shows the Wintax® screenshot where it is possible see the channel created with the neural network for estimating the slide slip angle, superimposed on the channel coming from the sensor.

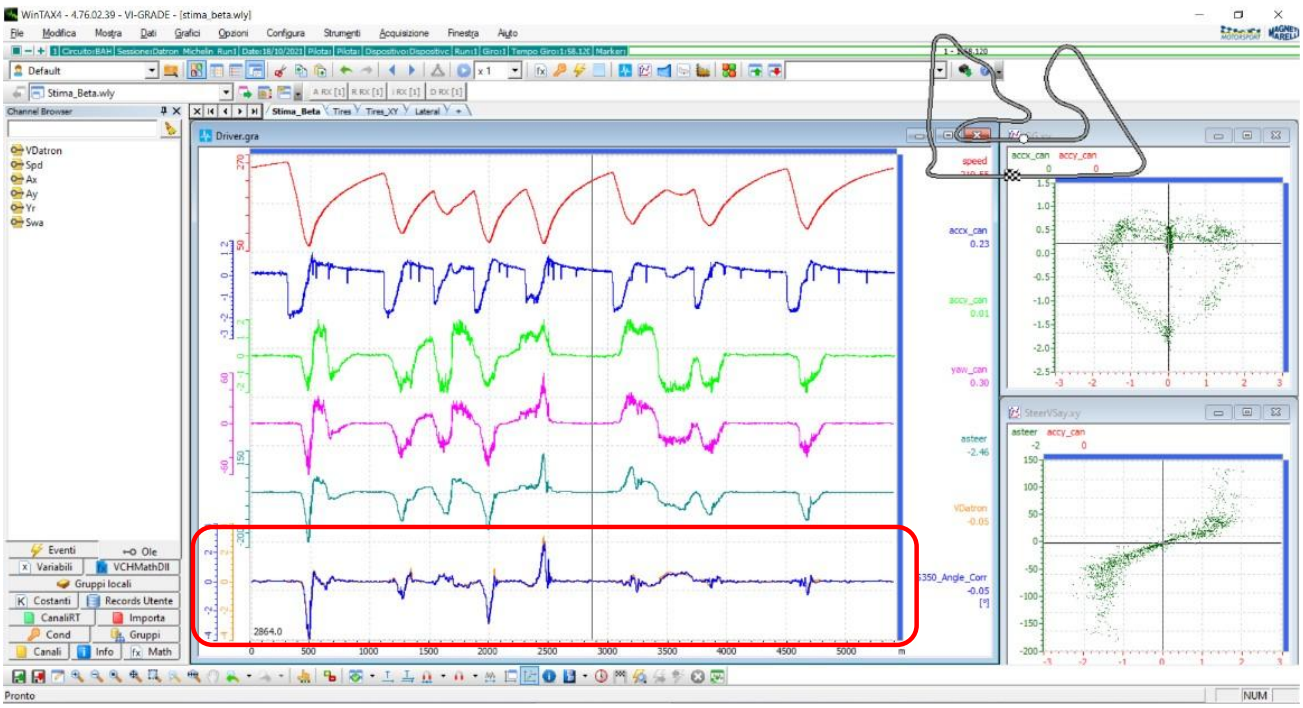


Figure 71. Screenshot of Wintax with the mathematical channel of VSA estimation via ANN.

5 Vehicle Subsystem Integration

This chapter presents the development of two vehicle subsystems, both modelled in Matlab Simulink®: the "inboard" part of a suspension and a limited-slip differential.

5.1 External Inboard Suspension in Simulink®

This work is the result of a research project in collaboration with YCOM, an Italian company that deals with the design and manufacturing of racing cars.

The aim of this work was to create a Simulink® model to manage the inboard part of a suspension in co-simulation with ViCarRealTime®. Inboard suspension refers to all those elements (elastic and/or damping) that make up the inner part of the suspension, i.e. not the arms. The anti-roll bar, heave element, roll element, etc. are part of the inboard suspension.

The need to create an external model arises mainly for two reasons: the first is that ViCarRealTime allows the modelling of a limited number of elements that make up the suspension inboard. The second reason is in order to be able to manage an active suspension, in particular this suspension must be equipped with an active Heave Element capable of dynamically changing the ride height of the axle.

Shown in the Figure 72 a 3DSusp screenshot is the front suspension in question, equipped with Heave Element, Roll Element and C-Shape Antiroll bar.

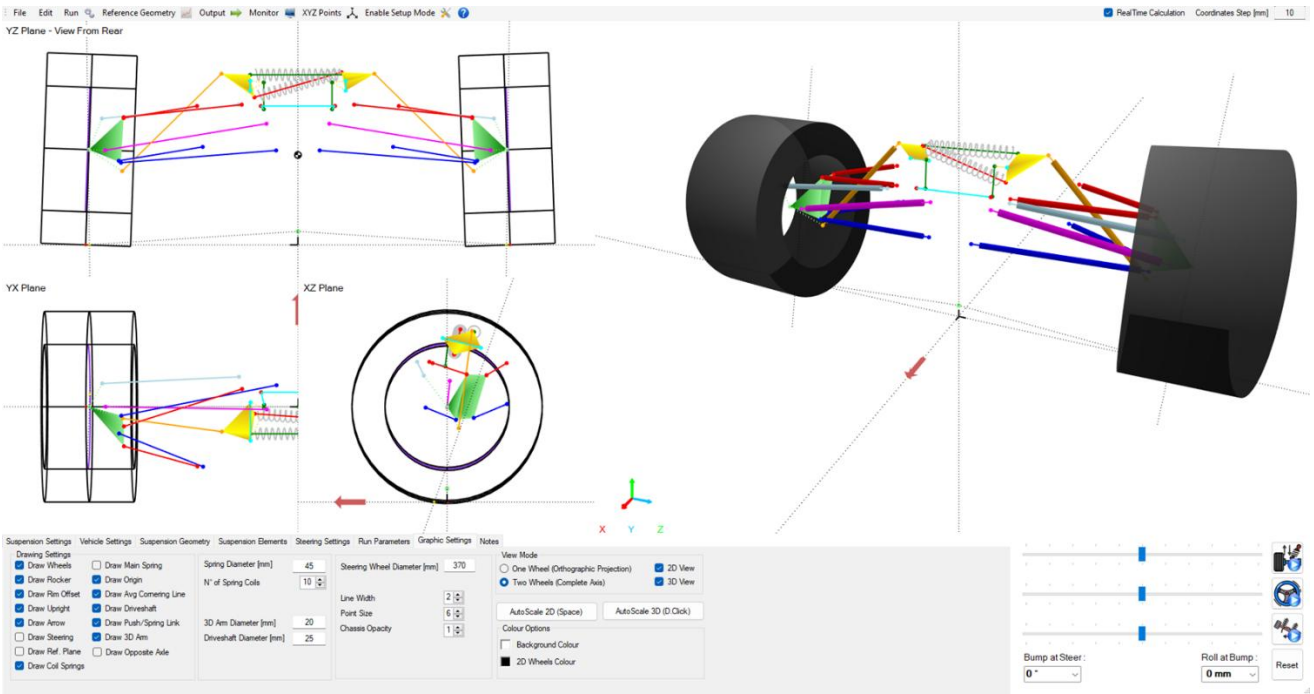


Figure 72. 3DSusp screenshot of the front suspension mentioned.

Figure 73 shows a screenshot of the rear suspension.

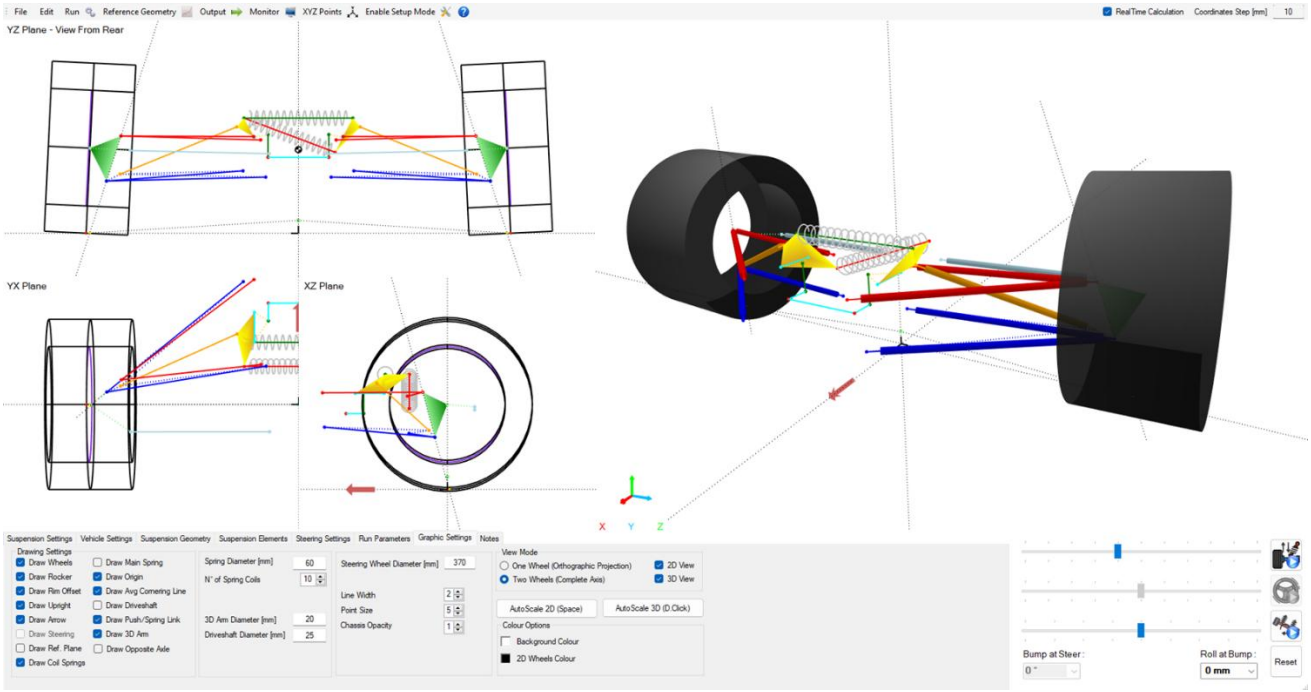


Figure 73. 3DSusp screenshot of the rear suspension mentioned.

5.1.1 Model Description

In order to be able to manage suspension externally, one must first deactivate the internal computation of all helical or damping elements in ViCarRealTime®. After that, four inputs must be provided to the solver:

- Auxiliary_vertical.force.L1
- Auxiliary_vertical.force.R1
- Auxiliary_vertical.force.L2
- Auxiliary_vertical.force.R2

These four channels represent the four vertical forces, one for each wheel of the vehicle, that the suspension exerts as a reaction on the suspended mass. These forces are then virtually applied to the vehicle chassis and applied at the point whose coordinates correspond to the wheel centre, for each wheel.

Instead, five channels are read from the solver:

- Suspension.Jounce.L1
- Suspension.Jounce.R1
- Suspension.Jounce.L2
- Suspension.Jounce.R2
- Driver_demands.steering

The first four channels represent the jounce of each individual wheel, that is the vertical displacement. The last channel, on the other hand, is the steering wheel angle imposed by the driver, since the suspension kinematics are also influenced by the steering for the front axle.

A screenshot of the Simulink® file can be seen in the Figure 74 to better understand how it is set up.

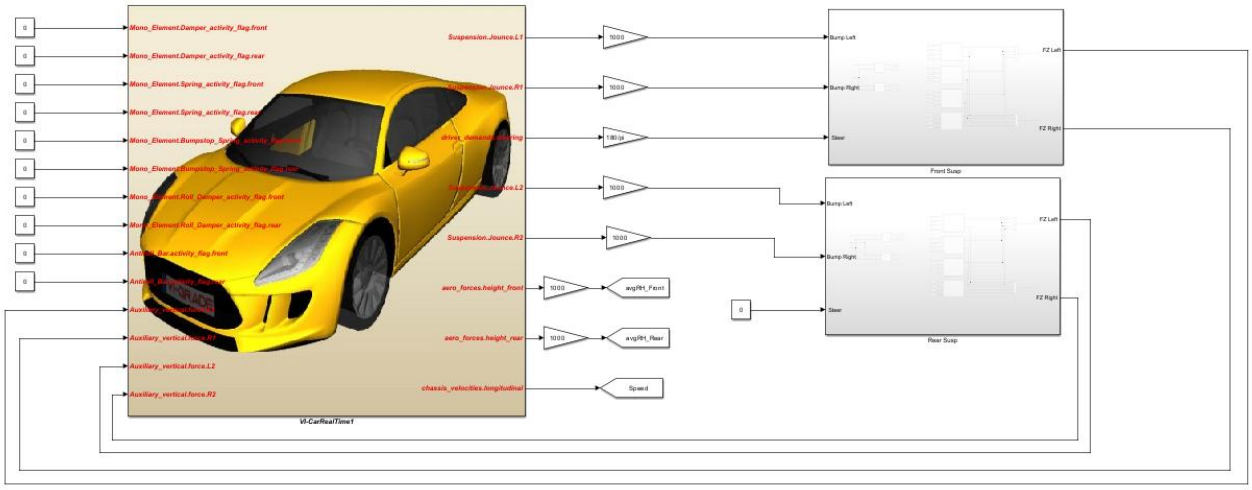


Figure 74. Screenshot of the Simulink® file handling inboard suspension.

The two sub-systems on the right contain the actual inboard suspension model for the front and rear axles respectively. The set of inputs linked to constants equal to zero serve to deactivate ViCarRealTime® internal computation of the elements.

The objective is to calculate the equivalent force at wheel centre, exerted by the suspension elements, as a function of jounce and steering.

The equivalent vertical force applied by the element applied at the wheel centre, i.e. that required by the ViCarRealTime® solver, is described in Eq. 6.1.

$$F_{at\ wheel} = \frac{F_{at\ element}}{MR_{element/rocker} \cdot MR_{rocker/wheel}} \tag{5.1}$$

Where $MR_{element/rocker}$ is the motion ratio between the element and the rocker, calculated as the differential ratio between the deformation of the element and the angle of the rocker relative to the corner being considered. $MR_{rocker/wheel}$, on the other hand, is the motion ratio between the rocker and the respective wheel.

In order to know the force at the element, it is necessary to calculate the deformation if it is an elastic element or the deformation rate if the element has a damping part. Obviously, stiffness and damping must be known. Element deformation is a function of the jounce and steer of the individual suspension, and particularly the angles of the two rockers of the axle. For this, a lookup table was created in 2 dimensions to calculate the deformation of the element, which describes the deformation of the element as a function of the rocker angles.

Knowing the deformation, it is possible to calculate the force of the helical element simply by multiplying it by the stiffness or by using a lookup-table if the stiffness is non-linear. If the element has a damping component, strain rates are used by simply deriving the newly calculated strain value with respect to time.

A screenshot of the Simulink® file for calculating the deformation of element, the forces on it and the equivalent vertical force on the wheel centre can be seen in the Figure 75, where the heave element is given as an example.

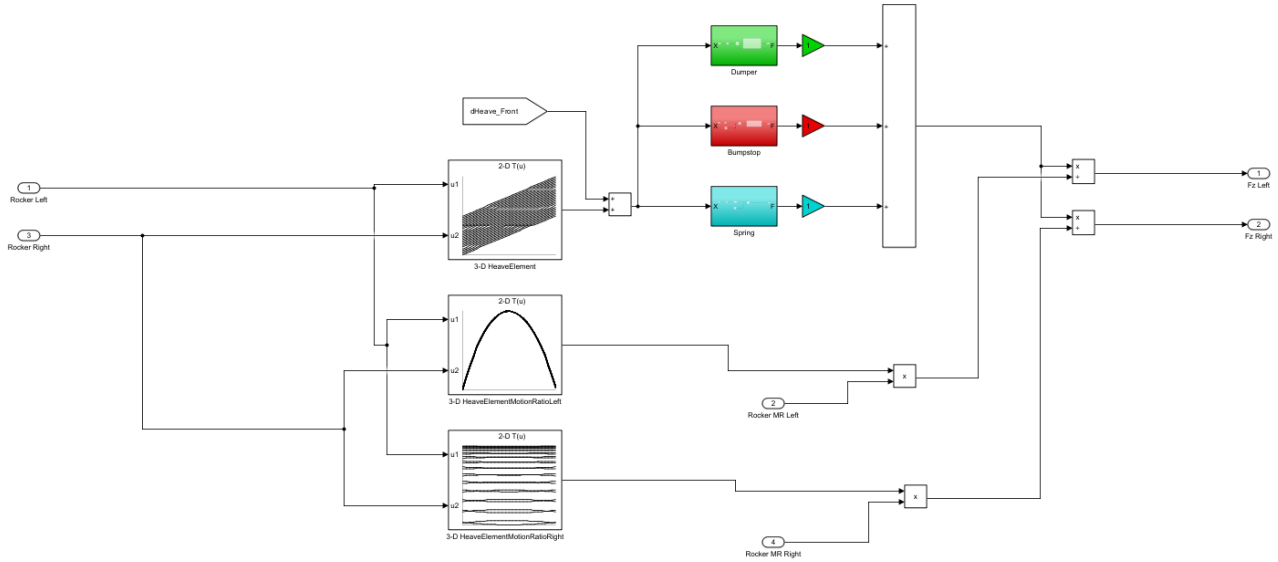


Figure 75. Screenshot of the Simulink® file for calculating the deformation of element, the forces on it and the equivalent vertical force on the wheel centre

The 2 input channels on the left can be seen which are the 2 rocker angles, in this case relating to the front axle as we are referring to the heave element of the front axle as an example. The way in which angles and rockers are calculated is explained later.

It can be seen that the two rocker angles enter the first lookup-table describing the deformation of the heave element. This lookup-table is shown as a surface in the Figure 76.

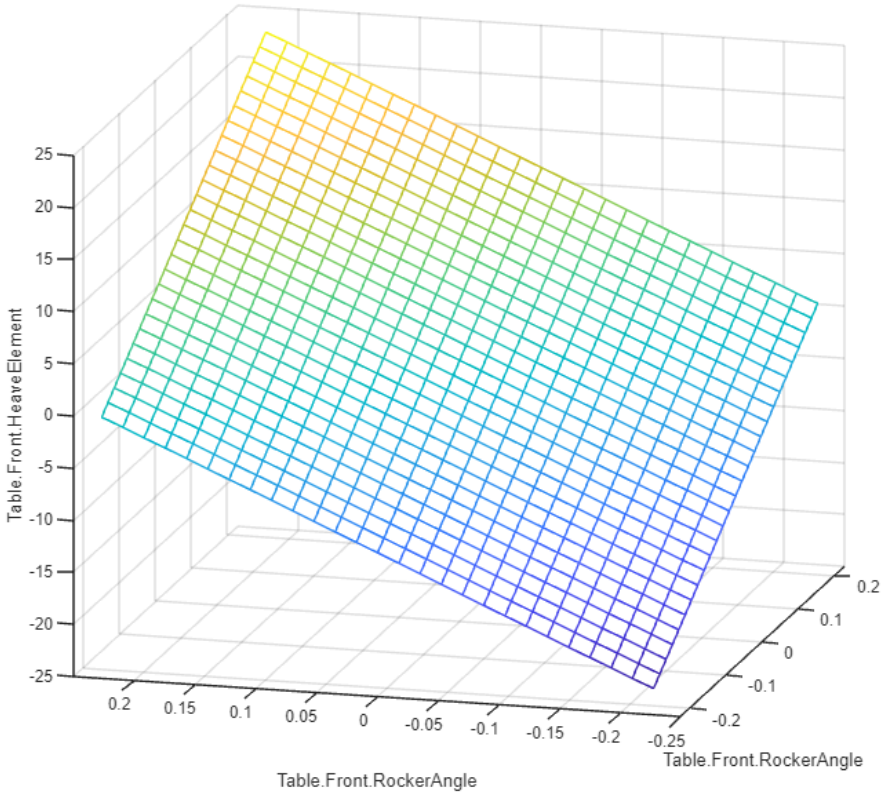


Figure 76. Screenshot of the surface representing the deformation of the heave element as a function of the two rocker angles.

Again, analysing the Figure 75, the deformation of the element becomes the input of 3 blocks representing the elastic and damping components of the heave element. The first block represents the damper, the second the bumpstop and the third the spring. In the Figure 77, these three components are shown in detail.

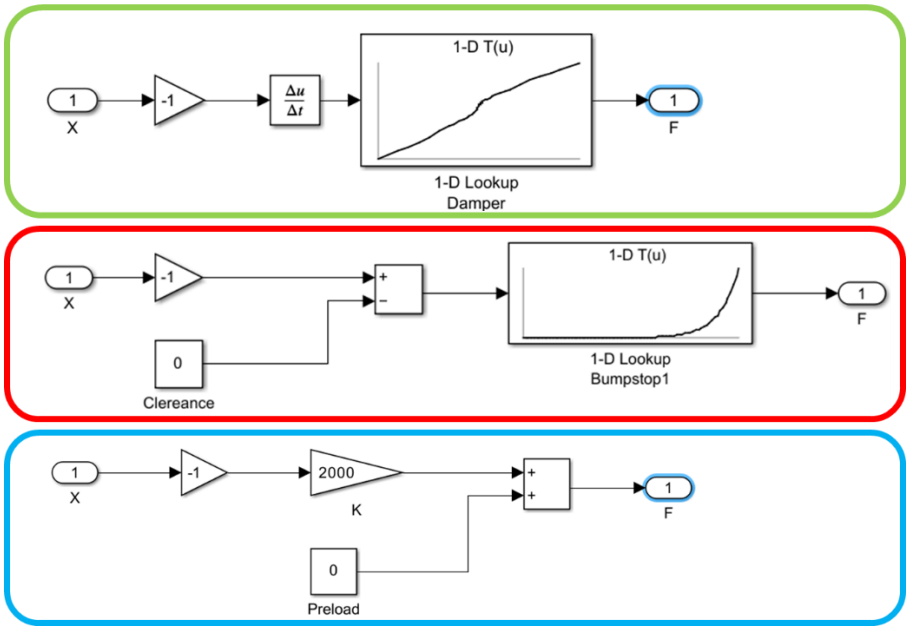


Figure 77. Screenshots of the 3 components of the heave element, in order: Damper, Bumpstop, Spring.

In the damper, it can be seen that the deformation is derived with respect to time and enters a lookup-table that describes the damping force as a function of deformation velocity.

In the bumpstop, the deformation instead enters a lookup-table that describes the force in a particularly non-linear manner. One can also note the possibility of adding an offset representing the clearance of the bumpstop.

In the spring, the deformation of the heave element is simply multiplied by its stiffness to calculate the force. A constant force can also be added as preload.

Looking at the Figure 75 again, it can be seen that the 3 component forces are added together and then divided by the product of the two motionratios to obtain the equivalent vertical force applied at the wheel centre.

The motionratios between element and rocker are 2 and are described by the other 2 lookup-tables shown in the Figure 75. The motionratios between element and rocker are 2 and are described by the other 2 lookup-tables shown in the Figure 75. They are a function of the 2 rocker angles. In the Figure 78, the surface describing the motionratio between element and left rocker of the heave element is shown as an example.

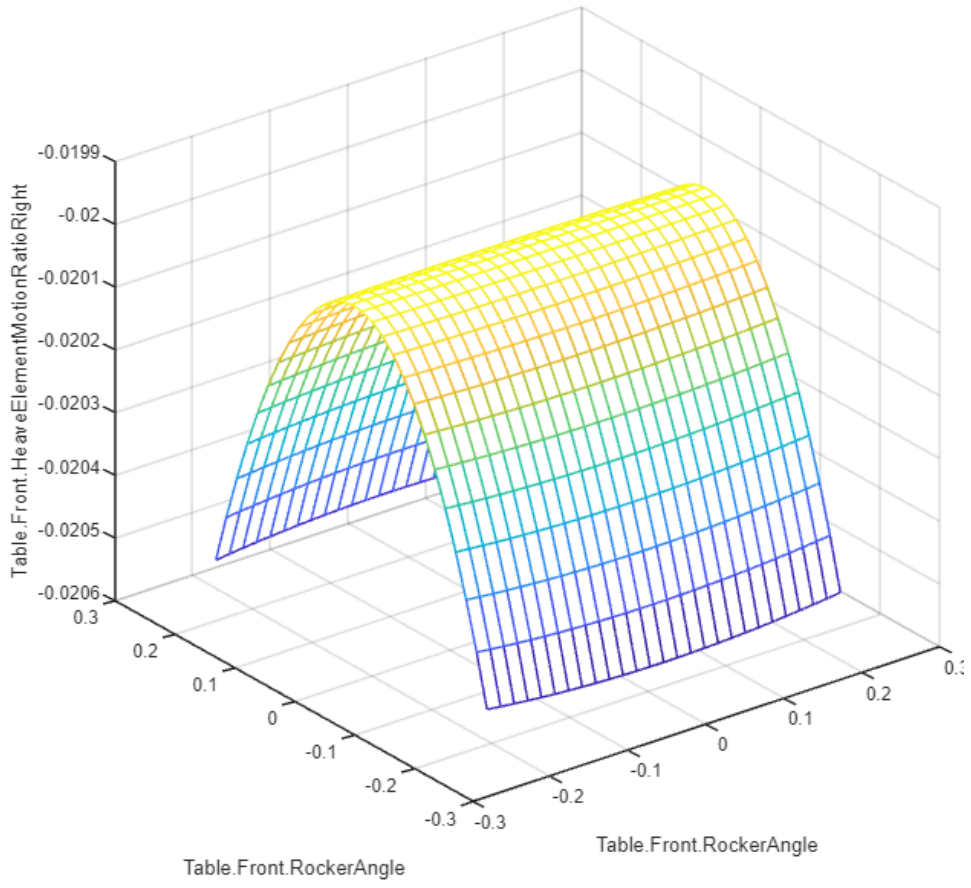


Figure 78. Screenshot of the surface describing the motionratio between element and left rocker of the heave element

Finally, in the Figure 75 we can also see the presence of an input called “dHeave”, this represents an offset on the deformation of the element and is the variable that is used by any active suspension logic. In this way it is

possible, via an actuator, to change the length of the heave element at will by changing the ride height of the axle.

The Figure 79 shows the inside of the Simulink block that manages the entire front suspension. On the right, it can be seen the blocks that manage all the suspension elements: heave element, roll element, ARB Cbar and torsion bar.

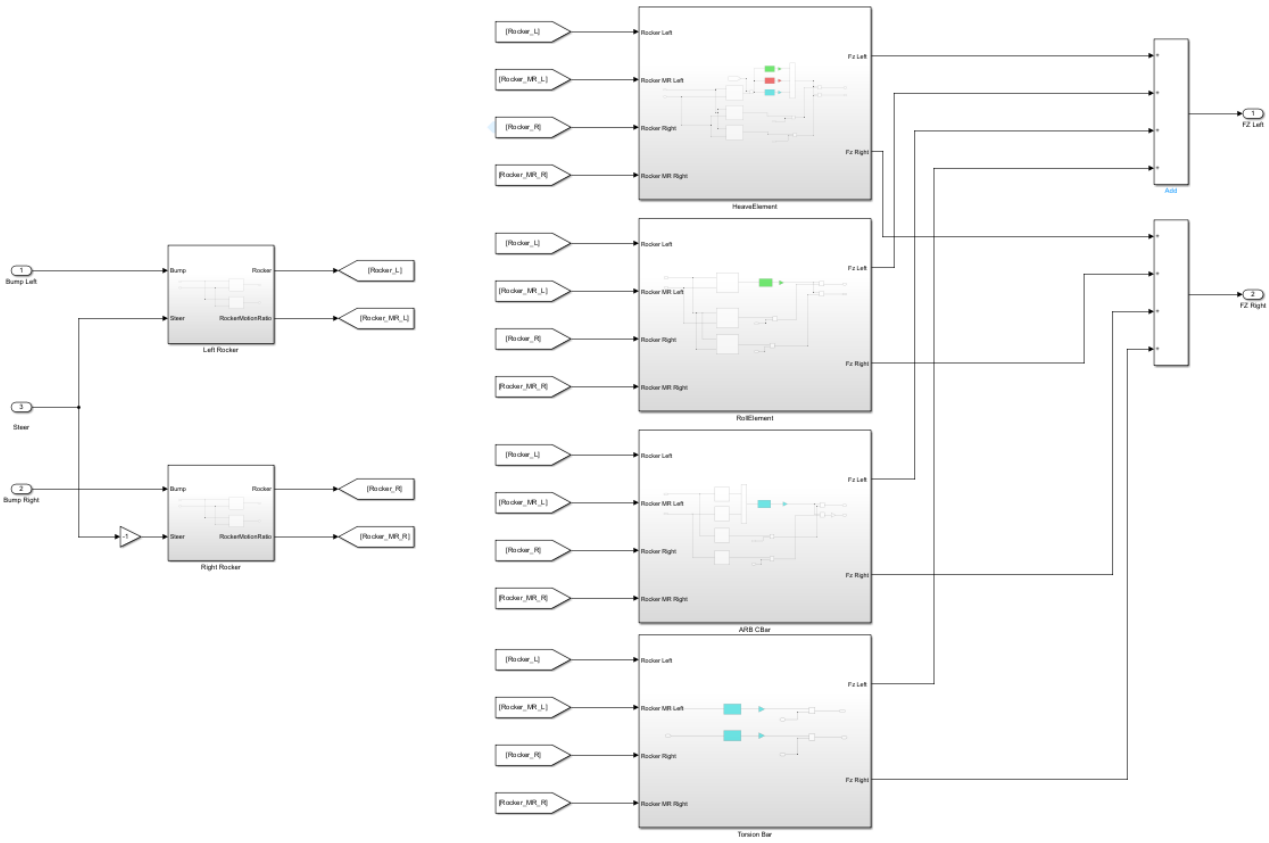


Figure 79. Screenshot of the inside of the Simulink® block that handles the front suspension.

The roll element is entirely analogous to the heave element just described. The deformation of the anti-roll bar, on the other hand, is calculated in a slightly different way, and the Figure 80 shows the inside of the Simulink® block that handles it.

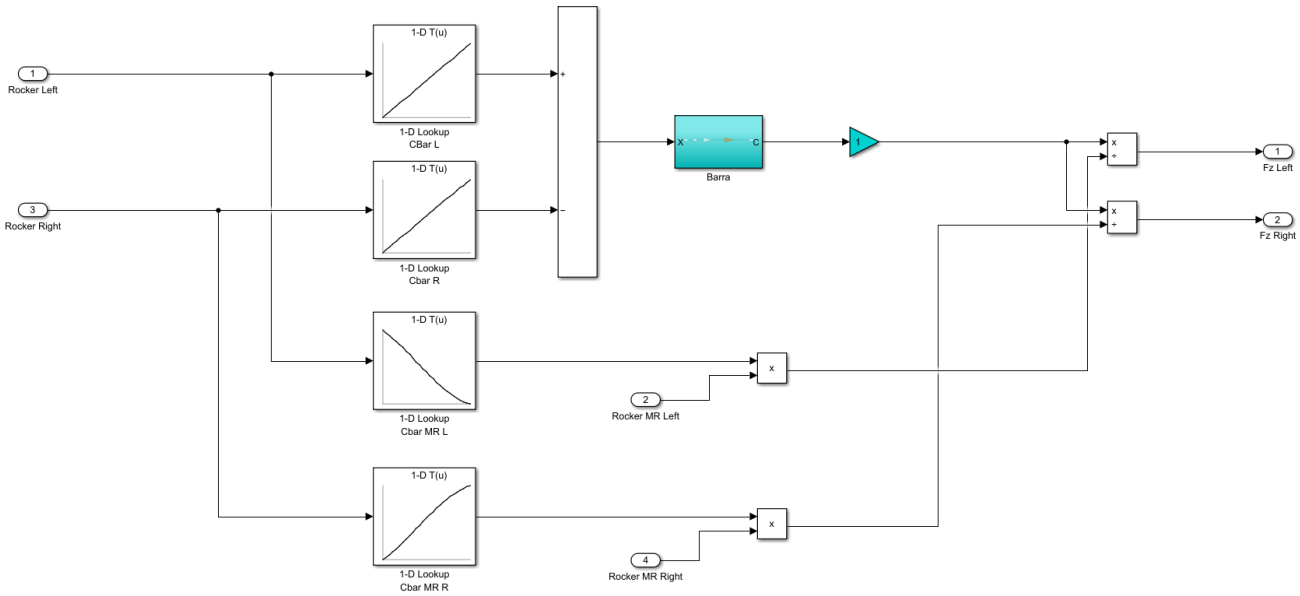


Figure 80. Screenshot of the Simulink® block that manages the c-shape anti-roll bar of the front suspension.

The main difference is the fact that the rotations of the two ends of the bar are calculated, each according to the relative rocker. To calculate the bar torsion, the difference between the two rotations is made. The subsequent calculation to find the equivalent vertical force is analogous to that seen above, with the difference that not the force on the element is calculated but the torque. Finally, it can be seen that the two motion ratios between element and rocker have opposite signs, precisely because the torque to which the anti-roll bar is subjected applies two forces of opposite signs to the wheels.

The calculation of the torsion bars, on the other hand, is rather easy since their torsion is equivalent to the angle of rotation of the rockers, so the torque exerted on the torsion bar is simply calculated by multiplying the angle of the respective rocker by the torsional rigidity of the bar. As it can be seen in the Figure 81.

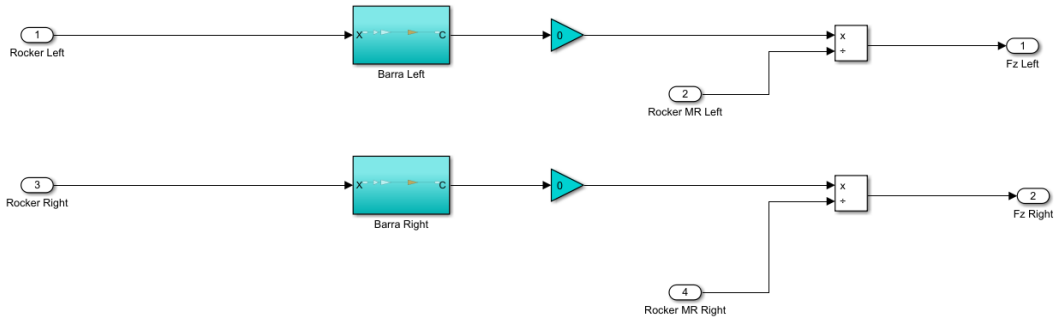


Figure 81. Screenshot of the Simulink® block that handles the calculation of forces exerted by torsion bars.

To calculate the equivalent vertical force, simply divide the torque of the torsion bar by the motion ratio between rocker and wheel.

Finally, on the left-hand side of the Figure 79 are the 2 blocks that manage the 2 rockers of the front suspension, each of which calculates the rotation of the rocker and the motion ratio with respect to the wheel, depending

on the jounce and steer of the corresponding wheel. The block that operates the left rocker is shown as an example in Figure 82.

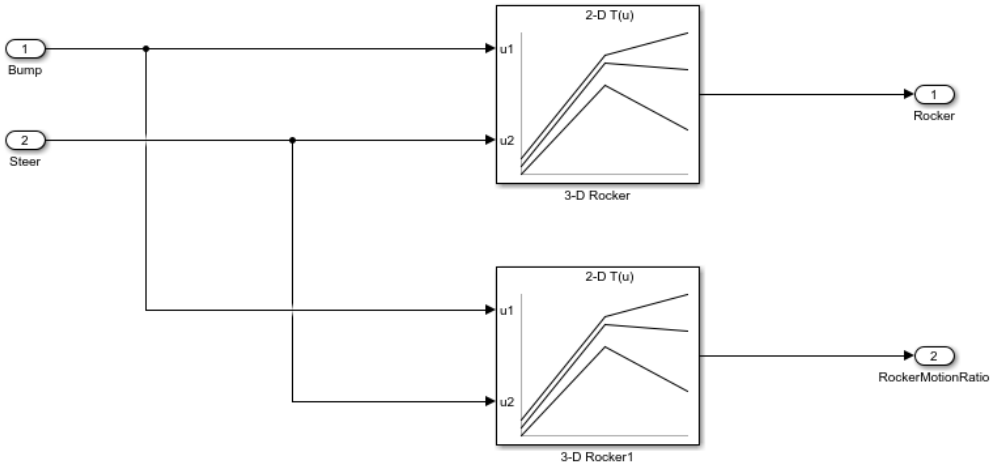


Figure 82. Simulink block managing the left rocker of the front suspension.

The 2 lookup-tables are both functions of jounce(bump) and steer. The two surfaces describing the rocker angle and its motion ratio, respectively, are shown in the Figure 83.

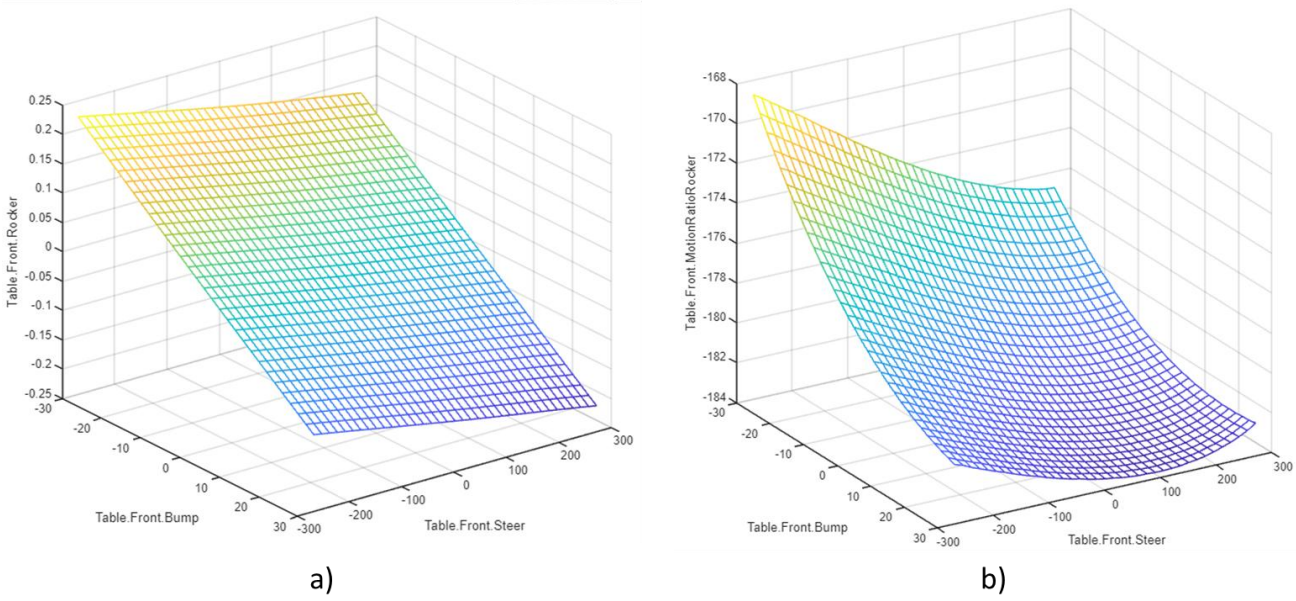


Figure 83. a) Surface representing the rotation angle of the left rocker as a function of bump and steer. b) Surface representing the motion ratio between rocker and left wheel as a function of bump and steer.

For reasons of space and compactness, the rear suspension is not shown as it is like what just seen before.

All the lookup tables used in the Simulink® file presented in this chapter were calculated using the same algorithms used in 3DSusp and presented in the first chapter.

The vectors and matrices that populate these tables are generated by a script created in Matlab® that has as input the points representing the kinematics of the two suspensions, rewriting it in MATLAB language.

In addition, external model management allows suspension parameters to be changed in real-time while driving on the simulator. Without having to stop and restart the simulator each time.

With this model, it was therefore possible to manage the inboard part of a suspension externally with a Simulink® file. This also made it possible to develop active suspension logic together with YCOM to dynamically manage the ride height of the vehicle. As an example, two simulations of the vehicle in a straight line are shown. The first simulation is carried out with passive suspension. In the second simulation, after 5 seconds the heave element is shortened instantly by 5 mm, causing the front axle to lower. The heights of the vehicle's front and rear axle from the ground for the two simulations are shown in the Figure 84. In red is the simulation with active suspension.

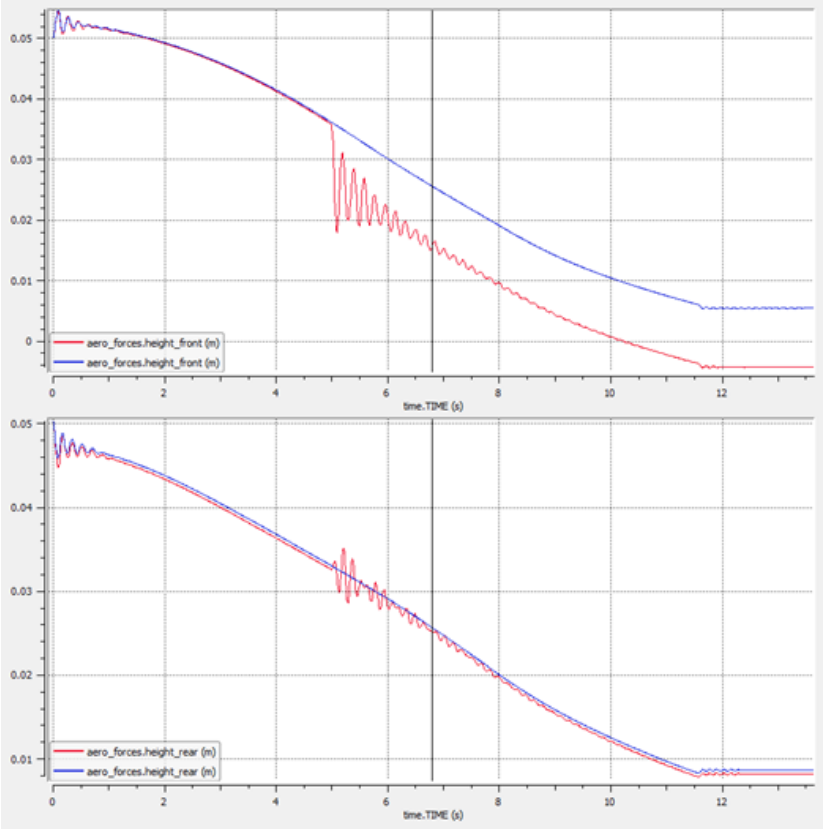


Figure 84. Front and rear ride height of the vehicle. In red the simulation with active suspension. In blue the simulation with passive suspension.

As the heave element shortens, the vehicle at the front lowers. Since the change in length of the heave element is instantaneous, the dynamic effects on the simulation are clearly visible from the oscillations. The stiffness of the suspension is not changed, as the curves remain parallel.

5.2 External Limited Slip Differential in Simulink®

A limited slip differential (LSD) is a differential that allows the two axle shafts to be connected in certain situations, to solve the classic driveability problem of an open differential and to distribute torque between the two wheels of the same axle to improve vehicle performance [58] .

An LSD generally consists of a clutch pack that connects the two axle shafts or one of the axle shafts with the carrier. The Figure 85 shows the clutch discs on a generic LSD.

Figure 85. Clutch discs on a generic LSD.

In a passive differential, the clutch packs are usually preloaded and there is a system that uses a pin and ramps to exert a force on the clutch plates depending on the torque input to the differential. In the Figure 86 it is possible to see a passive limited slip differential with the ramps to exert pressure on the clutch plates and change the differential lock.

Figure 86. Mechanical LSD with pin and ramps.

Detailed information on the mechanical functioning of LSD and its effects on vehicle dynamics can be found in [58], [59], [60], [61].

With a hydraulic system that applies variable pressure on the clutch plates, an active LSD can be achieved instead.

This work was also the result of a research project with YCOM, a company involved in the design and manufacturing of racing cars. The aim is to externally manage a limited slip differential model to actively control it. In addition, the limited slip differential of ViCarRealTime® basic model, which was only passive, also presented instability problems.

5.2.1 Model Description

Similar to the previous chapter, the model was created in Simulink® to be used in co-simulation. In order to manage a subsystem externally, it is necessary to provide the ViCarRealTime® solver with the appropriate inputs:

- Differential_Rear.wheel_rear_left_torque
- Differential_Rear.wheel_rear_right_torque
- Differential_Rear.diff_central_rear_outputshaft_angle
- Differential_Rear.diff_central_rear_outputshaft_omega
- Differential_Rear.diff_central_rear_outputshaft_alpha

These channels refer in this case to a rear differential, but it is of course possible to do the same with a front differential by changing 'rear' to 'front'. The first two channels are the torques on the two semi-axes output to the differential. The other three channels refer to the alabaster input to the differential and are angle, angular velocity, and angular acceleration, respectively.

Instead, some output from the solver is taken as input to the model:

- Differential_Rear.wheel_rear_left_angle
- Differential_Rear.wheel_rear_left_omega
- Differential_Rear.wheel_rear_left_alpha
- Differential_Rear.wheel_rear_right_angle
- Differential_Rear.wheel_rear_right_omega
- Differential_Rear.wheel_rear_right_alpha
- Differential_Rear. diff_central_rear_outputshaft_torque

The first three channels refer to the left semi-axis and represent angle, angular velocity, and angular acceleration respectively. The second three channels are analogous to the first three but refer to the right semi-axis. The last channel is the shaft torque input to the differential. A screenshot of the Simulink® file is shown in the Figure 87 to display the inputs and outputs just explained.

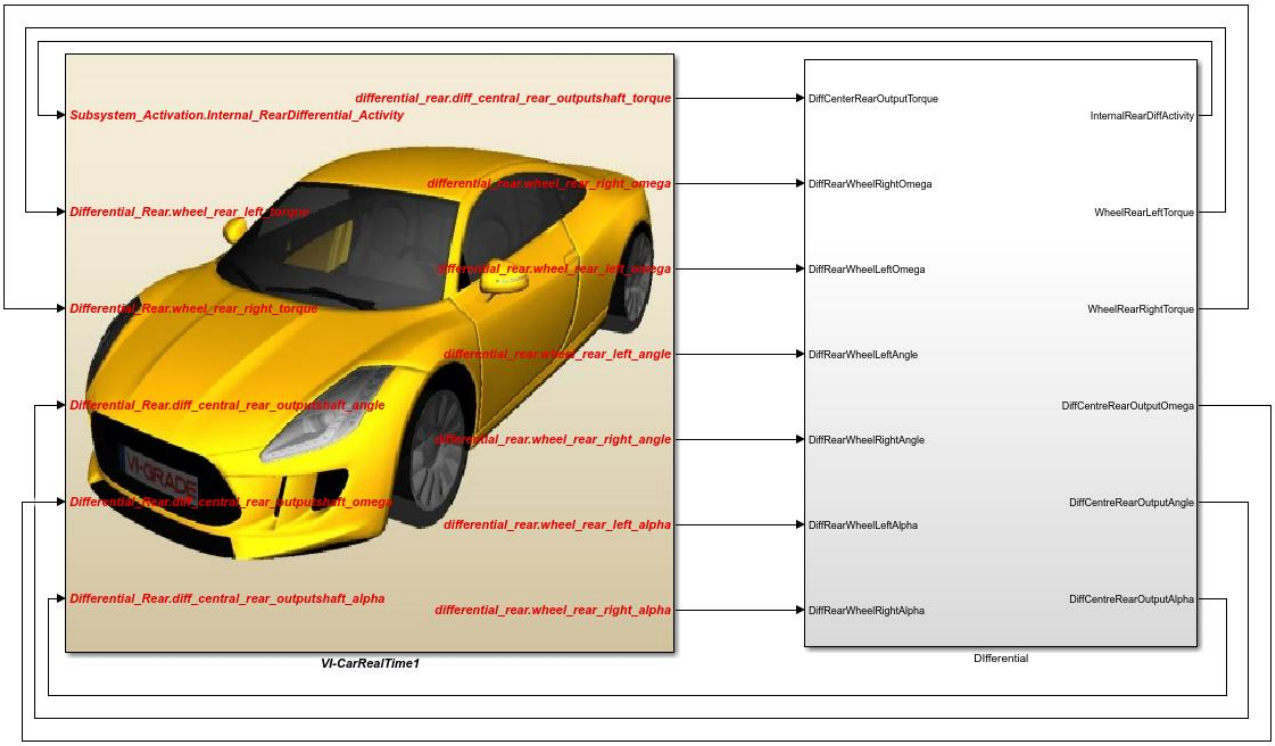


Figure 87. Screenshot of the Simulink® file of the LSD model.

The mathematical modelling of the differential, without considering the locking part, is quite simple and is described below.

The angle of rotation θ_{IN}, the velocity ω_{IN}, and acceleration α_{IN} of the shaft input to the differential are described by the equations 5.1, 5.2 and 5.3.

$$\theta_{IN} = r \cdot \frac{\theta_L + \theta_R}{2} \tag{5.1}$$

$$\omega_{IN} = r \cdot \frac{\omega_L + \omega_R}{2} \tag{5.2}$$

$$\alpha_{IN} = r \cdot \frac{\alpha_L + \alpha_R}{2} \tag{5.3}$$

Where subscripts L and R refer to the left and right axle shafts respectively, and r represents the bevel gear ratio of the differential.

The input torque acting on the differential carrier is calculated by multiplying the input shaft torque by the drive ratio r. If the differential were of the open type, the torque on the 2 semi-axles would be equal and would be described by the equation 5.4.

$$T_L = T_R = r \cdot \frac{T_{IN}}{2} \tag{5.4}$$

In fact, in an open differential, torque is divided equally between the two wheels. The Simulink® screenshot is shown in the Figure 88, displaying the above.

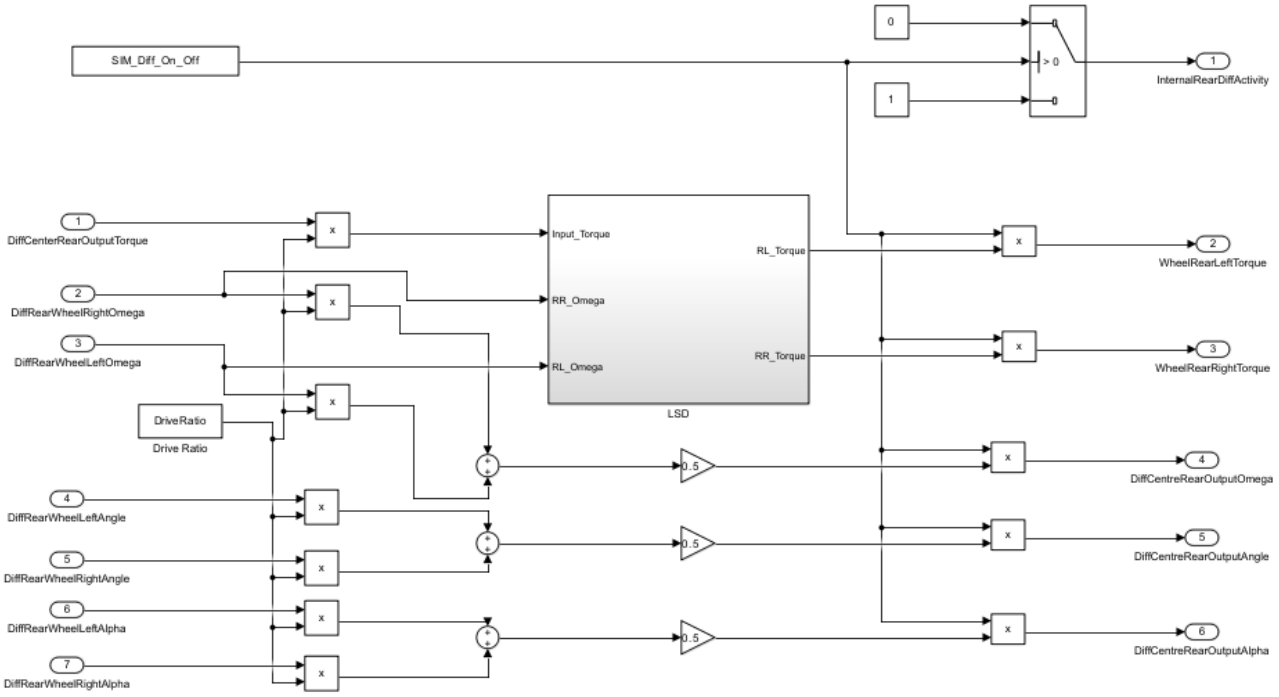


Figure 88. Simulink® screenshot describing the kinematic behaviour of the differential.

The subsystem called 'LSD' handles the differential lock instead, and a screenshot of its interior is shown in the Figure 89.

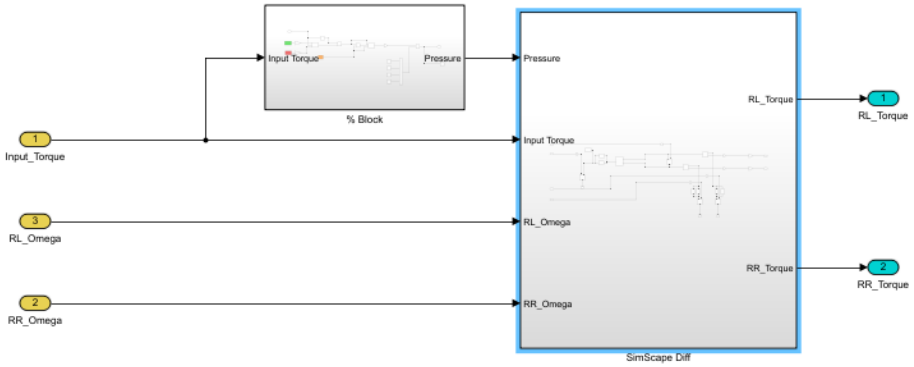


Figure 89. Screenshot of the 'LSD' subsystem that manages the locking of the two axle shafts.

The big difficulty in modelling locking is to find a correct mathematical model to describe the behaviour of the clutch. To overcome this problem, we chose to use Simulink's Simscape® Driveline library, which allows mechanical transmissions to be modelled intuitively. Furthermore, the solver used integrates seamlessly with the Vi-CarRealTime® simulation environment and Simulink®, and it can also be used in real-time simulations.

In particular, Simscape® driveline offers a differential block, and a block that allows a clutch pack actuated by a certain pressure. This made it possible to create a Simscape® model of the differential using an ideal

torque generator to provide the input torque to the differential, which is known. In addition, two ideal speed generators, which is known, were used to supply it to the two semi-axles, which were connected by a clutch pack. A screenshot of the differential model created with Simscape® Driveline can be seen in the Figure 90.

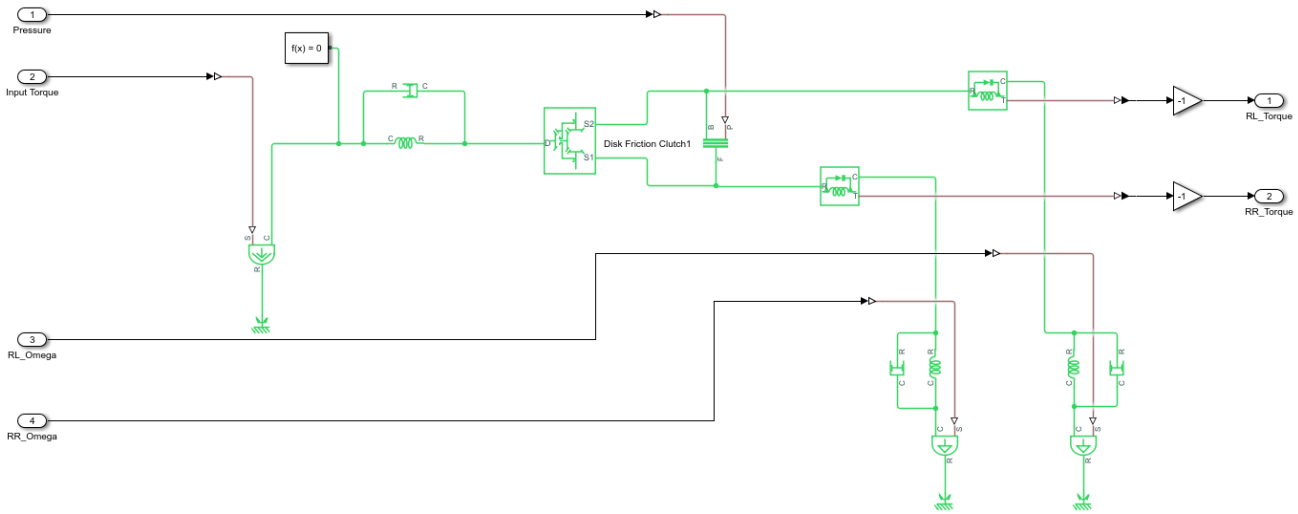


Figure 90. Screenshot of the differential model created with Simscape® Driveline.

Between the shafts and the torque generators, springs and rotational dampers, both of which are very stiff, were inserted in series, only with the intention of making the system mathematically stable and derivable, to 'help' the Simscape solver. The stiffnesses and damping values are very high and have negligible effects on the torque and rotational values of the differential. The differential used in this model has a drive ratio equal to 1, since the torque already present on the carrier is used as input. The data characterising the clutch pack is chosen arbitrarily. In fact, the input to the LSD will be the preload value and locking percentage, not the clutch or ramp parameters. It is therefore necessary to calculate before the pressure to be supplied to the clutch packs according to the required inputs. The pressure calculation is described in the equation 5.5.

$$Pressure = \frac{T_{Lock}}{2 \cdot r_e \cdot \mu_{disc} \cdot N_{disc} \cdot A_{piston}} \tag{5.5}$$

Where r_e is the effective radius of the friction discs, μ_{disc} is the coefficient of friction between the friction discs, N_{disc} is the number of discs and A_{piston} is the area of the pistons on which pressure acts. T_{Lock} is the differential locking torque and is defined in equation 7.6.

$$T_{Lock} = \begin{cases} C_1 \cdot |r \cdot T_{IN}| & if \quad C_1 \cdot |r \cdot T_{IN}| > C_0 \\ C_0 & if \quad C_1 \cdot |r \cdot T_{IN}| \leq C_0 \end{cases} \tag{5.6}$$

Where C_0 is the preload torque of the differential and C_1 is the locking percentage of t\he differential, which in reality depends on the inclination of the ramps and the clutch plates. C_1 can take on different values during acceleration or during release, depending on the ramps. therefore, the value of C_1 also depends on the sign of

the T_{IN}. A screenshot of the subsystem handling the pressure calculation in Simulink® is shown in the Figure 91.

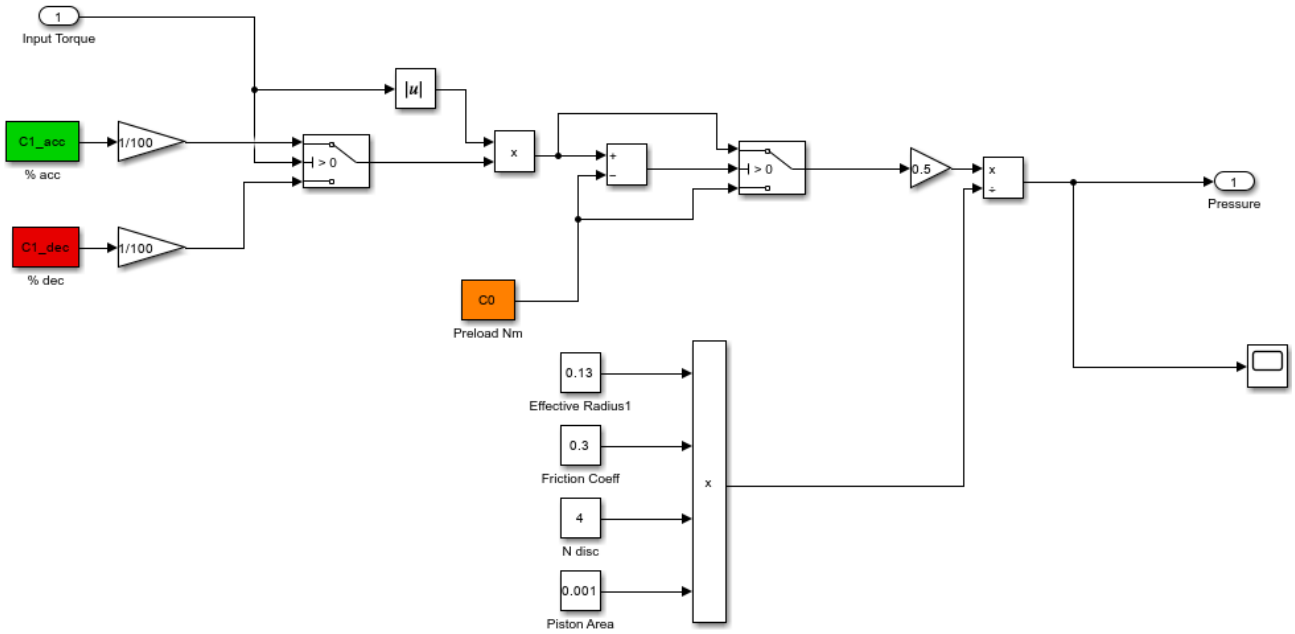


Figure 91. Screenshot of the subsytem handling the pressure calculation in Simulink®.

5.2.2 Results and comparison with ViCarRealTime®

The self-locking differential model developed proved particularly useful for the creation of active logic. By controlling the pressure pressing on the clutch plates, an active LSD can be created.

Furthermore, the model proved to be robust, and the stability problems observed with ViCarRealTime® built-in LSD differential were overcome. As an example, graphs are given below to show the differences between the LSD model created and that of ViCarRealTime®. The basic model, as explained above, often has instability problems due to friction modelling. Figure 92 shows the torques of the two differential semi-axles over time. The simulation represents a track lap on the Hockenheim circuit. An LSD model with a preload of 100Nm is used in both simulations, and a locking percentage of 65% in power and 25% in coasting.

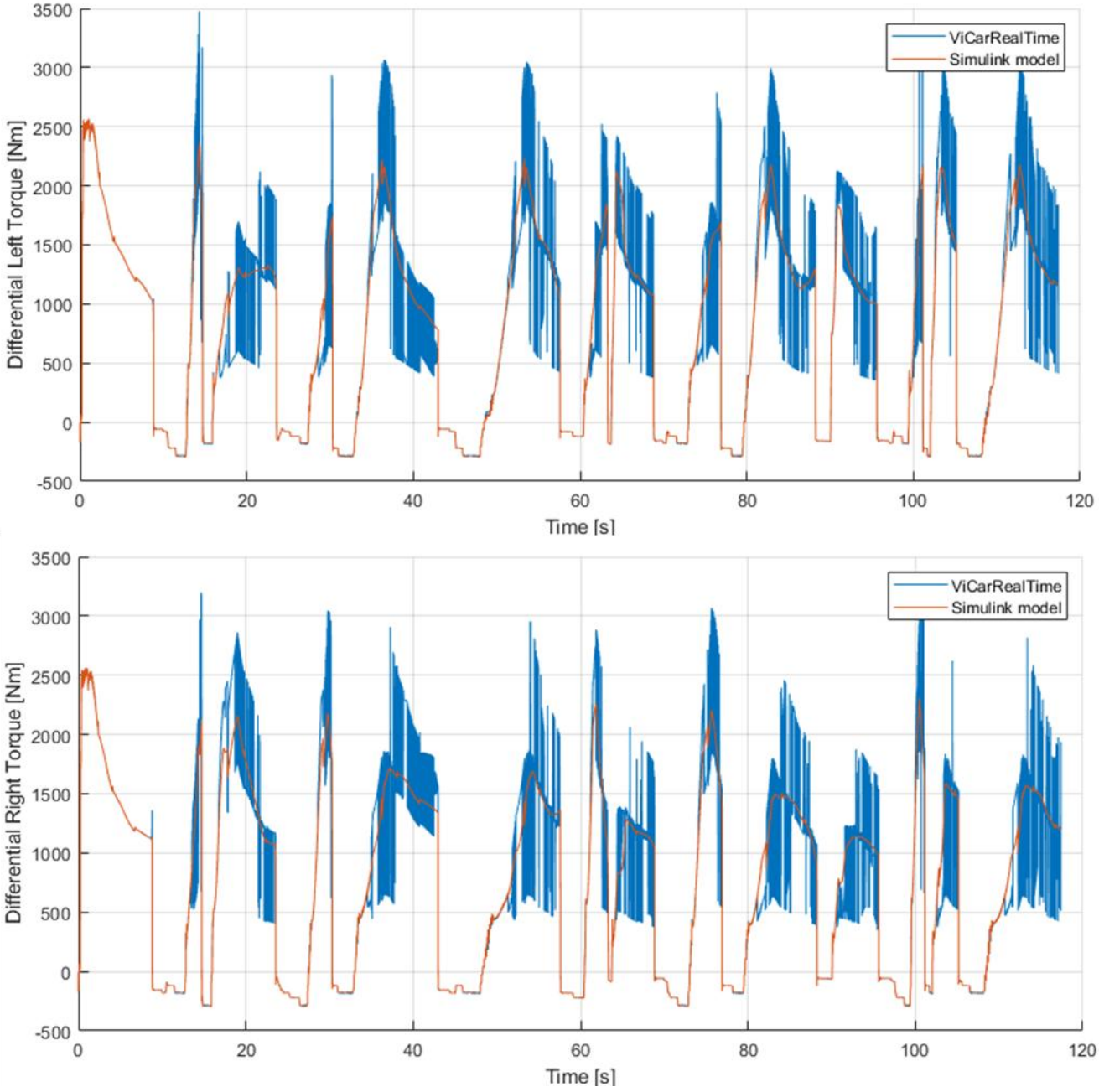


Figure 92. Torques on the axle shafts of the LSD diff in a simulated track lap. In blue the original ViCarRealTime® model, in orange the model created in Simulink®.

For high locking values, the original LSD vibrates, and torque continues to pass quickly from one axle to the other. The macroscopic effect of this phenomenon has no effect on the vehicle dynamics, as the average torque value remains correct, but these vibrations are annoying when the model is driven in the driving simulator. A further graph, often used to observe the behaviour of a self-locking differential, which shows the torque of one axle shaft as a function of the other, is shown in the Figure 93.

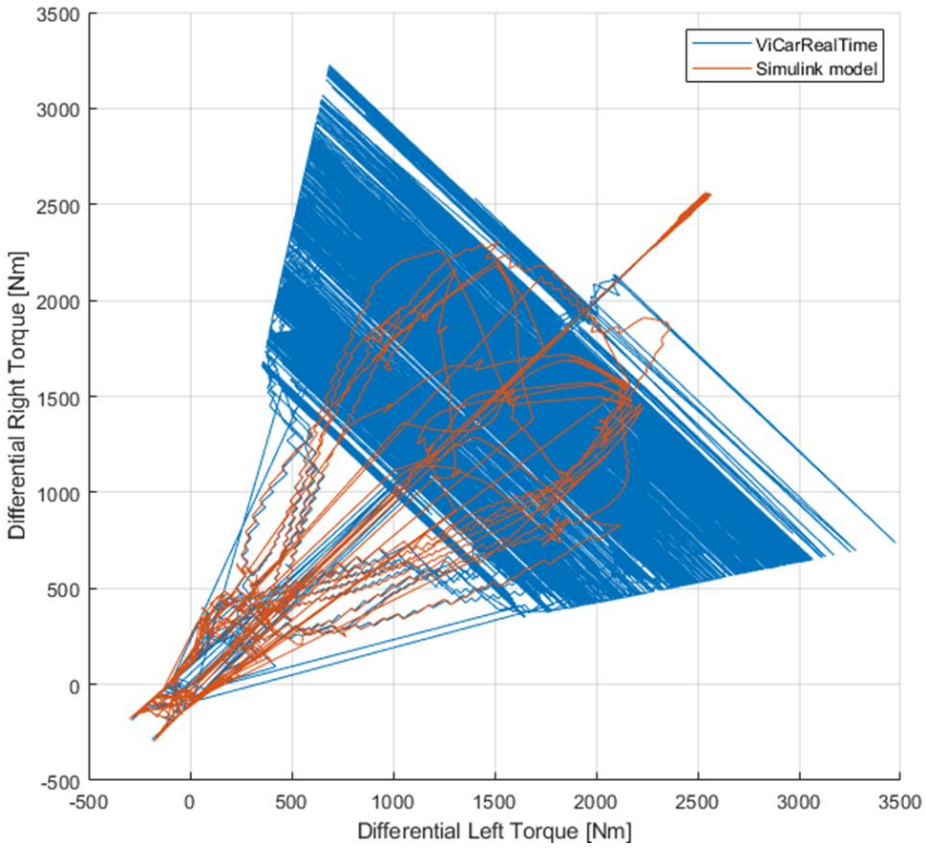


Figure 93. Left differential torque as a function of right differential torque. In blue the original ViCarRealTime® model, in orange the model created in Simulink®.

Also, in the Figure 94, one can see the instability of the differential and the continuous transition of torque from one semi-axis to the other. However, this effect does not manifest itself in the differential model created.

For completeness, the Figure instead shows the trends of the angular velocities of the wheels as a function of time.

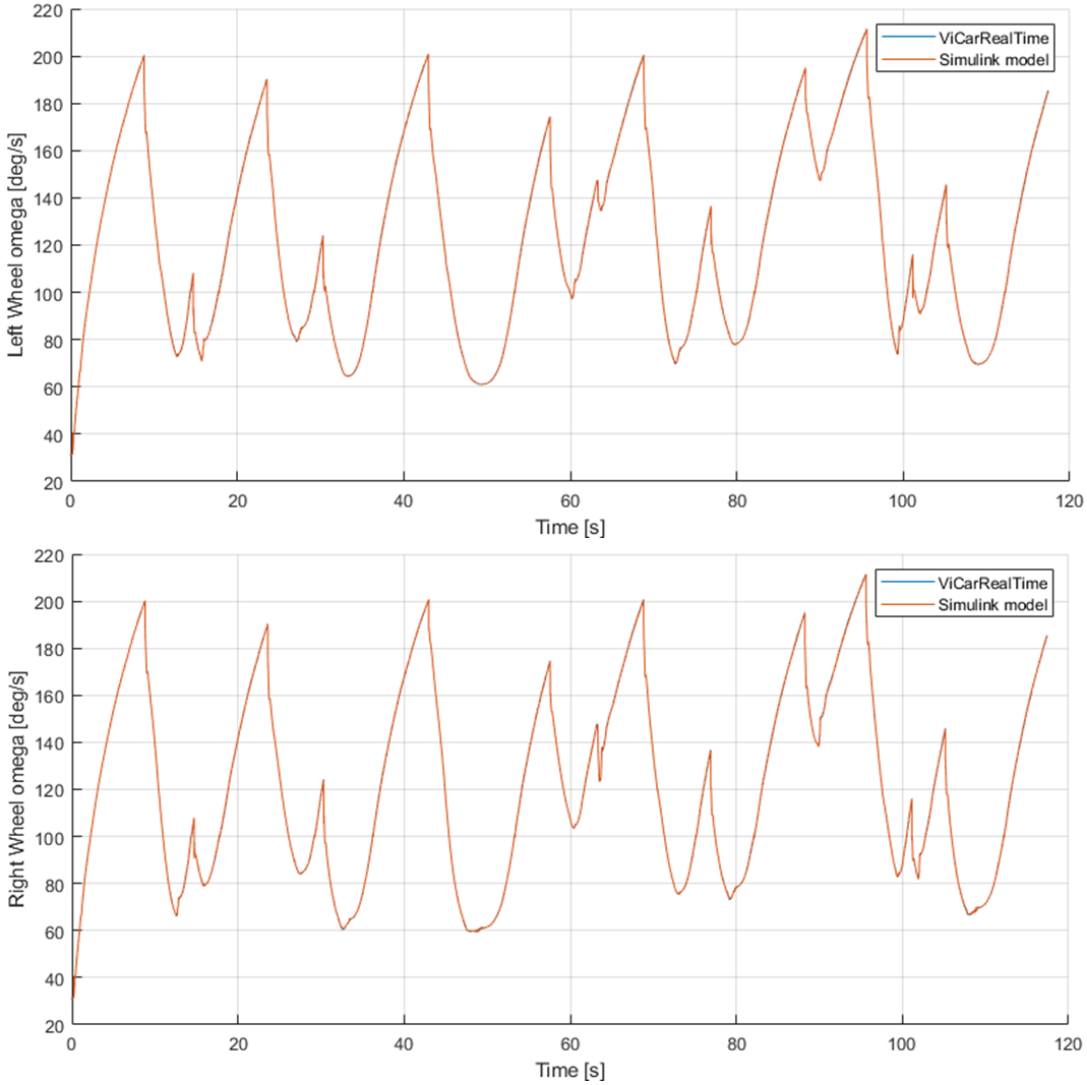


Figure 94. Wheel angular velocities as a function of time during a simulated lap. In blue the original ViCarRealTime® model, in orange the model created in Simulink®.

As far as the angular speeds of the wheels are concerned, there is no difference in the two models and the two curves are practically superimposed.

6 Exploring the Capabilities of the Driving Simulator

In this final chapter, which is somewhat more qualitative than the others, other parallel projects explored during the PhD are described. These projects investigate the various capabilities of the driving simulator to integrate with the vehicle and its components, as well as the opportunities that arise for their development.

In particular, the integration with hardware components such as dashboards and tactile pads is showed. Finally, Vi-WorldSim® is explored, which allows the development of ADAS logics in a fully virtual environment.

6.1 Tactile Pads

Another work carried out during the PhD course was the hardware and software implementation of Tactile Pads, shift paddles sensitive to finger touch. These paddles are also patented by the Automotive Engineering and Design research group of the University of Brescia.

The basic concept is to create a transition mode between manual and automatic mode shifting. If the paddles are not touched, the vehicle simply proceeds with the automatic transmission, however, from the moment one of the paddles is touched (without actuating it) it switches to manual mode and the automatic transmission logic stops working. When the paddles are released for longer than 500ms then the vehicle returns to automatic mode.

This control can be used, for example, to decide how to downshift gears when braking and then use the automatic transmission when accelerating. Or it is possible, for example, to upshift a gear (without the automatic transmission intervening) simply by keeping the hand resting on one of the two paddles.

Figure 95. Tactile Pads mounted on the driving simulator.

The paddles are made of carbon fibre, an LED and a vibrodine is mounted on top of them, and a capacitive sensor has been linked to the carbon fibre, which is conductive, to detect the actual touch on the paddle. Shown in the Figure 95 are photos of the paddles mounted on the steering wheel of the driving simulator.

The LED lights up and stays on while the paddle is being touched, the vibrodine on the other hand vibrates for a few milliseconds to give haptic feedback and let the driver know that contact has been made.

6.1.1 Hardware and Software Implementation

Vibrodine feedback, LED switching and capacitive sensor reading are handled by a specially programmed Arduino® controller. The Arduino also sends a digital signal to the simulator, signalling whether or not the paddles are touched. In the Figure 96 it can see the microcontroller inserted in a special 3D printed box with the appropriate connectors.

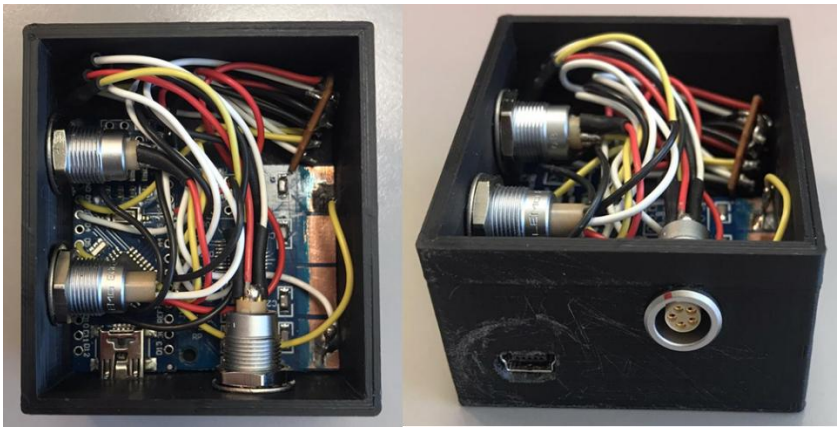


Figure 96. 3D printed box containing the Arduino microcontroller, capacitive sensor, and connectors to operate the Tactile Pads.

The driving simulator is equipped with an acquisition board capable of reading both digital and analogue inputs. In this case, two digital inputs are used to signal the contact between a finger and the right and left paddle respectively.

Software management was instead developed using Simulink®. ViCarRealTime®'s vehicle models do not allow switching between automatic and manual gearboxes during simulation, so it was necessary to create an externally managed gearbox model on Simulink®.

To manage the gearbox externally, it is first necessary to deactivate the gearbox model computation of ViCarRealTime and interface with the solver with appropriate inputs and outputs. Below is a list of the inputs that the solver needs to manage the gearbox externally:

- InternalGearboxActivity, to activate or deactivate internal gearbox computation.
- ClutchOutputplateAngle, angle of rotation of the gearbox input shaft connected to the clutch.
- ClutchOutputplateOmega, rotational speed of the gearbox input shaft connected to the clutch.
- ClutchOutputplateAlpha, rotational acceleration of the gearbox input shaft connected to the clutch.
- DiffCenterinputshaftTorque, torque of the gearbox output shaft, connected to the differential.

Below is the list of outputs to be taken from the solver and entered into the logic to manage the vehicle's gearbox:

- GearDemand, gear desired by the driver.
- DiffCenterInputAngle, rotation angle of the gearbox output shaft, connected to the differential.
- DiffCenterInputOmega, rotational speed of the gearbox output shaft, connected to the differential
- DiffCenterInputAlpha, rotation acceleration of the gearbox output shaft, connected to the differential
- ClutchOutputTorque, torque of the gearbox input shaft connected to the clutch.

In the Figure 97 it is possible to see a screenshot of the Simulink that manages the gearbox, the different ratios are managed by a lookup-table.

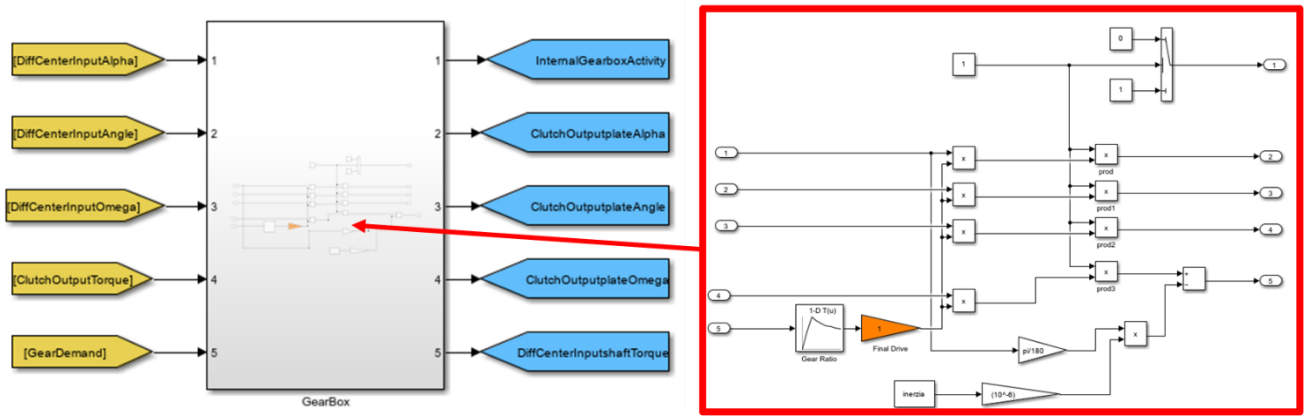


Figure 97. Screenshot of the Simulink® that manages the gearbox.

The logic to manage the transition between automatic and manual gearbox and vice versa takes the logic signals from the microcontroller installed on the steering wheel and processes them in a NOR logic port. In addition, a block (red box in Figure 97) was created to manage the transition from manual to automatic in a special way, so that the hands have to be detached from the paddles for a certain time value in milliseconds. The transition from automatic to manual, on the other hand, occurs instantaneously as soon as either hand makes contact with the paddles.

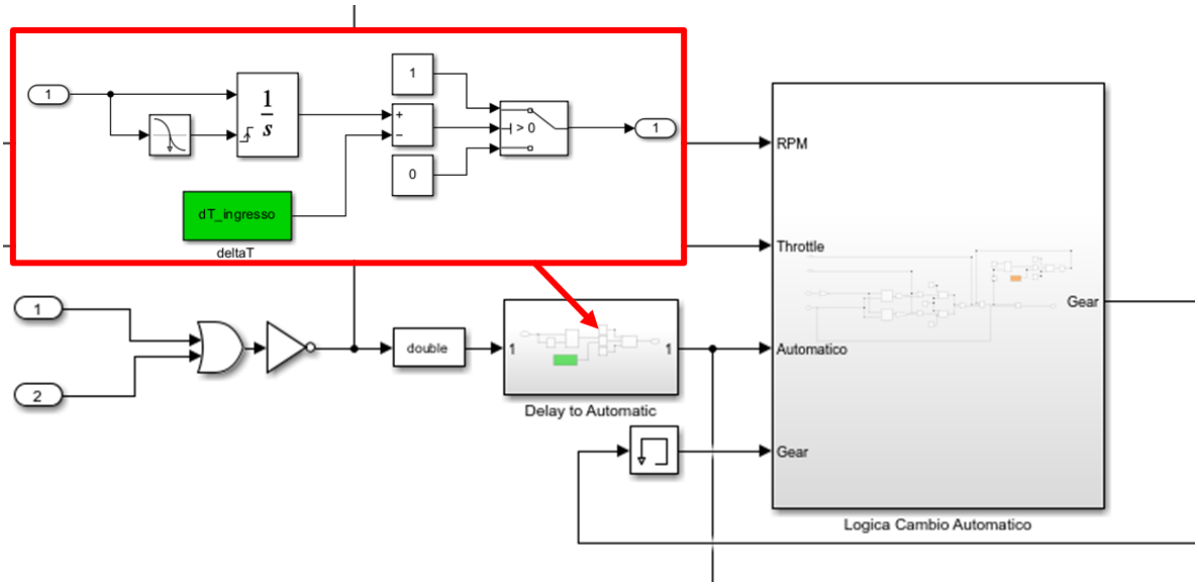


Figure 98. Simulink® screen showing the logic part for managing the transition between manual and automatic gearbox.

The logic of the automatic gearbox, which can be seen in the subsystem on the right-hand side of the Figure 98, is represented in its entirety in Figure 99.

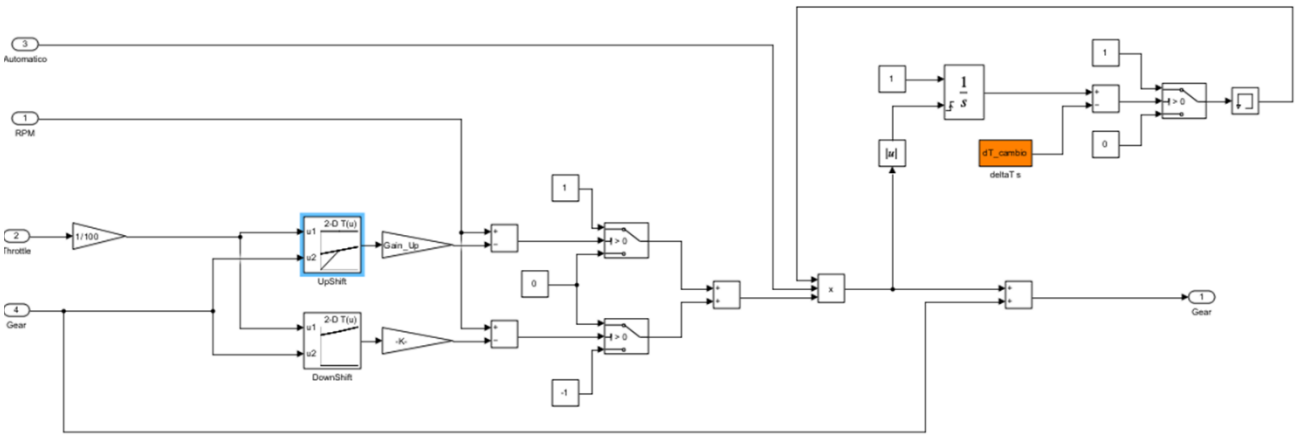


Figure 99. Simulink screen representing the management logic of the automatic gearbox.

Shift management is handled by two two-dimensional lookup-tables dependent on RPM and throttle percentage. The two lookup tables manage downshift and upshift separately. There is also a further control logic, which can be seen at the top right of the Figure 99, which prevents the logic from making upshift/downshift that are too close together; it is possible to vary the minimum number of milliseconds that must elapse between one gearbox and the next. Finally, logic was also created to avoid going over the RPM limiter when downshifting. This is done by simply calculating the RPM the engine would have if the gear is downshifted; if these exceed a threshold value, the shift is not accepted. Obviously, this logic works in the case of using manual gearbox mode.

6.1.2 Dashboard

A dashboard was also created, on a screen on the driving simulator in front of the driver, to display certain gearbox variables during the use of Tactile Pads. The dashboard consists of a screen like the one shown in the Figure 100.

Figure 100. Dashboard screen dedicated to the use of Tactile Pads on the driving simulator.

In the dashboard, vertical bars on the left show the accelerator and brake values. On the right is the steering wheel angle, and in the centre the engine RPM and vehicle speed. The horizontal bar at the top visually shows the engine RPM value scaled appropriately to the limiter value, the bar changes colour when approaching the threshold value, turning first orange and then red, indicating to the driver that it is necessary to change gearbox. In addition, the engaged gear is displayed by the letter M or A, respectively, to indicate if gearbox mode is manual or automatic. Finally, the two-coloured squares, normally red, turn green when the right or left paddle is touched respectively.

The dashboard has been created created using Microsoft® VisualStudio software using the VB.Net language. Figure 101 shows a screenshot of the VisualStudio software with part of the dashboard project.

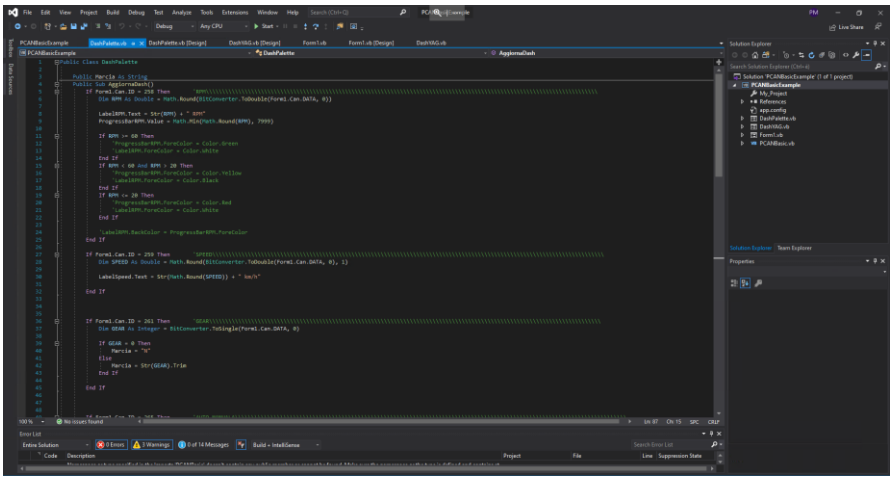


Figure 101. Screenshot of the Microsoft® VisualStudio software with part of the code for the dashboard.

To make the simulator communicate with the PC managing the dashboard, the CAN bus protocol was chosen. CAN bus is an asynchronous communication protocol used extensively in the automotive field and allows very efficient communication by being able to send digital signals up to 64bit in size.

The driving simulator is equipped with a programmable CAN board, so it was possible to choose which simulation channels to transmit. The PEAK-System's CAN-USB adaptor such as the one in the Figure 102 was used to make the PC that runs the dashboard read the channels coming from the CAN bus. This adapter makes it possible to translate the CAN signal into a serial signal that can be easily read by the PC.

Figure 102. CAN-USB adapter from PEAK-System.

6.2 ADAS and ViWorldSim®

This work started with a research project with the company Adas Mobile, which offers services for the recalibration of ADAS sensors, for example after an accident. The aim of this work is not strictly scientific, but it allowed the author to explore the potential of ADAS systems through the driving simulator.

The author's request was to be able to create some situations, on the driving simulator, in which users could have the opportunity to drive a vehicle equipped with ADAS under normal conditions and subsequently under conditions in which the ADAS sensor was misaligned (see Figure 103), in order to be able to make people understand the importance of having correctly calibrated radars and cameras.

Figure 103. Vehicle with virtual camera misaligned by 3°.

Part of the simulator hardware was purchased by Adas Mobile, this equipment was then taken to the Autopromotec® trade fair in Bologna, where guests had the opportunity to test the vehicle equipped with calibrated and uncalibrated ADAS under the supervision of the author. Shown in the Figure 104 is the driving simulator set up at the fair.

Figure 104. Simulator set up on the Adas Mobile stand at the Autopromotec trade fair in Bologna.

As a global public health issue, road safety has always been a subject of debate. Statistics from the World Health Organisation show that in 2016 there were 1.35 million deaths due to road accidents. More than 90 per cent of accidents are attributable to the human factor and 40 per cent of them are caused by distracted driving and high speed. With a view to reducing accidents attributable to human error of judgement, ADAS were introduced and developed and, as technology developed and sensors became cheaper, they became indispensable devices in a vehicle. ADAS (advanced driver assistance systems) are a group of electronic technologies that assist the driver in driving and parking manoeuvres. Through a safe human-machine interface, ADAS increase vehicle and road safety. They use automated technologies such as sensors and cameras to detect nearby obstacles or driver errors and respond or send alarms accordingly.

For this work in particular, a virtual vehicle model equipped with three ADAS systems was used:

- LKA (Lane Keeping Assistance)
- ACC (Adaptive Cruise Control)
- AEB (Active Emergency braking)

The LKA is able to keep the vehicle on track by autonomously controlling the steering wheel and providing a vibration on it to alert the driver. ACC is a cruise control, i.e. a controller that manages the speed of the vehicle but is able to detect vehicles ahead and adjust speed according to them to maintain a safe distance. AEB, on the other hand, is a system that detects possible collisions and prevents them by promptly acting on the braking system. The AEB is generally capable of detecting pedestrians, vehicles, or others, whether they are ahead of the vehicle or coming from the side.

The logics used were not developed from scratch but started from ADAS logics that are provided as an example in the ViWorldSim® software.

6.2.1 ViWorldSim®

Figure 105. ViWorldSim® software.

ViWorldSim® (Figure 105) is a software produced by Vi-Grade, the same company that produces ViCarRealTime® and the simulator at the University of Brescia.

ViWorldSim® allows the creation of scenarios in which different vehicles with autonomous driving capabilities can be placed and traffic situations can be created. If desired, it is also possible to make vehicles perform certain manoeuvres, such as pulling out of a stop sign or crossing an intersection, the same can be done for pedestrians or cyclists. Routines can be created and executed, for example, when the vehicle we are driving passes through a virtual 'portal'. All this, together with very realistic graphics, provides an environment for testing and developing ADAS systems in a completely virtual manner. There are different environments, there is a 3D scan of a city section of the city of Wolfsburg, motorway sections with also entrance and exit

ramps, or simpler scenarios representing residential districts. Shown in Figure 106 is a screenshot of the ViWorldSim® tool for creating scenarios.

Figure 106. Screenshot of the ViWorldSim® tool for creating scenarios.

ViWorldSim® integrates with ViCarRealTime®, so it is possible to drive a vehicle model created with ViCarRealTime® in the ViWorldSim® environment. The Simulink® environment is used to run simulations with ViWorldSim®, both for offline and online simulations (using the simulator).

Thanks to Simulink®, it is therefore possible to manage the logics in a comprehensive manner.

There are blocks that allow to manage virtual sensors. In particular, used in this work are Virtual Cameras, which are able to read the environment in the ViWorldSim® scenario and provide synthetic outputs with which ADAS logic can be constructed. A matrix of points indicating the presence of objects in front of the camera is used, through which it is possible to recognise the presence of any obstacles and their relative speed with respect to the vehicle. In addition, the virtual cameras can read the stripes on the asphalt and provide as output a polynomial function describing them, also to consider possible curvatures. A screenshot of the Simulink® block that manage virtual camera is shown in the Figure 107.

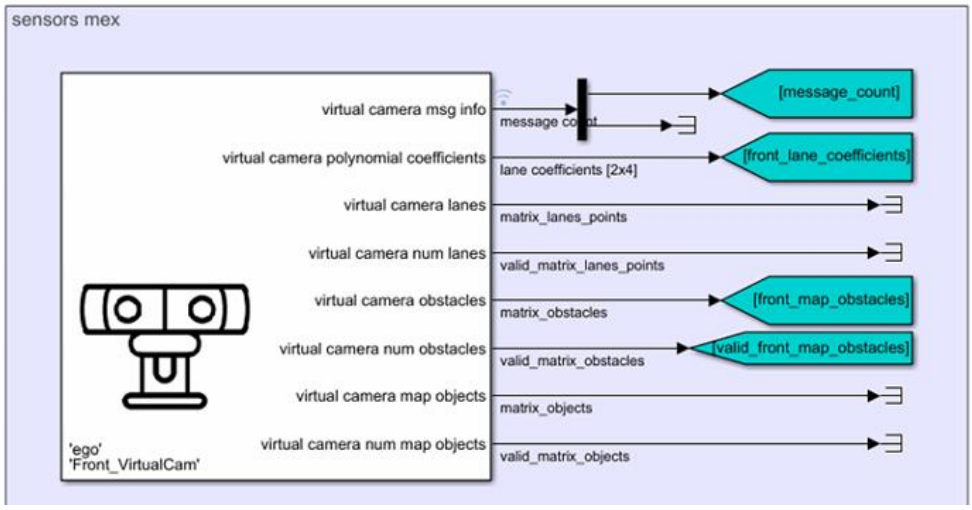


Figure 107. Screenshot of the Simulink blocks operating the virtual camera.

6.2.2 Work Description

About the ADAS logics used, ViWorldSim® offers a few already functioning examples; as the purpose of this work is purely demonstrative, no major developments have been made on these logics, apart from a few parameter adjustments. The point just made applies mainly to ACC and AEB.

The LKA logic provided by default was instead developed from scratch, as the one provided by ViWorldSim® is designed to work with the Vi-Grade simulator steering wheel, which allows it to be position controlled. The Adas Mobile simulator station was instead set up with another steering wheel, a Fanatec DD1®. The motor of this steering wheel does not allow it to be controlled directly in position, but only through the torque. It was therefore necessary to create a logic to manage lane keeping by controlling the steering wheel torque.

As mentioned earlier, one of the outputs of the virtual camera are two fifth-degree polynomial functions capable of fitting road stripes. This makes it possible to calculate the distance of the centre of the car from the two stripes. The calculation of the distance is not done at the point where the car is, but at a point that is 10 metres in front, so that an eventual curve can somehow be predicted. Obviously, as the purpose of this work is demonstration, we have not gone any further in predicting curve trajectories.

The logic created takes as input these two distances, i.e. the distances of the road stripes, from the right and left side. As output, a steering wheel torque is generated as a function of several parameters:

- Vehicle Speed, torque applied to the steering wheel is directly proportional to speed.
- Stripe distance, the torque applied to the steering wheel is quadratically proportional to the distance from the strip.
- Stripe approaching speed

The stripe approaching speed is the time-derivative of the stripe distance. Steering wheel torque is applied more or less quickly depending on speed and to make this, transfer functions with a negative pole has been used. In this way, the control action can be more aggressive or smoother.

Logic always takes the nearest strip into account and torque is delivered according to that. The model features several gains so that the logic can be calibrated in real-time via the driving simulator. Some tests in the driving simulator revealed some criticalities. For example, when a line is overtaken by the vehicle, the steering wheel torque instantly reverses, creating an unpleasant effect on the driver. For this reason, an additional control was induced to understand when the vehicle is moving away from the nearest strip, thus when it passes it, in order to deactivate the control logic.

Finally, when the vehicle is too close to the line, a low-amplitude sine wave is generated to make the driver feel a vibration on the steering wheel. A screenshot of the Simulink® handling the LKA logic just described is shown in the Figure 108.

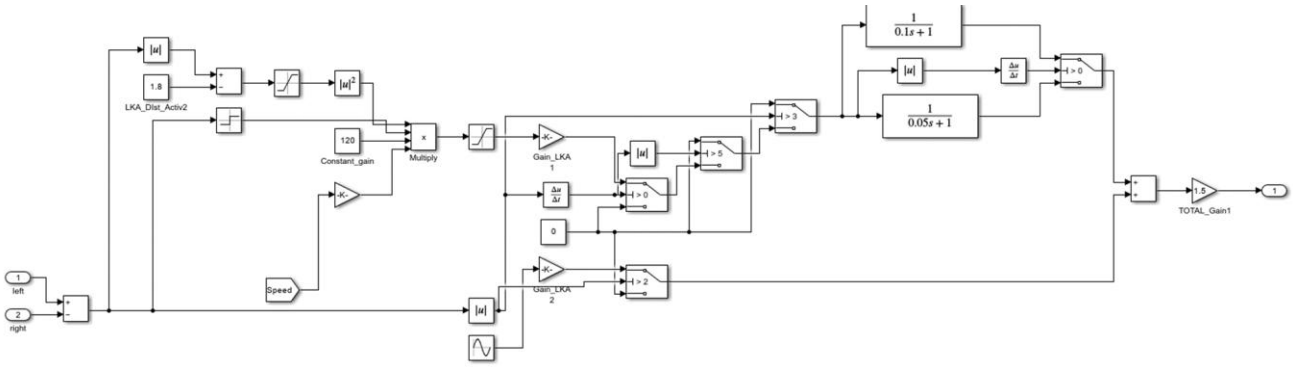


Figure 108. Screenshot of the Simulink® file that handles the LKA logic by controlling the steering wheel torque.

Two scenarios were therefore created for the demonstration. One set in a random traffic scenario in Wolfsburg and one set in a particularly busy motorway section. The Figure 109 represents the Wolfsburg scenario. The Figure 110 represents the motorway scenario.

Figure 109. Wolfsburg scenario created with ViWorldSim®.

Figure 110. Motorway scenario created with ViWorldSim®.

Each of the two scenarios was driven first with the camera correctly calibrated, then with the camera misaligned by 3 degrees. The cameras can be positioned and oriented in the scenario tool changing their yaw value.

In addition to the screen showing the pilot's view, the simulator setup has been designed with three other screens. Two screens were positioned to the side of the driving position so that other people could watch the simulation. The first screen was simply a view from outside the vehicle. The second screen displayed, via the ROS software, some of the parameters of the simulation and in particular what the virtual camera was able to pick up, show any obstacles and the stripes on the asphalt. This screen was particularly useful for visualising the effect that sensor misalignment had on the camera's output. What has just been said is clearly visible in Figure 111 and Figure 112, where the 4 screens displayed during the simulation are shown respectively, the first corresponding to a scenario with the camera correctly calibrated, and the second with the sensor misaligned by 3 degrees.

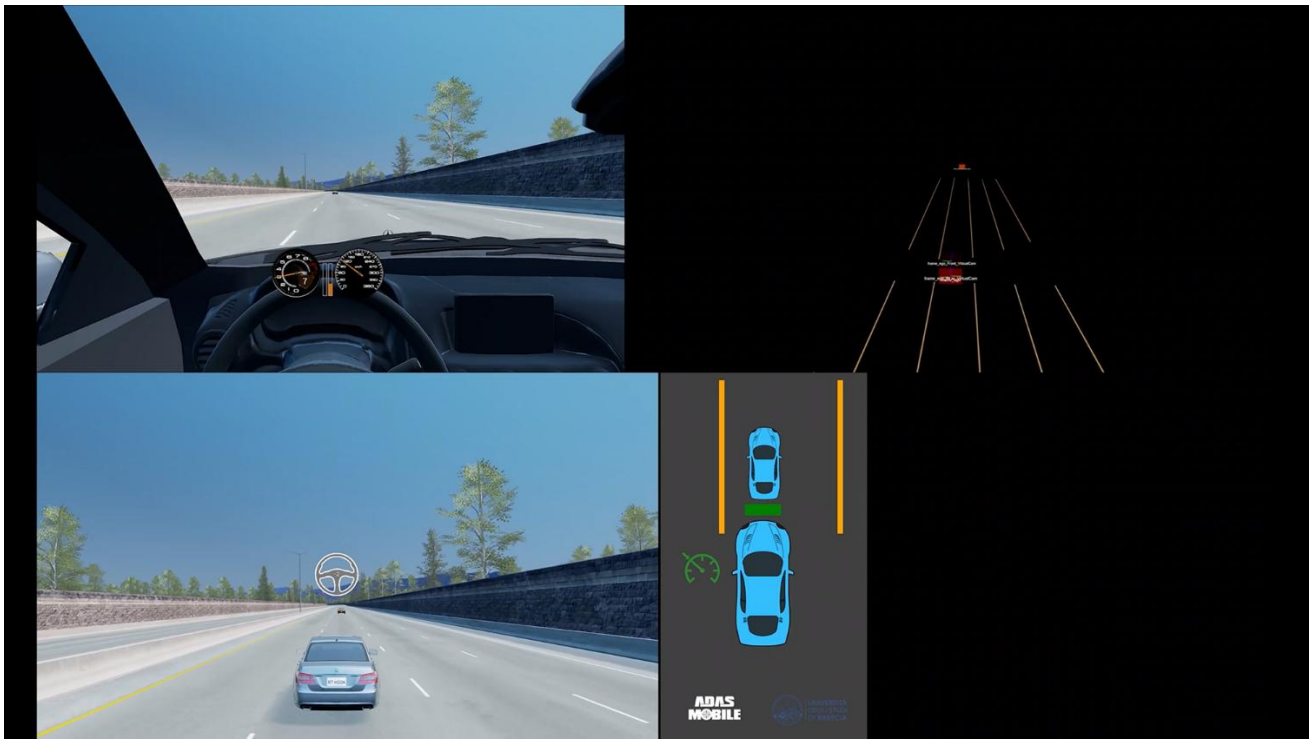


Figure 111. Screenshots of the 4 screens displayed during the simulation with the sensor correctly aligned. Top left the pilot's view. Bottom left the view from the outside. Top right the view from ROS with the output of the virtual camera. Bottom right the dashboard.

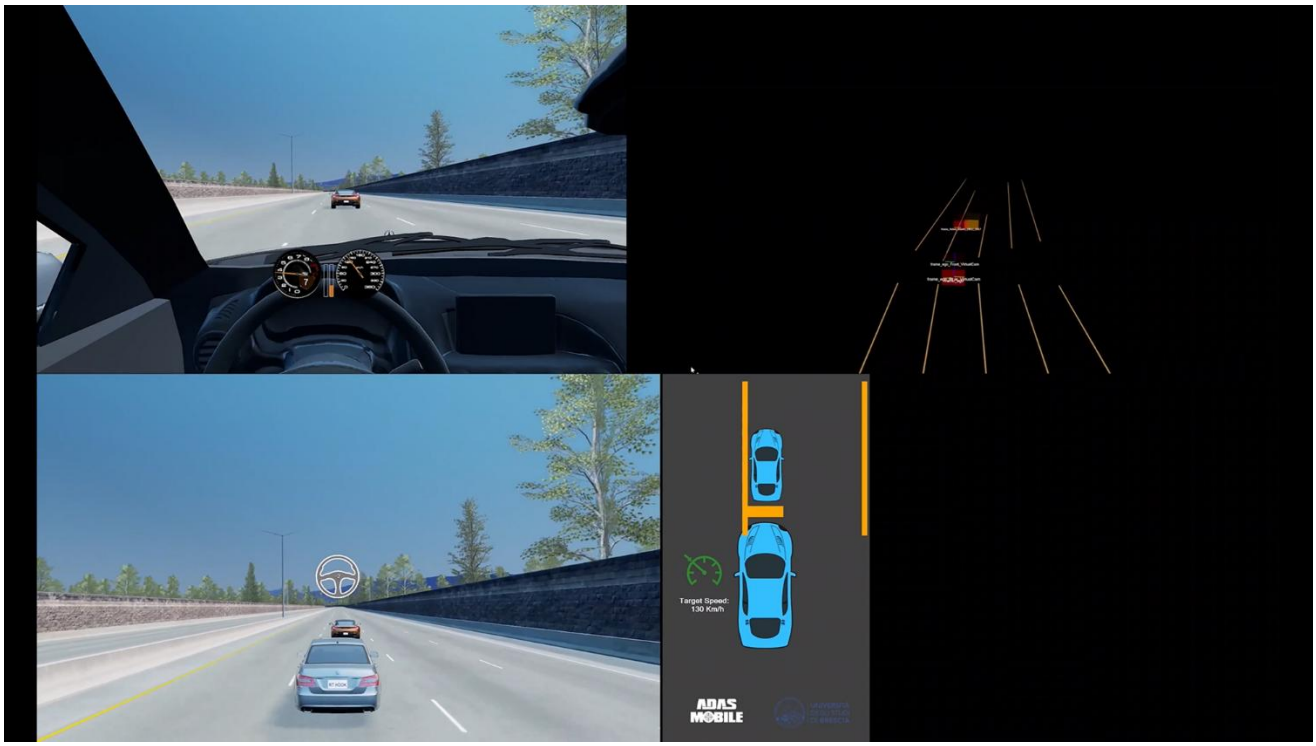


Figure 112. Screenshots of the 4 screens displayed during the simulation with the sensor misaligned. Top left the pilot's view. Bottom left the view from the outside. Top right the view from ROS with the output of the virtual camera. Bottom right the dashboard.

The last screen was placed on the driver's seat, visible to the pilot, as in the Figure 104, is a dashboard showing some parameters of the ADAS logics. This dashboard was developed in the VB.Net environment and

communicates with the simulator via the CAN protocol, the same as described in the chapter 4.2. An example layout of the dashboard is shown in the Figure 113.

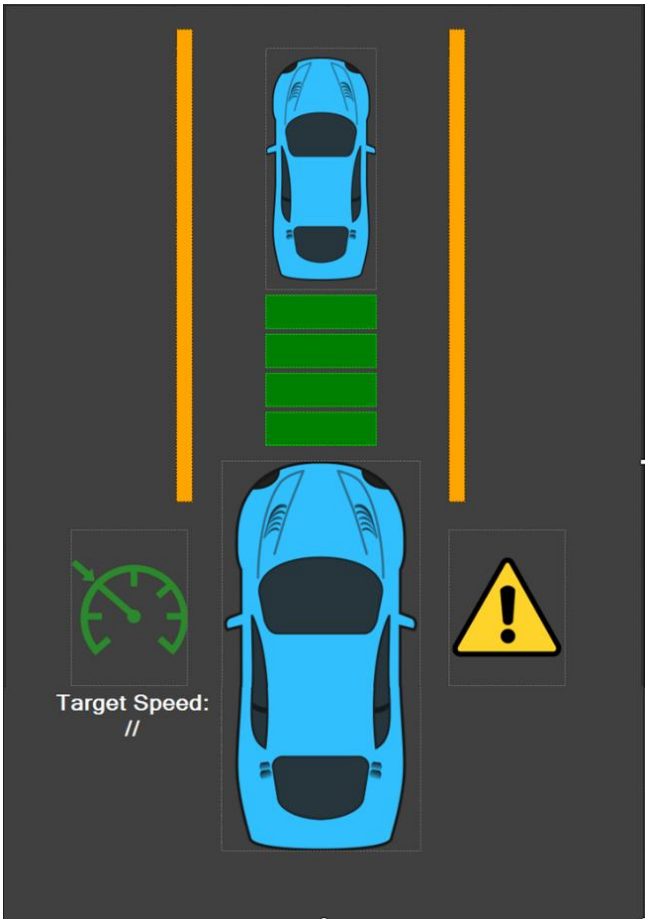


Figure 113. Dashboard layout to display ADAS logic parameters.

The dashboard shows some parameters related to ACC, in particular bars are shown between the two vehicles representing four different possible distances that the ACC logic must maintain from the vehicle in front. By means of a button mapped on the steering wheel of the simulator, it is possible to change the distance. When the bars are green, it means that the ACC logic has detected the vehicle in front and is trying to maintain the set distance. If the bars turn yellow, the vehicle is not detected, and the logic behaves like normal cruise control also shows the target speed in the dashboard. As far as the logic of LKA is concerned, two vertical lines representing the road stripes are displayed, the lines move in real-time during the simulation. While when AEB is activated, a warning signal appears on the dashboard.

7 Conclusion and further developments

The driving simulator proved to be a very powerful tool as it allows one to go beyond vehicle dynamics simulations, being able to evaluate subjective aspects and driver sensations. It also allows complex manoeuvres to be performed faster, though certainly less repeatable.

Several methodologies for system integration through driving simulator through driving simulator has been explored in this thesis. Most of the work was done trying to integrate algorithms for the modelling of suspensions, focusing on the elasto-kinematics aspects. The models were also validated using other software such as Shark®, for the calculation of constraint reactions, and Simscape® Multibody. These algorithms can be used both for the creation of subsystems of the virtual vehicle model and for the design of the vehicle itself; indeed, one of the future developments is the integration of these mathematical models into the 3DSusp software, giving the possibility to use these calculation models in a user-friendly manner. In future, 3DSusp will be equipped with a module for calculating loads and a module for calculating suspension compliance, giving the user the option of including bushings and its characteristics. Other future developments for 3DSusp are the inclusion of the two axles simultaneously to simulate more complex vehicle movements such as pitch, roll and combinations of both.

The complete model described in 3.3 proved particularly powerful for the purpose of assessing the non-linear characteristics of a suspension. Reviewing the literature, it was found to be one of the most comprehensive, since it takes into account mainly nonlinear stiffnesses and the possibility of using wheel-side bushings as well. In addition, the calculation is done iteratively and takes into account the kinematics of the suspension, unlike many works that simply calculate the stiffness matrix of the system, which is practically useless when large deformations occur. Considering stiffnesses as linear can only be a sensible approximation for small loads and small displacements.

The mathematical model is capable not only of calculating the deformed configuration of the suspension, but also of returning the constrained reactions on each bushing of the suspension. This is useful for suspension design process as an input to FEM models. Compared with a model created using multibody software, the computation time is essentially negligible; code written in Matlab® takes one or two tenths of a second to converge to the solution. The model has also proven to be particularly useful just to perform kinematic analysis of the suspension by appropriately modifying the stiffness of the bushings and elements.

Once it is understood how to write the equations depending on how the various elements are linked together, any type of independent-wheel suspension can be created. This makes this model particularly powerful in developing the elastokinematics of suspension quickly. In fact, the force-displacement curves that characterize bushings and elements can also be changed easily as a key feature in the design phase. The presented methodology proved to be particularly effective, especially for the possibility of handling the non-linear force-deformation curves of the joints. Most models in the literature do not consider these effects, but they make a

substantial difference. Designing suspension compliance correctly can have macroscopic effects, both on driving feel and vehicle balance, with implications for safety and performance as well.

It can be stated that this algorithm represents a kind of virtual K&C. Having a validated model can lead, at an early stage of the design, to saving time and money spent at a test bench. In addition, several combinations of forces and moments can be tested through simulations that are difficult to achieve with a K&C test rig.

As this method has a high computing speed, it can be combined with optimisation or sensitivity analysis algorithms to speed up the design process as much as possible. For example, having a function written in Matlab®, it is possible to use the proprietary 'Optimisation Toolbox' with relative ease. Future developments could include the use of optimization algorithms such as those based on Levenberg-Marquardt or genetic algorithms.

The algorithm is intended to be a substitute for a multibody solver, but only in an initial design phase. In fact, this method is based on the search for the equilibrium position, it is therefore a quasi-static solver where dynamic effects such as inertia of the suspension elements or damping of the joints are not considered. Thus, a limitation of this algorithm is precisely the dynamic analyses, such as the evaluation of eigenfrequencies. The inclusion of these effects can be a future development, for example by including forces dependent on the time derivatives of the element's deformations.

The method is designed to solve any type of independent wheel suspension. However, there are no elements connecting the wheels of the same axle together, such as anti-roll bars, which would in any case influence the compliance of the suspension during cornering. A future development will be the introduction of these elements into the algorithm. Finally, it is not possible to model suspensions with linkage and rockers, such as pushrods and pullrods, even though they are rather rare architectures in passenger cars. Much work can still be done on the real time performance of the elasto-kinematic model, which with the hardware in possession can be run at no more than 500Hz for now, but with optimisations and adjustments it is possible to go up in frequency.

Methodologies for the integration of sub-systems were presented; this was made possible by the vehicle dynamics simulation software ViCarRealTime®, which allows co-simulation with the Simulink® environment, where the development possibilities are almost endless. Particularly about the creation of active logic, for example for suspensions or a lockable differential.

Furthermore, the driving simulator, which allows the so-called driver in the loop, is an essential tool for the development of a vehicle and its subsystems also thanks to the possibility of interfacing it with different hardware, as seen in the chapter 6.1. Thanks to the management of a CAN bus, it is possible to make any type of sensor/actuator communicate during simulation. It also makes it possible to create customised dashboards, which are very useful from a human-machine interface point of view.

This thesis also proposed a comparison of two main methods for estimating the vehicle side sleep angle, one of the fundamental parameters for understanding vehicle behaviour: Kalman filter and artificial neural network

(ANN). All of them have positive aspects and drawbacks: the ANN method can give the best overall estimation with a low computational burden, but it needs a very demanding training procedure, moreover it is not able to deal with changes of the system and boundary conditions that were not included in the training. The EKF itself can give a good VSA estimation as well but the computational burden increases with the accuracy of the vehicle model, which requires in turn many parameters (i.e. tire data) that might be difficult to get. This makes it difficult to use in key applications such as the typical model-based design approach required by real time operations in state-of-the-art driving simulators.

As far as ANNs are concerned, this has only scratched the surface; in fact, one of the future developments is precisely the development of much more complex neural networks for estimating the side slip angle. Focusing on neural networks that also take into account the input and output values of previous time steps. The development of these more advanced logics may also give the possibility of creating more reliable control logics, such as torque vectoring, electronic stability control or rear wheel steering.

Finally, an exploration of the ViWorldSim® environment is presented in this work, thanks to a research project with Adas Mobile®. It shows that it is possible via the driving simulator to integrate ADAS logic. This exploration was only superficial, as the aim of the work was demonstrative; the possibilities this software offers are still to be explored. In any case, the research project ended with excellent results, hundreds of people at the fair were able to drive the simulator by trying out some simple ADAS logics both with the sensors aligned and not. Everyone was able to appreciate the difference, and the importance of calibrating these systems, which was the goal of this work.

List of Figures

References

[1] C. Himmels, J. Venrooij, A. Parduzi, M. Peller, and A. Riener, "The bigger the better? Investigating the effects of driving simulator fidelity on driving behavior and perception," *Transp Res Part F Traffic Psychol Behav*, vol. 101, pp. 250–266, Feb. 2024, doi: 10.1016/J.TRF.2024.01.007.

[2] "Within- and Between-Subject Designs in DrivingSimulator Validation: A Case Study - Driving Simulation Proceedings." Accessed: Nov. 28, 2024. [Online]. Available: https://proceedings.driving-simulation.org/proceeding/dsc-2023/within-and-between-subject-designs-in-drivingsimulator-validation-a-case-study/

[3] A. Soltani and F. Assadian, "A Hardware-in-the-Loop Facility for Integrated Vehicle Dynamics Control System Design and Validation," *IFAC-PapersOnLine*, vol. 49, no. 21, pp. 32–38, 2016, doi: 10.1016/J.IFACOL.2016.10.507.

[4] G. Lukianov, "Vehicle Steering and Suspension Kinematics/Compliance and Their Relationship to Vehicle Performance©," *CISM International Centre for Mechanical Sciences, Courses and Lectures*, vol. 603, pp. 61–138, 2022, doi: 10.1007/978-3-030-75884-4_2.

[5] M. Von Der Ohe, "Front and rear suspension of the new mercedes model W201," *SAE Technical Papers*, 1983, doi: 10.4271/831045.

[6] J. E. Andersson, O. Bane, and A. Larsson, "Volvo 760 GLE multi-link rear axle suspension," *SAE Technical Papers*, 1989, doi: 10.4271/890082.

[7] T. Kasahara, N. Morioka, M. Satou, T. Yagi, and K. Muraoka, "Development of a New Multi-link Suspension," *SAE Technical Papers*, Mar. 2000, doi: 10.4271/2000-01-0092.

[8] R. Koide, Y. Kawabe, K. Nakajima, and K. Kazuhiro, "Development of a New Multi-link Rear Suspension," *SAE International Journal of Passenger Cars - Mechanical Systems*, vol. 5, no. 2, pp. 974–980, 2012, doi: 10.4271/2012-01-0978.

[9] D. Jung, J. K. Lee, B. K. Lee, and S. P. Kim, "The Development of FR-Based 4WD Multi-Link Suspension," *SAE Technical Papers*, vol. 2013-April, 2013, doi: 10.4271/2013-01-1235.

[10] P. Zandbergen and A. Girelli Consolaro, "Ford motor company's new rear suspension architecture for the global CD platform," *Lecture Notes in Electrical Engineering*, vol. 198 LNEE, no. VOL. 10, pp. 9–20, 2013, doi: 10.1007/978-3-642-33795-6_2/FIGURES/11.

[11] G. Legnani, Irene. Fassi, and Antonio. Visioli, "Robotica industriale : modellazione, pianificazione, controllo, programmazione, componentistica, normativa e sicurezza," 2018.

[12] W. A. Cao, H. Ding, and D. Yang, “A method for compliance modeling of five degree-of- freedom overconstrained parallel robotic mechanisms with 3T2R output motion,” *J Mech Robot*, vol. 9, no. 1, Feb. 2017, doi: 10.1115/1.4035270/473080.

[13] S. Huang and J. M. Schimmels, “Planar Compliance Realization with Two 3-Joint Serial Mechanisms,” *J Mech Robot*, vol. 14, no. 5, Oct. 2022, doi: 10.1115/1.4053284/1129388.

[14] M. Arredondo-Soto, E. Cuan-Urquizo, A. Gómez-Espinosa, A. Roman-Flores, P. D. U. Coronado, and M. Jimenez-Martinez, “The compliant version of the 3-RRR spherical parallel mechanism known as ‘Agile-Eye’: Kinetostatic analysis and parasitic displacement evaluation,” *Mech Mach Theory*, vol. 180, Feb. 2023, doi: 10.1016/J.MECHMACHTHEORY.2022.105160.

[15] G. Yu, L. Wang, J. Wu, D. Wang, and C. Hu, “Stiffness modeling approach for a 3-DOF parallel manipulator with consideration of nonlinear joint stiffness,” *Mech Mach Theory*, vol. 123, pp. 137–152, May 2018, doi: 10.1016/J.MECHMACHTHEORY.2018.01.005.

[16] C. Yang, Q. Li, and Q. Chen, “Analytical elastostatic stiffness modeling of parallel manipulators considering the compliance of the link and joint,” *Appl Math Model*, vol. 78, pp. 322–349, Feb. 2020, doi: 10.1016/J.APM.2019.10.024.

[17] M. B. Gerrard, “Roll centres and jacking forces in independent suspensions - A first principles explanation and a designer’s toolkit,” *SAE Technical Papers*, 1999, doi: 10.4271/1999-01-0046.

[18] M. B. Gerrard, “Kinematic suspension linkages - A model for their behaviour and a procedure for their design,” *SAE Technical Papers*, 2002, doi: 10.4271/2002-01-0281.

[19] K. P. Balike, S. Rakheja, and I. Stiharu, “Kinematic Analysis and Parameter Sensitivity to Hard Points of Five-Link Rear Suspension Mechanism of Passenger Car,” *Proceedings of the ASME Design Engineering Technical Conference*, vol. 5, pp. 755–764, Jul. 2009, doi: 10.1115/DETC2008-49243.

[20] P. A. Simionescu and D. Beale, “Synthesis and analysis of the five-link rear suspension system used in automobiles,” *Mech Mach Theory*, vol. 37, no. 9, pp. 815–832, Sep. 2002, doi: 10.1016/S0094-114X(02)00037-X.

[21] P. A. Simionescu, “A unified approach to the kinematic synthesis of five-link, four-link, and double-wishbone suspension mechanisms with rack-and-pinion steering control,” *http://dx.doi.org/10.1177/0954407016672775*, vol. 231, no. 10, pp. 1374–1387, Nov. 2016, doi: 10.1177/0954407016672775.

[22] F. P. Marchesin, R. S. Barbosa, M. Gadola, and D. Chindamo, “High downforce race car vertical dynamics: aerodynamic index,” *https://doi.org/10.1080/00423114.2017.1413196*, vol. 56, no. 8, pp. 1269–1288, Aug. 2017, doi: 10.1080/00423114.2017.1413196.

[23] F. P. Marchesin, R. S. Barbosa, M. A. L. Alves, M. Gadola, D. Chindamo, and C. Benini, "Upright mounted pushrod: The effects on racecar handling dynamics," *The Dynamics of Vehicles on Roads and Tracks - Proceedings of the 24th Symposium of the International Association for Vehicle System Dynamics, IAVSD 2015*, pp. 543–552, 2016, doi: 10.1201/B21185-59/SIGNIFICANCE-HIGH-ORDER-DYNAMICS-LAP-TIME-SIMULATIONS-LOT-DAL-BIANCO.

[24] F. Marchesin, R. S. Barbosa, M. Gadola, and D. Chindamo, "A Road-Holding Index Based on Ride Dynamics for High-Downforce Racing Cars," *IOP Conf Ser Mater Sci Eng*, vol. 538, no. 1, p. 012069, May 2019, doi: 10.1088/1757-899X/538/1/012069.

[25] M. Gadola, D. Chindamo, G. Legnani, and M. Comini, "Teaching automotive suspension design to engineering students: Bridging the gap between CAD and CAE tools through an integrated approach," *International Journal of Mechanical Engineering Education*, vol. 47, no. 1, pp. 23–43, Jan. 2019, doi: 10.1177/0306419018762803/ASSET/IMAGES/LARGE/10.1177_0306419018762803-FIG9.JPEG.

[26] E. Bonera, M. Gadola, D. Chindamo, S. Morbioli, and P. Magri, "Integrated Design Tools for Model-Based Development of Innovative Vehicle Chassis and Powertrain Systems," *Lecture Notes in Mechanical Engineering*, pp. 118–128, 2020, doi: 10.1007/978-3-030-31154-4_11/COVER.

[27] J. Knapczyk and S. Dzierżek, "Displacement and Force Analysis of Five-Rod Suspension With Flexible Joints," *Journal of Mechanical Design*, vol. 117, no. 4, pp. 532–538, Dec. 1995, doi: 10.1115/1.2826715.

[28] J. Knapczyk and S. Dzierzek, "Elastokinematic Analysis of Five-rod Suspension with Flexible Joints, Including Effects of Shock Absorber," vol. 29, no. SUPPL., pp. 270–279, 2007, doi: 10.1080/00423119808969564.

[29] J. Knapczyk and M. Maniowski, "Elastokinematic modeling and study of five-rod suspension with subframe," *Mech Mach Theory*, vol. 41, no. 9, pp. 1031–1047, Sep. 2006, doi: 10.1016/J.MECHMACHTHEORY.2005.11.003.

[30] Jü. Knapczyk and M. Maniowski, "Optimization Of 5-Rod Car Suspension For Elastokinematic and Dynamic Characteristics," *Archive of Mechanical Engineering*, vol. 57, no. 2, pp. 133–147, 2010, doi: 10.2478/V10180-010-0007-X.

[31] X. Yang, "Effects of bushings characteristics on suspension ball joint travels," *https://doi.org/10.1080/00423110903166300*, vol. 49, no. 1–2, pp. 181–197, Jan. 2009, doi: 10.1080/00423110903166300.

[32] P. Zhao, G. F. Yao, M. Wang, X. Wang, and J. Li, "A new method to calculate the equivalent stiffness of the suspension system of a vehicle," *Structural Engineering and Mechanics*, vol. 44, no. 3, pp. 363–378, 2012, doi: 10.12989/SEM.2012.44.3.363.

[33] G. F. Yao, J. Hou, and P. Zhao, “A new methodology to calculate the equivalent stiffness matrix of the suspension structure with flexible linkages,” *Advances in Mechanical Engineering*, vol. 9, no. 7, Jul. 2017, doi: 10.1177/1687814017700548/ASSET/IMAGES/LARGE/10.1177_1687814017700548-FIG6.JPEG.

[34] M. B. Gerrard, “The equivalent elastic mechanism: A tool for the analysis and the design of compliant suspension linkages,” *SAE Technical Papers*, 2005, doi: 10.4271/2005-01-1719.

[35] J. S. Kang, J. R. Yun, J. M. Lee, and T. O. Tak, “Elastokinematic analysis and optimization of suspension compliance characteristics,” *SAE Technical Papers*, 1997, doi: 10.4271/970104.

[36] J. S. KANG, “ELASTOKINEMATIC ANALYSIS OF A SUSPENSION SYSTEM WITH LINEAR RECURSIVE FORMULA,” *International Journal of Automotive Technology*, vol. 6, no. 4, pp. 375–381, 2005.

[37] E. D. Morales, “Sensitivity study of front suspension parameters in elastokinematics and handling behavior of a vehicle,” *SAE Technical Papers*, vol. Part F127082, no. October, Oct. 2016, doi: 10.4271/2016-36-0182.

[38] J. Meissonnier, J. C. Fauroux, G. Gogu, and C. Montezin, “Geometric Identification of an Elastokinematic Model in a Car Suspension,” *http://dx.doi.org/10.1243/09544070JAUTO239*, vol. 220, no. 9, pp. 1209–1220, Sep. 2006, doi: 10.1243/09544070JAUTO239.

[39] E. Rocca and R. Russo, “A feasibility study on elastokinematic parameter identification for a multilink suspension,” *http://dx.doi.org/10.1243/0954407021528995*, vol. 216, no. 2, pp. 153–160, Feb. 2002, doi: 10.1243/0954407021528995.

[40] L. Tang, W. Bin Shangguan, and L. Dai, “A calculation method of joint forces for a suspension considering nonlinear elasticity of bushings,” *http://dx.doi.org/10.1177/1464419312452214*, vol. 226, no. 4, pp. 281–297, Aug. 2012, doi: 10.1177/1464419312452214.

[41] M. J. Burgess, N. P. Fleming, M. Wootton, and S. J. Williams, “A tool for rapid vehicle suspension design,” *SAE Technical Papers*, 2004, doi: 10.4271/2004-01-3543.

[42] J. Knapczyk and S. Dzierzek, “Elastokinematic Analysis of Five-rod Suspension with Flexible Joints, Including Effects of Shock Absorber,” *Vehicle System Dynamics*, vol. 29, no. SUPPL., pp. 270–279, 1998, doi: 10.1080/00423119808969564.

[43] D. Ammon, M. Gipser, J. Rauh, and J. Wimmer, “High Performance System Dynamics Simulation of the Entire System Tire-Suspension-Steering-Vehicle,” *Vehicle System Dynamics*, vol. 27, no. 5–6, pp. 435–455, 1997, doi: 10.1080/00423119708969341.

[44] Y. Duan, Y. Zhang, J. Wu, and T. Lv, “A data-driven suspension kinematics and compliance model considering multi-axis coupling effects,” *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 2023, doi: 10.1177/09544070231207160.

[45] J. Park, J. Yi, D. Lee, J. Park, J. Yi, and D. Lee, “Investigation into Suspension Dynamic Compliance Characteristics Using Direct Measurement and Simulation,” *SAE Technical Papers*, Mar. 2004, doi: 10.4271/2004-01-1065.

[46] A. T. Van Zanten, “Bosch ESP Systems: 5 Years of Experience,” *SAE Technical Papers*, May 2000, doi: 10.4271/2000-01-1633.

[47] C. Crema *et al.*, “Smartphone-based system for the monitoring of vital parameters and stress conditions of amatorial racecar drivers,” *2015 IEEE SENSORS - Proceedings*, Dec. 2015, doi: 10.1109/ICSENS.2015.7370521.

[48] D. Chindamo, M. Gadola, and F. P. Marchesin, “Reproduction of real-world road profiles on a four-poster rig for indoor vehicle chassis and suspension durability testing,” *Advances in Mechanical Engineering*, vol. 9, no. 8, pp. 1–10, Aug. 2017, doi: 10.1177/1687814017726004/ASSET/IMAGES/LARGE/10.1177_1687814017726004-FIG11.JPEG.

[49] D. Chindamo and M. Gadola, “What is the Most Representative Standard Driving Cycle to Estimate Diesel Emissions of a Light Commercial Vehicle?,” *IFAC-PapersOnLine*, vol. 51, no. 5, pp. 73–78, Jan. 2018, doi: 10.1016/J.IFACOL.2018.06.213.

[50] D. Chindamo, B. Lenzo, and M. Gadola, “On the vehicle sideslip angle estimation: a literature review of methods, models and innovations,” *Appl. Sci.*, vol. 8, no. 3, p. 355, Mar. 2018, doi: 10.3390/app8030355.

[51] M. J. Burgess, N. P. Fleming, M. Wootton, and S. J. Williams, “A Tool for Rapid Vehicle Suspension Design,” 2018.

[52] F. P. Marchesin, R. S. Barbosa, M. A. L. Alves, M. Gadola, D. Chindamo, and C. Benini, “Upright mounted pushrod: The effects on racecar handling dynamics,” *The Dynamics of Vehicles on Roads and Tracks - Proceedings of the 24th Symposium of the International Association for Vehicle System Dynamics, IAVSD 2015*, pp. 543–552, 2016, doi: 10.1201/B21185-59.

[53] D. Chindamo and M. Gadola, “Estimation of Vehicle Side-Slip Angle Using an Artificial Neural Network,” *MATEC Web of Conferences*, vol. 166, p. 02001, Apr. 2018, doi: 10.1051/MATECCONF/201816602001.

[54] G. G., “Motor vehicle dynamics: modeling and simulation,” 1997, *World Scientific*.

[55] L. K., “A Method for the Solution of Certain Problems in Least-Squares,” *Quarterly Applied Math.*, vol. 2, pp. 164–168, 1944.

[56] M. D., “An Algorithm for Least-Squares Estimation of Nonlinear Parameters,” *SIAM Journal Applied Math.*, vol. 11, pp. 431–441, 1963.

[57] E. G., “Data assimilation,” 2009, *Springer-Verlag*.

[58] M. Gadola, D. Chindamo, and B. Lenzo, “Revisiting the mechanical limited-slip differential for high-performance and race car applications,” *Engineering Letters*, vol. 29, no. 3, pp. 824–839, Sep. 2021.

[59] G. Mastinu and E. BATTISTINI, “THE INFLUENCE OF LIMITED-SLIP DIFFERENTIALS ON THE STABILITY OF REAR- WHEEL-DRIVE AUTOMOBILES RUNNING ON EVEN ROAD WITH DRY SURFACE,” *International Journal of Vehicle Design*, vol. 14, no. 2/3, 1993.

[60] H. Huchtkoetter and H. Klein, “The Effect of Various Limited-Slip Differentials in Front-Wheel Drive Vehicles on Handling and Traction,” *SAE Technical Papers*, Feb. 1996, doi: 10.4271/960717.

[61] “Race Car Vehicle Dynamics - Milliken, William F.; Milliken, Douglas L.: 9781560915263 - AbeBooks.” Accessed: Apr. 10, 2024. [Online]. Available: https://www.abebooks.it/9781560915263/Race-Vehicle-Dynamics-Milliken-William-1560915269/plp