UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

MOTOR VALLEY
UNIVERSITY OF
EMILIA-ROMAGNA

Motor Valley University of Emilia-Romagna

Advanced Automotive Engineering

Racing Car Design

DEVELOPMENT OF A SUSPENSION KINEMATICS DESIGN TOOL

Supervisor
Prof. Ing. Andrea Toso

Company Supervisor
Ing. Jacopo Gentile

Candidate
Francesco Pio Cotugno

Academic Year 2025-2026

Abstract

Suspension hardpoint definition is one of the first steps in racing car development, as small changes in their position may lead to significant differences in kinematic behaviour. Camber gain, toe variation, Ackermann characteristic and roll centre migration strongly depend on how these points are spatially arranged around the car. For this reason, suspension kinematic synthesis is often an iterative and labour-intensive process, where the engineer adjusts the geometry and evaluates its effect on the main performance targets.

This thesis presents a Python-based optimization tool for suspension kinematic synthesis. The tool supports and streamlines this process while preserving the use of existing Excel-based engineering models. It interfaces with validated suspension workbooks, imports the initial geometry and target kinematic curves, updates the selected hardpoint coordinates and evaluates the resulting behaviour through the original calculation environment. The optimization is guided by an objective function based on the weighted difference between target and calculated kinematic values, allowing the user to prioritize selected behaviours according to the specific design objective.

A key advantage of the proposed methodology is the possibility to optimize only a subset of the Cartesian coordinates defining the hardpoint locations. The remaining coordinates are kept fixed, ensuring that the process respects packaging constraints, modelling assumptions and engineering judgement. To avoid altering the original files, the procedure is performed on automatically created copies of the Excel workbooks, preserving traceability and the integrity of the validated models.

The tool was validated on two representative suspension case studies, showing its ability to converge towards the assigned targets and support a more methodical exploration of the design space. The results validate the proposed workflow and show how it can improve the efficiency of suspension development by reducing manual trial-and-error iterations and providing a repeatable framework for kinematic optimization.

Table of Contents

List of Figures

List of Tables

Chapter 1 – Introduction

The present chapter introduces background, motivation and objectives of this thesis. This work concerns the development of a Python-based optimization framework for suspension kinematic synthesis and focuses on automating the modification of selected hardpoint coordinates while still preserving the use of existing Excel-based kinematic models. The background information and problem definition are first presented, followed by the aim of the thesis, adopted methodological approach and thesis structure.

1.1 Background and Motivation

Vehicle development involves several engineering disciplines such as aerodynamics, structural design, powertrain tuning and vehicle dynamics. Among these subjects, suspension design has primary importance as it defines how the vehicle will physically interact with the road through the tyres. A suspension must provide enough clearance to allow vertical wheel movement while supporting the car mass, but also manage wheel position and orientation over different operating conditions.

Suspension kinematics is of particular interest for high performance and racing applications where small differences in wheel alignment can significantly alter the handling balance and steering response of the car. Kinematic quantities such as camber gain, toe variation, roll centre height and steering-related responses depend strongly on the suspension geometry and hardpoint positioning [1], [2].

For these reasons, suspension kinematic synthesis is typically carried out through an iterative design process during which the engineer modifies suspension hardpoints to tune the kinematic response to a satisfactory level. While retaining manual control over the optimization process allows the designer to retain direct control over the engineering decisions, having to test several kinematic variables one at a time can be time consuming.

This limitation is further increased by the fact that suspension parameters are usually coupled. An alteration to the suspension geometry designed to improve one kinematic response will likely influence other responses too, sometimes in undesired ways. For instance, moving the location of a wishbone hardpoint may alter camber gain but may also affect roll centre height or toe variation. Similarly, modifying the steering linkage geometry could reduce bump steer but also influence steering ratio or other steering-related

responses. As a result, suspension kinematic synthesis becomes in general a multi-objective task, where multiple responses need to be considered and traded off against each other. This makes the process difficult to manage manually and motivates the use of a more structured optimization procedure during the design phase [3], [4].

Numerical tools can help to automate this task in modern development environments by making the evaluation of design alternatives faster and more repeatable. Optimization routines are well suited for this purpose as they allow some user-defined design variables to be modified automatically until the selected target criteria are approached. However, for the framework to be adopted in an engineering context, it should be able to interface with existing kinematic models rather than replacing them completely.

In this regard, the present work aims to address the need for an easier and more repeatable suspension kinematic synthesis process through the development of an optimization framework based on Python scripting. Integrated with existing Excel-based kinematic models, the framework automates the process of testing new hardpoint configurations while remaining compatible with the established modelling workflow.

1.2 Problem Statement

The suspension kinematic synthesis process involves defining a hardpoints layout which provides wheel motion and alignment changes matching a desired target. While working on a suspension design, this is often achieved by manually modifying selected hardpoint coordinates and observing how the resulting kinematic outputs change. As wheel alignment and suspension behaviour depend directly on suspension hardpoint positions, defining their location is considered one of the key tasks during suspension design [2].

Manually tuning individual hardpoints can be feasible when only a few design variables are involved. However, the complexity of this task increases with the number of hardpoints and kinematic targets to consider. Literature examples highlight that double wishbone suspension geometry typically features several design variables, nonlinear kinematic behaviour and conflicting objectives that need to be considered simultaneously [3].

One challenge lies in the inherent coupling between different suspension responses. As previously mentioned, camber gain, toe variation, roll centre height and steering-related quantities are driven by the same underlying suspension geometry. Therefore, designing the suspension geometry to improve one response often affects the others, making the manual identification of an ideal hardpoint configuration particularly challenging [2], [3].

A second issue is related to the adoption of existing engineering tools. Many development environments already rely on pre-existing kinematic models, typically implemented in Excel. These models embed existing calculation routines and represent part of the current engineering workflow. Building a completely new standalone model would require additional validation work and could create discontinuity with the pre-existing design process.

The proposed development therefore does not only deal with the numerical optimization of suspension design variables, but also with its integration within pre-existing Excel-based models. By combining these two goals, the framework described in this thesis automatically explores new sets of hardpoint coordinates while preserving the original modelling environment used to calculate suspension responses.

Achieving this requires the developed framework to read the initial suspension geometry, modify only the selected coordinates, update the resulting kinematic outputs through existing spreadsheets, compare the responses to user-defined targets and solve the resulting optimization problem automatically. Ideally, the framework should also be flexible enough to accommodate different optimization configurations such as vertical-only kinematic synthesis or combined vertical and steering synthesis tasks.

1.3 Aim of the Thesis

The aim of this thesis is to develop and evaluate a Python-based optimization framework to support suspension kinematic synthesis. The framework will allow the user to modify selected suspension hardpoint coordinates in search of improved agreement between calculated kinematic responses and user-defined targets.

The work focuses on integrating numerical optimization routines with pre-existing Excel-based kinematic models. Rather than replacing them, the proposed framework uses these models as external evaluation functions within an automatic optimization loop. In this way, the original calculation structure remains unchanged, while the optimization

process is performed externally by the Python tool. This preserves compatibility with existing engineering workflows and makes hardpoint synthesis faster, more systematic and more repeatable, while reducing the need for repeated manual modifications of the suspension geometry during the preliminary design phase.

The resulting framework will be structured in order to allow the user to select which hardpoint coordinates can be modified, define admissible ranges for coordinate variations, choose which kinematic outputs are used as targets and set their relative importance through weighting coefficients and sensitivity factors.

Finally, the thesis will assess the behaviour of the proposed framework through a series of case studies. These will be used to evaluate the tool's capability to optimize vertical suspension kinematics, quantify the computational overhead introduced by the steering model and explore the usage of combined vertical and steering-related optimization tasks.

The main goal of this thesis is therefore not to provide a fully automatic method to generate final suspension designs independently of the designer, but to provide a tool that structures and automates parts of the synthesis process in order to help engineers explore potential improvements to their hardpoint configuration.

1.4 Methodological Approach

The thesis follows a methodological approach based on interfacing numerical optimization routines with pre-existing Excel-based suspension kinematic models. More specifically, the Python scripting language is used as the main automation platform while preserving the original Excel spreadsheets as the source of the kinematic calculations.

The communication between Python scripts and Excel workbooks is handled through the xlwings [5] library. This allows reading, writing and controlling Excel workbooks directly from Python scripts. In practice, this will allow modifying suspension coordinates, triggering workbook recalculations and reading the resulting kinematic outputs without manual intervention at each iteration of the optimization loop.

The optimization itself starts from an initial set of suspension coordinates. The user defines which hardpoint coordinates are active during the optimization, specifies their admissible upper/lower bounds and sets the target values for selected kinematic outputs. Only these active coordinates are considered as design variables, while all other coordinates stay fixed during the entire optimization process.

For each new set of variables proposed during optimization, the framework writes the updated coordinates back to the Excel-based kinematic model. It then triggers a calculation of the spreadsheet and reads back the resulting kinematic outputs of interest. These outputs can include vertical kinematic quantities such as camber gain, toe variation or roll centre height, but can also include steering-related quantities such as steering ratio or steer heave.

The retrieved kinematic outputs are finally compared to their respective target values and fed into an objective function. This function computes a single scalar value summarizing the overall agreement between calculated responses and target values. Weighting coefficients and sensitivity factors allow tuning the relative importance given to each response as well as their individual numerical scaling. This allows treating multiple kinematic outputs within the same optimization problem, while still giving the user control over the trade-offs between conflicting targets.

Numerical optimization is performed using SciPy's optimization toolkit [6]. In particular, a constrained optimization approach capable of enforcing upper/lower bounds on the design variables is adopted. Starting from the initial geometry, the optimizer iteratively proposes new sets of design variables by trying to minimize the objective function while staying inside the feasible design space.

Once the optimization framework is developed, its performance is validated through several case studies. These are used to analyze its behaviour with different optimization configurations, ranging from vertical-only suspension optimization to the inclusion of the steering model and complete vertical + steering-related optimization problems. The case studies therefore allow the framework to be evaluated not only in terms of the obtained kinematic results, but also in terms of practical usage and incurred computational cost.

1.5 Main Contributions

The main contribution of this thesis is the development of a Python-based framework for automating suspension kinematic optimization while retaining the use of existing Excel-based kinematic models. The tool acts on selected suspension hardpoint coordinates within a structured optimization workflow, without altering the original spreadsheets or the validated calculation environment.

The framework adopts a flexible definition of design variables, where each Cartesian coordinate of each hardpoint can be activated or excluded independently. In this way, the optimization can be limited to the coordinates that are physically meaningful or practically admissible, while coordinates constrained by packaging, modelling or design requirements remain fixed throughout the process.

A Configuration Workbook is introduced to define the optimization setup for each run without requiring direct changes to the Python code. Through this workbook, the user specifies the active kinematic models, design variables, admissible coordinate variations, target values and weighting parameters. The framework then handles the interaction with the Excel models by importing the initial geometry, creating writable copies of the workbooks, updating the selected coordinates, recalculating the models and reading the resulting kinematic responses during each run.

The thesis also defines an objective function that combines multiple weighted kinematic responses into a single scalar quantity. This makes it possible to include vertical and steering-related targets within the same optimization problem, while still allowing their relative importance to be controlled.

Finally, the framework is assessed through representative case studies, including vertical-only optimization, the evaluation of the additional computational cost introduced by the steering model, and a combined vertical and steering-related optimization. These applications illustrate the practical use of the tool and quantify the trade-offs between kinematic improvement, target prioritization and computational effort.

1.6 Thesis Structure

This thesis comprises six chapters.

Chapter 1 introduces the background, motivation, problem statement and thesis aim.

Chapter 2 summarizes background information on vehicle and suspension kinematics required to understand the presented framework.

Chapter 3 describes the optimization methodology adopted in this thesis.

Chapter 4 details the software architecture and implementation of the framework.

Chapter 5 presents the case studies used for validation of the framework.

Chapter 6 concludes this thesis and outlines possible directions for future work.

Chapter 2 – Suspension Kinematics Background

Chapter 2 gives the background needed to understand the suspension optimization framework developed in this thesis. Since the tool acts on suspension hardpoint coordinates, the chapter focuses first on the double wishbone layout and on the meaning of hardpoints, then introduces the main kinematic responses used later as optimization targets: camber gain, toe variation, roll centre height and steering-related quantities.

2.1 Role of Suspension Kinematics in Vehicle Dynamics

The suspension system is the mechanical interface between the vehicle body and the wheels. Besides supporting the vehicle mass and filtering road surface irregularities, it also controls how the wheel moves with respect to the chassis. For this reason, suspension kinematics has a direct influence on how the tyre contact patch is positioned during bump, rebound, steering and cornering manoeuvres [1].

From a vehicle dynamics point of view, the motion imposed by the suspension affects several quantities related to handling, stability and driver feel. Camber angle, toe angle, roll centre position and steering geometry all change the orientation or position of the wheel as the suspension moves. These changes influence how the vehicle generates lateral force, reacts to steering inputs and remains stable in different operating conditions [1].

This becomes especially important in high-performance and racing applications, where small wheel-alignment variations can noticeably change the balance of the car. In these cases the suspension is not designed only for ride comfort, but also to manage tyre attitude and wheel motion throughout the useful travel range. Suspension kinematic synthesis is therefore an important part of the vehicle design process [1], [2].

The kinematic response of a suspension is mainly dictated by the geometry of its links and joints. Hardpoint positions define where the chassis, control arms, upright and wheel are connected to each other. Moving these points changes the way the wheel travels in bump, rebound and steering, and previous suspension design studies show that hardpoint locations are closely related to wheel alignment parameters and overall suspension performance [2].

In vehicle development, suspension behaviour is often studied through Kinematics and Compliance evaluations. Kinematics considers the rigid-body motion of the suspension mechanism, while compliance also accounts for elastic deformation of bushings and structural components. K&C characteristics are commonly used to interpret steering, comfort, ride and handling behaviour under wheel travel, and they provide useful target quantities for suspension development [4].

This thesis focuses on the kinematic part of the problem. The aim is to modify selected hardpoint coordinates in order to obtain desired kinematic responses from existing models. The next sections introduce the double wishbone suspension and the main kinematic parameters involved in this process.

2.2 Double Wishbone Suspension Layout

The double wishbone suspension is one of the architectures most commonly associated with high-performance and racing cars, mainly because it gives the designer a high degree of control over wheel motion. The layout uses two control arms, usually referred to as upper and lower wishbones, to connect the chassis to the upright. The upright carries the wheel assembly and, on a front axle, also provides the connection to the steering system through the steering arm [1], [2].

The main components of a typical double wishbone suspension are shown in Figure 1.

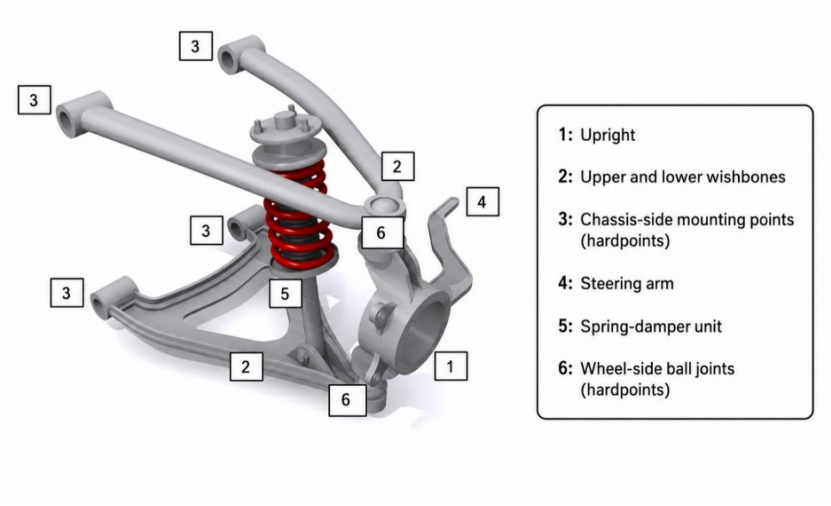

Figure 1 - Main components of a double wishbone suspension. Chassis-side mounting points and wheel-side ball joints are highlighted as the main hardpoint groups defining the suspension geometry. Redrawn and adapted by the author from the suspension layout shown in [7].

Kinematically speaking, the double wishbone can be viewed as a spatial linkage mechanism. The relative position of the upper arm, lower arm and upright determines how the wheel moves as it goes through bump and rebound. By changing the geometry of these links, the designer can influence camber gain, toe variation, track variation, roll centre position and other kinematic quantities [1], [3].

This is one of the main advantages of the double wishbone layout. By selecting the length, inclination and relative placement of the upper and lower arms, it is possible to shape the instantaneous motion of the upright and therefore the variation of the wheel alignment angles during suspension travel. This makes the layout particularly suitable for kinematic synthesis and optimization studies [2], [3].

In a simplified representation, each wishbone can be treated as a link connected to the chassis through inner joints and to the upright through an outer joint. The inner joints describe the attachment to the vehicle body, while the outer joints define the connection to the wheel carrier. As shown in Figure 1, these joints form the basic hardpoint groups that define the suspension geometry [2].

Especially for racing applications, this architecture is useful because the geometry can be tuned to meet specific handling targets. By modifying the hardpoint layout, the designer can influence camber behaviour, bump steer, roll centre height and steering-related responses. These quantities are coupled, however, so a change that improves one response may also alter another. This is one reason why double wishbone suspension synthesis lends itself well to optimization [3].

The suspension layout considered in this thesis follows the same principle. Its geometry is described through a set of hardpoints that define the position of the links and joints. The optimization framework acts on selected hardpoint coordinates, modifying the kinematic response while keeping the general structure of the existing model unchanged.

2.3 Suspension Hardpoints

Suspension hardpoints define the location of joints and connections between the main suspension components. In a double wishbone suspension, they identify the connections between the control arms and the chassis, the control arms and the upright and, if the axle is steered, the steering linkage points [2].

From a kinematic point of view, these points define the suspension geometry. Their position determines the length, inclination and relative orientation of the suspension links. As a result, the hardpoint layout directly affects upright trajectory and wheel orientation during bump, rebound and steering motion [2], [4].

The role of hardpoints is especially clear in a double wishbone suspension. The upper and lower arms constrain upright motion, and their relative placement determines wheel behaviour. By moving inner or outer joints, the designer can influence camber gain, toe variation, roll centre height and steering-related characteristics. Hardpoint definition is therefore one of the main tasks in suspension kinematic synthesis [2].

A hardpoint is usually described by its Cartesian coordinates in the reference system adopted by the suspension model. Even a small movement of a single coordinate can change the mechanism geometry. The same hardpoint can also affect several kinematic outputs, because the suspension responses are coupled through the link geometry [2], [3].

For example, changing the vertical position of an inner wishbone joint modifies the control-arm inclination and can strongly affect camber gain and roll centre position. Lateral movements of suspension joints can also change the relative motion between upright and chassis, affecting wheel alignment during travel. In a steered suspension, tie-rod hardpoints strongly influence steering-related quantities and bump steer [2], [4].

In practice, not every hardpoint can be moved freely. Packaging, structural requirements, component integration, manufacturing constraints and vehicle architecture may all limit the admissible motion of a joint. For this reason, synthesis and optimization procedures are normally restricted to physically feasible coordinates.

The optimization approach used in this thesis follows this idea. The hardpoint layout of an existing suspension model is taken as the reference geometry, and only selected coordinates are allowed to vary. This makes it possible to change the kinematic response while preserving the overall structure of the original model.

2.4 Main Kinematic and Steering-Related Parameters

The kinematic behaviour of a suspension is commonly described through geometrical quantities that define the position and orientation of the wheel relative to the vehicle body. Camber gain, toe variation, roll centre height and steering-related parameters are particularly relevant, because they influence tyre behaviour and the vehicle response during vertical travel, steering and cornering [1], [4].

These quantities are not independent from the suspension layout. They are a consequence of the spatial arrangement of hardpoints and of the relative motion between chassis, control arms, upright, steering linkage and wheel. Changing the hardpoint layout therefore changes the evolution of these parameters during suspension travel and steering motion [2].

2.4.1 Camber Gain

Camber angle describes the inclination of the wheel plane with respect to the vertical direction when the vehicle is viewed from the front. If the upper part of the wheel is inclined towards the vehicle centreline, the wheel is described as having negative camber; if it is inclined in the opposite direction, it has positive camber [1].

Camber gain is the change in camber angle caused by vertical suspension travel. It describes how the wheel camber evolves as the wheel moves in bump or rebound. In a double wishbone suspension, this behaviour is strongly influenced by the relative length, inclination and position of the upper and lower control arms [1], [2].

From a vehicle dynamics perspective, camber gain affects the orientation of the tyre contact patch with respect to the road. During cornering, the suspension geometry should help maintain a favourable tyre attitude so that the tyre can generate lateral force efficiently. For this reason, camber gain in bump is often used as a design target in performance and racing applications [1].

Camber gain still has to be tuned carefully. Excessive camber change may give an unfavourable tyre attitude in some conditions, increase tyre loading asymmetry or make the vehicle response more sensitive. The target camber behaviour is therefore usually selected as a compromise between lateral performance, tyre usage and overall balance.

An important part of hardpoint optimization is camber gain, because camber gain is sensitive to the relative position of wishbone mounting points. Altering either the inner or outer hardpoint will change the instant centre location and thus change the camber curve throughout suspension travel. As a result camber-related targets act as good metrics of how well a new geometry holds or improves wheel attitude.

2.4.2 Toe Variation and Bump Steer

Toe angle describes the orientation of the wheel when the vehicle is viewed from above. If the front part of the wheel points towards the vehicle centreline, the wheel is said to have toe-in; if it points away from it, it is said to have toe-out [1].

Toe variation during bump and rebound is commonly associated with bump steer. This occurs when the wheel changes its steering angle as a consequence of suspension motion, even without an intentional steering input from the driver. On a front suspension, uncontrolled toe variation can reduce straight-line stability, alter steering response and affect driver confidence and consistency [1], [4].

The toe behaviour depends on the relative motion between the upright and the steering linkage. In a steered axle, the tie-rod hardpoints are particularly important because they define how the steering system follows the movement of the wheel carrier. If the steering linkage is not properly matched with the suspension motion, vertical wheel travel can generate unwanted toe changes [4].

For this reason, toe variation is treated as a sensitive parameter during suspension design. Small angular changes can have a noticeable effect on perceived stability and responsiveness, so toe-related targets often receive significant attention in kinematic synthesis and optimization procedures [4].

In the context of this thesis, toe variation is therefore not considered only as a secondary steering effect, but also as one of the main responses used to evaluate the quality of a modified hardpoint configuration. Its control is important because even small toe changes can influence the consistency of the vehicle response during wheel travel.

2.4.3 Roll Centre Height

The roll centre is a geometrical point associated with the lateral motion of the suspension and with the path through which lateral forces are transferred between the tyre contact patch and the vehicle body. In a double wishbone suspension, its position is determined by the geometry of the control arms and by the location of the instantaneous centre of the mechanism [1].

The front-view geometrical construction of the instantaneous centre, roll centre and roll centre height is shown in Figure 2.

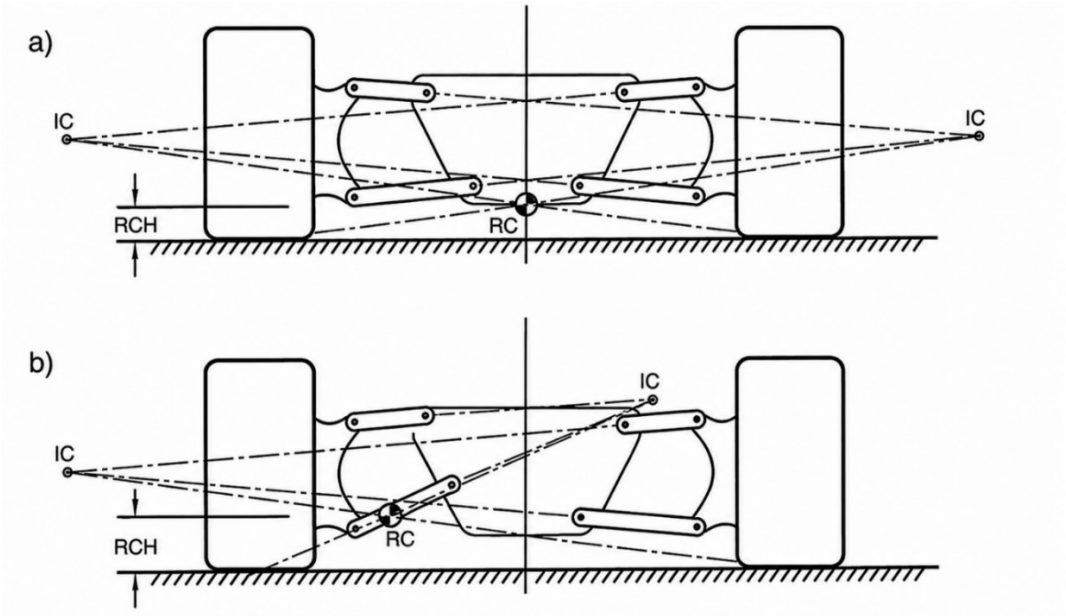

Figure 2 - Front-view construction of the instantaneous centre, roll centre and roll centre height for independent suspension layouts. Redrawn and adapted from Milliken and Milliken [1].

Figure 2 shows the front-view construction used to determine the roll centre of an independent suspension in static conditions. The dashed lines extend the upper and lower suspension links, and their intersection defines the instantaneous centre (IC) for each side of the suspension. A construction line is then drawn from each instantaneous centre to the corresponding tyre contact patch. Where these lines intersect the vehicle centre plane, the roll centre (RC) is obtained. The vertical distance, measured in front view, between the roll centre and the ground plane is the roll centre height (RCH).

In case (a), the suspension is shown in a symmetric reference condition. The left and right instantaneous centres are located on opposite sides of the vehicle, and the resulting roll centre lies on the vehicle centreline. This is the usual construction used to identify the roll centre position for a static suspension configuration.

In case (b), the geometry is shown in a non-symmetric condition, representative of a different wheel travel or roll configuration. The instantaneous centres move because the suspension geometry has changed, and the roll centre is no longer in the same position as in the symmetric case. This shows that the roll centre is not fixed on the vehicle, but can migrate vertically and laterally as the suspension moves.

This construction is relevant for suspension kinematic synthesis because hardpoint changes modify the orientation of the control arms, the location of the instantaneous centres and, consequently, the roll centre height. Roll centre behaviour must therefore be considered together with other responses such as camber gain and toe variation.

Roll centre height is an important design parameter because it influences the balance between geometric and elastic contributions to vehicle roll behaviour. Changing the roll centre height modifies the lever arm between the centre of gravity and the roll centre, affecting roll moment distribution, jacking effects and the vehicle response under lateral acceleration during cornering manoeuvres [1].

During kinematic synthesis, roll centre height is usually considered together with camber gain and toe behaviour. Hardpoint modifications that improve one of these responses may also alter the others. For instance, moving the inner wishbone joints vertically may help reach a desired camber gain, but it can also produce a significant change in roll centre height [2].

For this reason, roll centre height is not normally optimized in isolation. It is part of a wider compromise that includes wheel alignment variation, steering behaviour and packaging constraints. In the framework developed in this thesis, roll centre height is treated as one of the possible targets used to guide the synthesis process.

2.4.4 Steering Ratio

Steering ratio describes the relationship between the steering input and the resulting angular displacement of the steered wheels. Depending on the model convention, it can be expressed either as the ratio between steering wheel angle and road wheel angle, or as a relationship between rack travel and wheel steer angle [4].

This parameter directly affects the steering response perceived by the driver. A lower steering ratio generally gives a more direct response, while a higher steering ratio makes the wheels react less for the same steering input. In racing applications, steering ratio is important because it contributes to the balance between precision, responsiveness and controllability over the full steering range [1], [4].

From a kinematic point of view, steering ratio depends on the steering linkage geometry, including the steering rack position and the tie-rod inner and outer joints on the upright. Since these hardpoints define the motion transmission between rack and wheel carrier, their position can affect the steering response over the rack travel range [4].

2.4.5 Steer Heave

Steer heave describes the vertical displacement of the wheel centre associated with steering motion. Ideally, steering input should mainly rotate the wheel around the steering axis, but the suspension and steering geometry can also induce a vertical movement of the wheel centre during steering [4].

This effect introduces a coupling between steering motion and vertical wheel displacement. Depending on the geometry, steering can therefore produce a small lifting or lowering effect at the wheel. From a vehicle dynamics point of view, this may contribute to variations in vertical load distribution and may influence steering feel and consistency during cornering manoeuvres.

Steer heave is affected by the spatial arrangement of the upright, the steering axis and the tie-rod geometry. It can therefore be treated as one of the steering-related kinematic responses that may need to be monitored or controlled during suspension synthesis. This is particularly relevant because an excessive vertical displacement in the steering system can modify the relative position between the steering links and the wheel assembly, leading to less predictable behaviour over the considered wheel travel range.

2.4.6 Steer Roll

Steer roll describes the roll-related effect generated by steering motion. It is usually associated with an asymmetric vertical displacement, or a different kinematic response, between the left and right sides of the axle during steering. As the wheels steer, the suspension and steering geometry may therefore produce a small roll contribution.

Although steer roll is less intuitive than camber gain or toe variation, it can still be useful when steering kinematics are evaluated. It represents another coupling between steering input and vehicle attitude, and excessive steer-induced roll may affect the consistency of the vehicle response and the balance perceived by the driver.

The magnitude and sign of steer roll depend on the suspension layout, steering axis geometry and tie-rod arrangement. It is usually interpreted together with other steering-related quantities, such as steering ratio, steer heave, Ackermann behaviour and wheel centre displacement during steering [4].

2.5 Kinematic Coupling and Design Trade-Offs

Most suspension kinematic parameters are coupled because they have the same geometrical origin. Wheel motion is not controlled by independent elements dedicated to a single output, but by the combined arrangement of control arms, upright, steering linkage and hardpoints. Adjusting a hardpoint therefore normally affects more than one suspension behaviour at once [2], [3].

This coupling is particularly clear in a double wishbone suspension. The position of the upper and lower control arm hardpoints defines the inclination and motion of the upright during wheel travel. Modifying these points can influence camber gain, roll centre height and track variation at the same time. Similarly, tie-rod hardpoints influence both toe variation during vertical wheel motion and steering response during rack travel, especially in steered suspension configurations [2], [4].

Kinematic coupling means that suspension synthesis is rarely a single-objective process. Improving one behaviour may make another less favourable. Increasing camber gain in bump may improve tyre attitude during cornering, but the same geometrical change may also shift the roll centre height or introduce undesired toe variation. In the same way, modifying the tie-rod position to reduce bump steer may also affect steering ratio or other steering-related quantities [3], [4].

The final suspension geometry is therefore the result of a compromise between different requirements. The objective is not simply to maximize or minimize every parameter independently, but to obtain a combination of responses that matches the intended vehicle behaviour. This is especially true in racing applications, where suspension layout has a strong influence on handling performance, predictability and driver confidence under demanding operating conditions [1], [3].

How this compromise is defined depends on the design objective. A configuration aimed at maximum lateral performance may give more priority to camber gain and roll centre control, while a stability-oriented setup may focus more on toe behaviour and steering consistency. The relative importance of each parameter must therefore be chosen according to the intended vehicle behaviour and operating conditions.

These couplings also explain why hardpoint movements must be considered carefully. A hardpoint that is very effective for changing one response may be unsuitable if the same movement produces undesirable effects elsewhere. The relationship between hardpoints and kinematic quantities should therefore be treated qualitatively during the design process, while the final engineering judgement should be based on the complete suspension response over the full travel range [2], [3].

The optimization framework presented in this thesis helps to manage this type of trade-off by allowing several kinematic targets to be considered within the same numerical process. The physical coupling between parameters remains, however. The optimizer can search for a better compromise inside the selected design space, but interpretation of the result is still an engineering task.

2.6 Relationship Between Kinematic Parameters and Hardpoint Layout

There is no fixed one-to-one relationship between suspension hardpoints and kinematic parameters. A single hardpoint movement can influence several suspension behaviours, and the same kinematic quantity can be affected by different hardpoint groups. For this reason, the following table should be read as a qualitative engineering guideline rather than as an absolute rule.

The most influential hardpoints depend on the suspension architecture, reference system, linkage geometry and available design space. Within a double wishbone suspension, however, some general relationships can still be identified between the main kinematic responses and the hardpoint groups that most commonly affect them in practical suspension design [1], [2], [4]. The qualitative relationships between the main kinematic parameters and the most influential hardpoint groups are summarized in Table 1.

Table 1 - Qualitative relationship between suspension kinematic parameters and the most influential hardpoint groups.

Kinematic parameter	Description	Positive effect	Possible drawback	Most influential hardpoints
Camber gain	Camber variation during vertical wheel travel.	Improved tyre attitude and lateral force generation.	Uneven tyre loading, tyre wear, sensitive response.	Upper/lower wishbone hardpoints; vertical and lateral positions.
Toe variation / bump steer	Toe variation during bump and rebound.	Improved stability and steering consistency.	Reduced straight-line stability, undesired steering effects.	Tie-rod hardpoints, rack position, steering arm geometry.
Roll centre height	Geometrical lateral force transfer parameter.	Roll behaviour, load transfer and balance control.	Jacking effects, undesirable roll/load transfer behaviour.	Upper/lower wishbone hardpoints; inner joints; outer ball joints.
Steering ratio	Relation between steering input and wheel angle/rack travel.	Steering directness, precision and controllability.	Nervous response or reduced agility.	Rack position, tie-rod hardpoints, steering arm length/position.
Steer heave	Wheel-centre vertical displacement induced by steering.	Consistent steering behaviour and predictable wheel motion.	Coupling between steering and vertical load distribution.	Steering axis, upright hardpoints, tie-rod geometry, rack position.
Steer roll	Roll-related effect induced by steering motion.	Consistent vehicle attitude during steering manoeuvres.	Balance, steering feel and predictability issues.	Steering axis inclination, upright geometry, tie-rods, rack symmetry.

This qualitative relationship explains why hardpoint optimization should normally consider more than one output at a time. Moving the inner wishbone hardpoints may be effective for camber gain adjustment, but it can also change roll centre height. Similarly, modifying the tie-rod hardpoints may help reduce bump steer, while also influencing steering ratio, steer heave and other steering-related responses.

Hardpoint selection must therefore consider the coupling between kinematic quantities. The optimization process can support the engineer by evaluating the combined effect of hardpoint movements on several targets, but the final geometry still requires engineering judgement based on the specific vehicle design objectives.

2.7 Chapter Summary

This chapter introduced the suspension kinematics concepts needed to understand the optimization framework developed in this thesis. The role of suspension kinematics in vehicle dynamics was discussed first, showing how wheel motion relative to the chassis affects handling, stability, steering response and tyre behaviour.

The double wishbone suspension layout was then presented as a geometry that allows precise control of wheel motion, making it suitable for high-performance and racing applications. The importance of suspension hardpoints was also introduced, since their relative position defines the suspension geometry and directly affects the resulting kinematic behaviour.

The main kinematic and steering-related parameters used throughout this work were then presented, including camber gain, toe variation, roll centre height, steering ratio, steer heave and steer roll. These quantities describe how the wheel position and orientation evolve during vertical travel and steering motion.

Finally, the chapter highlighted the coupling between suspension parameters and hardpoint locations. Since moving one hardpoint generally affects several kinematic responses at the same time, suspension synthesis must be treated as a compromise between different and sometimes conflicting requirements. This coupling provides the technical motivation for the optimization methodology introduced in Chapter 3.

Chapter 3 – Optimization Methodology

After the kinematic background introduced in Chapter 2, this chapter explains how the suspension synthesis task was translated into an optimization problem. The objective is not to describe the software line by line, but to define the logic behind the method: selected hardpoint coordinates are treated as design variables, the allowed geometry changes define the design space, and the resulting suspension responses are compared with engineering targets. The practical implementation of this methodology is then presented in Chapter 4.

3.1 From Manual Kinematic Synthesis to Numerical Optimization

The starting point of the methodology is a practical one. Suspension kinematic synthesis is normally an iterative task. A given hardpoint layout produces a certain camber, toe, roll centre and steering response. If the response is not satisfactory, the engineer modifies part of the geometry and checks the result again. The process continues until the geometry gives a compromise that is acceptable for the intended vehicle behaviour.

Hardpoint positions have a direct influence on this process because they define the geometry of the suspension links. Changing a hardpoint coordinate changes the motion of the wheel in bump, rebound and steering, and can therefore affect quantities such as camber gain, toe variation, roll centre height and steering-related responses. For this reason, hardpoint layout definition is one of the main tasks in suspension design [1], [2].

When the synthesis is carried out manually, the engineer explores the relationship between hardpoint movements and kinematic outputs by trial and error. This can be effective when the number of variables is small, or when the required correction is already clear from experience. The same approach becomes less practical when several hardpoints can be moved and more than one kinematic response has to be controlled at the same time.

The difficulty is mainly linked to the coupling between suspension responses. A movement that improves one quantity may also move another quantity away from its desired value. For example, a hardpoint displacement chosen to improve camber behaviour can also affect toe variation, roll centre height or steering-related quantities. The design task is therefore not a sequence of independent corrections, but a search for a compromise between connected objectives.

This type of problem is also reported in suspension optimization studies. Double wishbone suspension design is usually treated as a problem with several design variables, nonlinear kinematic behaviour and objectives that may conflict with each other [3]. In a similar way, Kinematics and Compliance (K&C) development often uses target values for specific suspension characteristics, with hardpoint positions acting as design variables inside a limited design environment [4].

In this thesis, the manual synthesis procedure is reformulated as a constrained numerical optimization problem. The engineer still defines the meaningful part of the problem: which coordinates can move, how far they can move, which targets should be pursued and which responses should be given priority. Once this has been defined, the optimizer searches inside the admissible design space for a hardpoint configuration that reduces the difference between computed responses and target values.

The optimization problem can be written in compact form as:

$$\min_{\mathbf{x}\in\Omega} J(\mathbf{x})$$

where $\mathbf{x}$ is the vector of active design variables, Ω is the admissible design space and $J(\mathbf{x})$ is the scalar objective function used to evaluate each candidate suspension geometry.

This expression is introduced only as the general mathematical form of the problem. The complete objective function, including weights and sensitivity factors, is described later in Chapter 4 together with the software implementation.

The manual trial-and-error process can then be expressed as an automated loop made of four main operations:

- definition of the hardpoint coordinates that can be modified;
- evaluation of the suspension kinematic responses for a candidate geometry;
- comparison between computed responses and target values;
- update of the candidate geometry in order to reduce the objective function.

The logic remains close to the manual workflow, but the search becomes more structured and repeatable. The engineer remains responsible for variables, bounds, targets and priorities, while the optimizer explores the resulting problem more efficiently and reduces the need for repeated manual trial-and-error iterations.

It is important to stress that this formulation does not remove engineering judgement from the design process. The result obtained by the optimizer depends on the choices made by the engineer, especially on targets, bounds and weighting strategy. The optimizer should therefore be considered as a support tool, able to search for improved geometries inside a controlled and physically meaningful design space.

3.2 Design Variables and Design Space

In the proposed methodology, the design variables are the hardpoint coordinates that the engineer includes in the optimization. Since the suspension geometry is defined by these points, changing their coordinates changes the kinematic behaviour of the mechanism. Previous studies on suspension design and optimization also treat hardpoint locations as key variables for modifying wheel alignment parameters and K&C characteristics [2], [4].

A relevant aspect of the methodology is that the complete suspension geometry is not modified freely. Only a selected subset of hardpoint coordinates is included in the optimization problem. This makes it possible to preserve the parts of the layout that must remain fixed for packaging or design reasons, while allowing the optimizer to work only on the coordinates that can realistically be adjusted.

The choice of active design variables is therefore part of the engineering formulation of the problem. Activating many coordinates gives the optimizer more freedom and can improve the chance of reaching the prescribed targets. At the same time, it increases the size of the problem and can produce geometry changes that are harder to interpret. A smaller set of active variables keeps the optimization more controlled, but it may limit the improvement that can be achieved.

For this reason, the design space must balance numerical freedom and engineering feasibility. The optimizer should explore enough geometry variation to improve the suspension responses, while remaining within a meaningful region for the suspension layout. In this work, the design space is therefore not only a mathematical domain, but also the set of admissible modifications to the initial geometry.

The initial suspension geometry plays a central role. It is the configuration from which the optimization starts, and it is also the reference used to define the admissible coordinate variations. The optimized geometry is therefore not generated from an arbitrary design, but obtained as a controlled modification of an existing suspension model.

This approach is consistent with the purpose of the framework. The aim is not to generate a new suspension architecture from zero, but to help the engineer improve or tune an existing kinematic model. The definition of design variables and bounds remains a user-driven choice, while the optimizer searches for the most suitable compromise inside the region defined by those choices.

3.3 Kinematic Targets and Objective Prioritization

The definition of kinematic targets is another essential part of the methodology. These targets describe the suspension behaviour that the optimized geometry should approach. They can refer to vertical responses, such as camber gain, toe variation and roll centre height, or to steering-related quantities, such as steering ratio and steer heave. These K&C and steering-related responses are commonly used to evaluate suspension behaviour and to guide suspension development [4].

Each target is linked to a specific output and to a specific operating condition. A camber or toe target, for example, can be assigned at a certain wheel travel value. A steering-related target can instead be assigned at a prescribed rack travel. In this way, the optimization does not act on a vague idea of suspension behaviour, but on selected points of the kinematic response that are relevant for the design objective.

The target values are not decided by the optimizer. They are defined by the engineer according to the desired suspension behaviour and to the purpose of the run. Sometimes the target represents an improvement over the initial configuration. In other cases, the target may be equal to the current value of a response, when that characteristic has to be preserved while other quantities are modified during the same optimization process.

This distinction is important. Suspension kinematic synthesis is not simply a matter of reducing every error to zero. Some responses may need to increase, others to decrease, and others to remain close to their starting value. The optimization problem must therefore reflect the engineering intention behind the selected targets, allowing the same framework to handle both improvement and preservation objectives.

When several responses are considered together, a prioritization strategy is needed. The outputs can have different physical meanings, different units and very different numerical magnitudes. A small toe-angle error may be important from a vehicle dynamics point of view, whereas a larger numerical change in roll centre height has to be interpreted on a different scale. The importance of a target cannot be judged only from its raw numerical error in absolute terms.

For this reason, the methodology uses weighting coefficients and sensitivity factors. These parameters define how much each output contributes to the overall objective function. They allow the engineer to give more importance to selected responses and to combine quantities expressed in different units or characterized by different numerical ranges.

The weighting strategy has a visible effect on the final optimized geometry. A high weight assigned to one response usually improves its agreement with the target, but it may also reduce the possibility of improving other coupled quantities. A low weight, on the other hand, may allow a larger deviation from that target if this helps the global objective. The choice of weights is therefore not just a numerical setting, but an engineering decision that defines the desired compromise.

This is particularly relevant for suspension design, where camber, toe, roll centre and steering-related quantities are all connected through the hardpoint layout. The optimized solution should therefore be interpreted as the best compromise found within the selected variables, targets, priorities and bounds, not as an absolute optimum independent of the formulation of the problem defined by the engineer.

3.4 Optimization as an Engineering Support Tool

The optimization methodology used in this thesis is intended to support the suspension design process, not to replace the engineer. The optimizer can explore the design space automatically and find hardpoint configurations that reduce the objective function, but the problem itself is still defined through engineering choices.

The engineer decides which hardpoint coordinates can be modified, which coordinates must remain fixed, the allowed range of variation for each active coordinate, the targets to be pursued and the relative importance of the selected responses. These choices define what the optimizer is allowed to explore and strongly influence the final result.

For this reason, the optimized solution should not be interpreted as a globally optimal suspension geometry in an absolute sense. It is the best compromise found for a specific formulation of the problem, with specific variables, bounds, targets and weights.

This point is important because many suspension design requirements are not fully represented by the objective function. Packaging limitations, structural constraints, manufacturability, component integration and vehicle-level performance can all impose additional limits on the final layout. The numerical result must therefore be checked critically before it can be considered acceptable from an engineering point of view.

The framework can be useful because it accelerates and organizes the exploration of possible hardpoint modifications. It allows the engineer to evaluate the effect of several geometry changes in a systematic way and to identify promising configurations more efficiently than with a purely manual trial-and-error approach.

At the same time, the design process remains interpretable. Since the design variables are physical hardpoint coordinates, the optimized solution can be examined in terms of actual geometry changes. It is possible to see which hardpoints have moved, in which direction and within which admissible range.

The final decision remains an engineering decision. The optimizer provides a candidate solution that satisfies the mathematical formulation of the problem, while the engineer must verify whether that solution is coherent with the suspension concept and with the practical constraints of the vehicle architecture. In this sense, numerical optimization acts as a decision-support tool inside the suspension development process.

3.5 Chapter Summary

This chapter presented the methodological basis of the optimization approach adopted in the thesis. Suspension kinematic synthesis was treated as an iterative engineering task that can be reformulated as a constrained numerical optimization problem.

The selected hardpoint coordinates represent the design variables, while the admissible design space defines the region in which the optimizer searches for improved geometries. Kinematic targets define the desired suspension behaviour, and their prioritization allows different responses to be balanced inside the same optimization process.

The chapter also clarified that the optimization result must be read as a compromise within the formulation chosen by the engineer. The optimizer does not replace engineering judgement; it supports the design process by making the exploration of hardpoint modifications more systematic and repeatable.

The next chapter describes how this methodology was implemented in the Python-based optimization framework, focusing on the software architecture, the interaction with the Excel-based kinematic models and the numerical optimization workflow.

Chapter 4 – Software Architecture and Implementation

The framework described in this chapter was built with a practical constraint in mind: the existing Excel tools had to remain part of the workflow. Instead of replacing the kinematic workbooks already used for suspension studies, Python is used around them to update hardpoints, read the outputs and drive the optimization loop. The calculation logic therefore stays inside the validated spreadsheets, while the repetitive manual part of the synthesis process is automated.

The resulting architecture is deliberately split into a few separate parts. The Configuration Workbook contains the information needed to describe the optimization case, including the suspension configuration, requested outputs, target values, weights, sensitivity factors and allowed hardpoint movements. The Python script reads these data, communicates with the workbooks, updates the geometry and passes the resulting objective value to the optimizer. Excel still evaluates the suspension kinematics; Python manages the sequence of operations.

Three use cases are currently covered. A rear suspension run only uses the vertical kinematic workbook. A front suspension run can be carried out either with the vertical workbook alone or with both the vertical and steering workbooks active. In the latter case, the script also calls the VBA procedure required by the steering model when the selected layout needs it during each optimization iteration.

The following sections describe the structure of the tool and the way information is exchanged between the Configuration Workbook, Python, the Excel models and the numerical optimizer throughout the optimization workflow.

4.1 Overall Software Architecture

The software architecture was organized by keeping the main tasks of the tool separate. Configuration, Excel communication and optimization are treated as different parts of the workflow, since each of them has its own role and its own possible sources of error. The Python application is the central element of the implementation. It does not replace the kinematic calculations already available in Excel. Its role is to connect the Configuration Workbook, the Excel-based models and the optimizer, so that the existing models can be used inside an automatic optimization loop.

Figure 3 shows the overall software architecture diagram, including the main framework components and the associated data flow. The input is provided by the Configuration Workbook, which contains the user-defined settings, optimization variables, bounds and targets. Suspension kinematics are assessed by the Excel models, while the Python script coordinates data exchange and feeds the resulting objective value to the numerical optimizer.

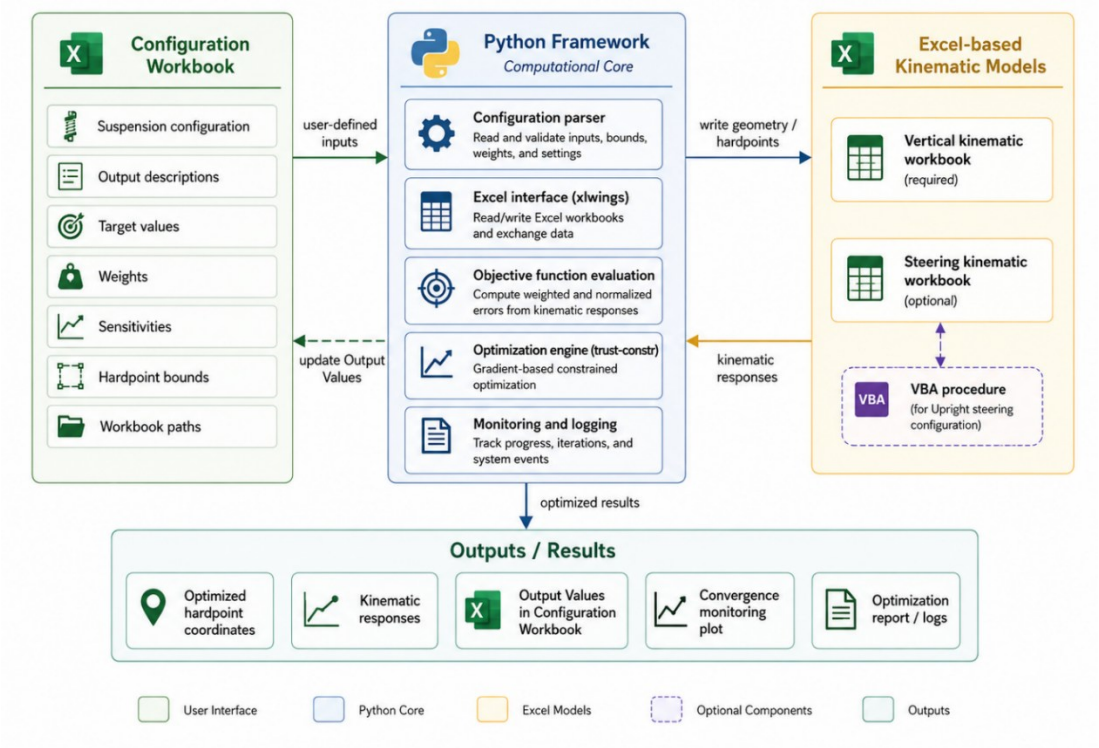

Figure 3 - Overall Software Architecture of the Optimization Framework

The workflow starts from the data entered in the Configuration Workbook. These settings define the selected mode, the workbooks to be used, the outputs to be monitored, the targets, the weights, the sensitivity factors and the admissible hardpoint bounds. Once these inputs have been read, the program initializes the optimization problem and opens the required Excel files.

During the run, Python writes each candidate hardpoint layout into the workbooks, lets the kinematic models calculate the corresponding responses and then reads the updated output values. Those values are used to compute the objective function, which guides the optimizer towards the next candidate geometry.

4.2 Configuration Workbook

The Configuration Workbook was introduced to give the engineer a direct way to prepare an optimization run. Instead of modifying the Python source code, the user fills in a set of structured tables in Excel, keeping the setup of the optimization case separate from the implementation of the tool.

This choice makes the tool easier to reuse. A new suspension case can usually be prepared by changing workbook paths, targets, weights or bounds, while the same Python implementation is kept unchanged.

In practice, the Configuration Workbook acts as the user interface of the framework. It stores the paths to the kinematic workbooks, the selected suspension configuration, the output descriptions, the target values, the weights, the sensitivity factors and the admissible hardpoint bounds for each run.

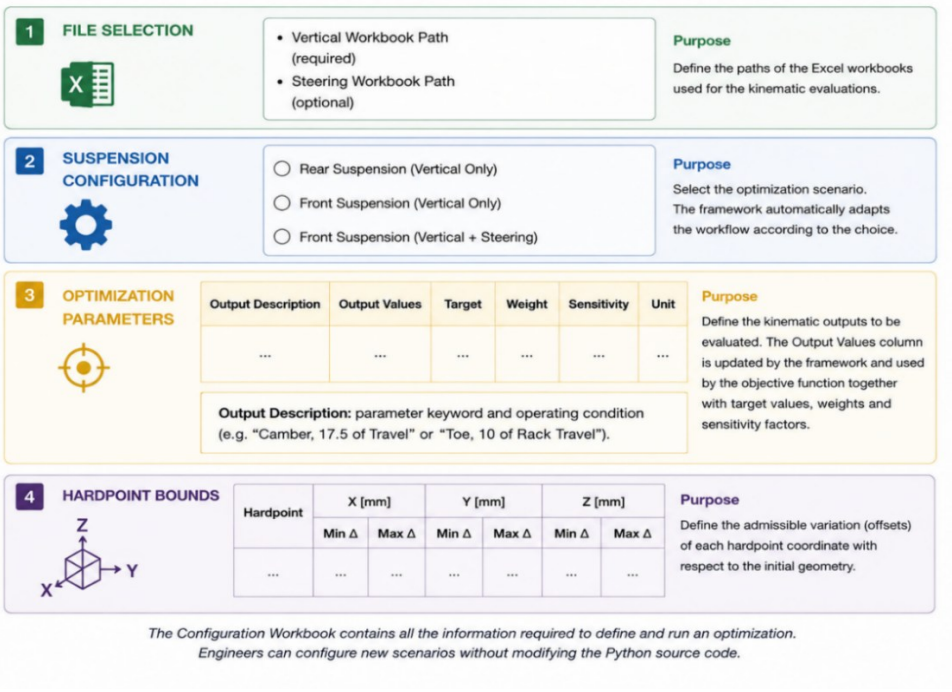

Figure 4 - Structure of the Configuration Workbook

Figure 4 shows the organization of the Configuration Workbook. The worksheet is divided into sections, and each section defines a specific part of the optimization setup. These sections collect the user inputs, the selected suspension model, the active design variables, the imposed bounds and the target quantities. The most relevant parts are described in the following subsections.

4.2.1 File Selection

The first section of the Configuration Workbook contains the paths of the Excel workbooks used during the optimization. These paths allow the Python script to identify the correct source files before the temporary working copies are created and used for the optimization process in a controlled and repeatable way.

Depending on the selected scenario, the program opens either one workbook or two workbooks. Rear suspension optimization and front vertical-only optimization require only the vertical kinematic model. When a combined front vertical and steering optimization is selected, both the vertical workbook and the steering workbook are required.

4.2.2 Suspension Configuration

The suspension configuration is selected through a dedicated field in the Configuration Workbook. The current implementation includes three modes:

- Rear Suspension (Vertical Only)
- Front Suspension (Vertical Only)
- Front Suspension (Vertical + Steering)

After this selection, the program adapts the evaluation procedure automatically. If the steering model is part of the run, the VBA routine associated with the steering workbook may be executed when required by the selected layout.

4.2.3 Definition of Optimization Inputs

The Configuration Workbook also contains the numerical inputs used to define the optimization case before the optimization run starts.

For each kinematic output included in the objective function, the user specifies:

- the kinematic model to which the output belongs;
- the output description, parameter name and the operating condition;
- the target value to be reached;
- the weighting coefficient used in the objective function.

Target values can be inserted as raw numbers or defined relative to the existing suspension setup. Relative targets are helpful if the intended change is easier to quantify as an adjustment to the original geometry instead of specifying the desired end value. As an example, a target could call for increasing/decreasing a kinematic response by some amount compared to the baseline model.

The same section contains the Output Values column. This column is updated automatically during the run and stores the current value of each selected response. The value may be read directly from the Excel output table or obtained by interpolation when the requested operating condition is not available as an exact table entry.

The final part of the setup defines how much each hardpoint coordinate is allowed to move. The user enters lower and upper offsets relative to the initial geometry. During initialization, the script converts these offsets into absolute bounds for the optimizer.

4.3 Excel Interface

The tool communicates directly with the Excel workbooks used for suspension kinematic analysis. This keeps the validated calculation procedures in the workflow and avoids reimplementing the kinematic models in Python.

The Python-Excel connection is handled through xlwings. This library allows the script to read from and write to workbook cells while preserving formulas, worksheets and VBA procedures [5]. In this implementation, xlwings is used both for the Configuration Workbook and for the kinematic workbooks.

During initialization, the script reads the settings from the Configuration Workbook and identifies where the required information is located in the selected Excel files. During each iteration, it updates the hardpoint coordinates, triggers recalculation and reads or interpolates the requested responses. The updated values are written back into the Output Values column and are then used by the objective function.

Figure 5 illustrates an overview of the dataflow between Python and the Excel models during optimization iterations. Python writes updated hardpoint locations to the workbooks, recalculates as necessary, then reads kinematic outputs. These are passed to calculate the objective function value.

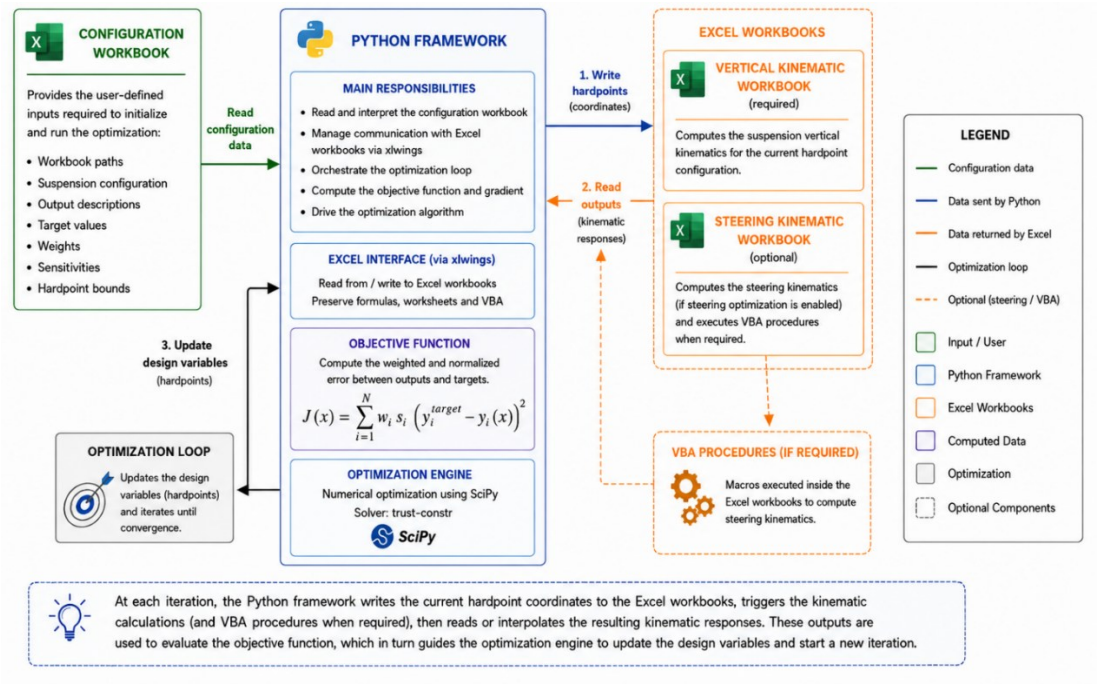

Figure 5 - Python-Excel Data Exchange

The following subsections describe the main data-exchange steps used by the program, from reading the initial geometry to retrieving the kinematic responses required by the optimizer during each optimization cycle.

4.3.1 Reading Initial Hardpoint Coordinates

The optimization starts from the initial suspension geometry contained in the selected Excel-based kinematic workbook. This geometry defines the reference configuration and provides the starting value of each design variable.

The geometry is saved as the three-dimensional coordinates of the suspension hardpoints. These coordinates define the baseline geometry for optimization. These values are stored in known locations within the workbooks. The xlwings library can automatically read their values, without copy-and-paste operations required by the user.

Once the initial geometry has been acquired, the program identifies the hardpoint coordinates that have been selected as active by the user. These coordinates are extracted from the complete suspension geometry, ordered consistently with the Configuration Workbook and collected into a design variable vector:

$$\mathbf{x} = (x_1, y_1, z_1, \dots, x_n, y_n, z_n)^T$$

Each entry of this vector corresponds to one Cartesian coordinate of a suspension hardpoint. The Configuration Workbook determines which coordinates are active and which ones remain fixed. Only the active coordinates are passed to the optimizer.

This is useful because a hardpoint does not always need to move in all three directions. For a given study, the engineer may decide to release only the coordinates that are physically acceptable or relevant to the design question.

4.3.2 Reading Optimization Targets and Settings

To make the tool adaptable to different workbook structures, the framework does not rely only on fixed output cell addresses. The Output Description entered by the user is interpreted as a parameter keyword combined with an operating condition.

The keyword is used to locate the required output column in the Excel-based kinematic model. For example, a camber-related request leads the program to search for the corresponding camber column. The operating condition then identifies the row, such as a wheel travel value for vertical kinematics or a rack travel value for steering kinematics.

Once the output column and the reference travel column have been found, the program looks for the requested operating value. If that value is present in the table, the response is read directly. If it is not present, the response is obtained by interpolation between the neighbouring table entries.

This value is recorded in the Output Values column of the Configuration Workbook upon completion of each workbook run. The Output Values column serves as the communication interface between the Excel models and the objective function, as it holds the responses which are assessed against the desired targets when optimizing.

This identification step is performed during initialization, before the optimization loop starts and before the first objective-function evaluation is carried out. The output locations and interpolation data are then stored and reused during the iterative process, so that the same search operation does not have to be repeated at every function evaluation.

4.3.3 Writing Updated Hardpoint Coordinates

After initialization, the optimizer supplies a new design vector at each iteration. These values correspond to the active Cartesian coordinates and must be written back into the suspension geometry before the models can be evaluated.

The active coordinates are identified from the hardpoint matrix in the Configuration Workbook. Non-empty cells define coordinates included in the optimization; empty cells correspond to coordinates that remain fixed. The same step also reads the lower and upper admissible variations used to build the bounds.

The original Excel workbooks are not edited directly. Before the run starts, timestamped copies of the selected workbooks are created, and all read/write operations are performed on those copies. This keeps the original models unchanged and also makes each optimization run traceable.

At each iteration, the current design vector is mapped back to the corresponding hardpoint coordinates. Only the active coordinates are overwritten in the working copies. The coordinates not selected by the user keep their initial values. Since the cell locations are already known from initialization, the program can access them directly during the loop.

If both the vertical and steering models are active, the same updated geometry is written to both workbooks at each function evaluation. Even though the two workbooks calculate different outputs, they represent the same physical suspension model, so their hardpoint coordinates must remain consistent throughout the optimization process.

Once the active coordinates have been written, the Excel workbooks contain the current candidate geometry generated by the optimizer and can be used to evaluate the corresponding kinematic responses for that specific iteration. The original formulas and calculation sheets of the engineering models are therefore preserved, while only the selected hardpoint coordinates are updated by the Python script. After the workbook recalculation, the required output quantities are read from the predefined result cells and returned to the optimization routine.

4.3.4 Evaluation of the Kinematic Models

After the hardpoint coordinates have been written, the suspension response is evaluated. No separate kinematic solver is introduced in Python; the calculation is still carried out by the existing Excel-based models, using the same formulas and worksheet structure as in the original manual workflow.

Once the geometry is updated, the workbook recalculates the kinematic quantities using the formulas already present in the engineering model. Depending on the selected configuration, either only the vertical model is evaluated or both the vertical and steering models are used for the current iteration.

For steering analyses, the procedure depends on the steering layout in the workbook. With the Upright layout, a dedicated VBA procedure is executed after the hardpoint update, because the steering workbook requires this internal step before the outputs can be read. With the Lower Control Arm (LCA) layout, or in vertical-only analyses, this VBA step is not required.

It is needed because the wheel side of the push/pull rod could be linked on the LCA or on the Upright; if it is linked on the Upright, the Steering Excel Workbook needs to evaluate the position with the VBA.

Since the output locations and interpolation information are stored during initialization, the program can retrieve the updated responses at each iteration without repeating the search procedure described earlier.

The retrieved values are assembled into the response vector of the current suspension geometry. This vector is passed to the objective function, which evaluates the error with respect to the selected targets before the optimizer proposes the next step.

4.4 Optimization Engine

This section focuses on the numerical part of the tool. The previous sections described how information is organized and exchanged with Excel; the following subsections explain how the optimization problem is formulated and solved. This represent the core of the Python workflow, that provides the link between the software workflow and the mathematical structure used by the optimizer.

The optimization engine is built around a constrained nonlinear problem and a solver that updates the suspension geometry iteratively. The design variables, objective function, bounds, derivative strategy and stopping criteria are described below. Together, these elements define how each candidate geometry is generated, evaluated and accepted or rejected during the optimization process.

Figure 6 summarizes the optimization workflow. Starting from the user inputs and the initial geometry, the program updates the design variables, evaluates the Excel models, computes the objective function and checks whether convergence has been reached. If the convergence criteria are not satisfied, the optimizer continues the iterative search by proposing an updated set of design variables.

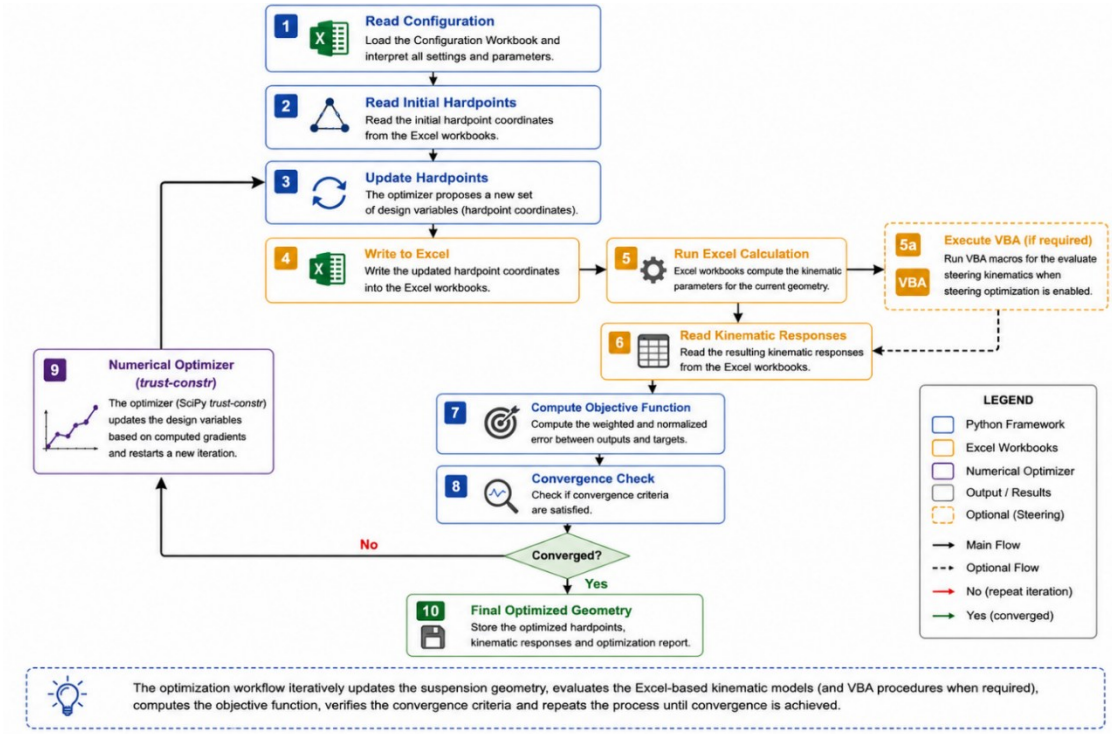

Figure 6 - Optimization Workflow

The main steps of this optimization process are described in the following subsections. Starting from the definition of the initial design vector, the workflow updates the selected hardpoint coordinates, evaluates the resulting kinematic responses through the Excel models and computes the objective function value. This sequence is repeated by the optimizer until a satisfactory compromise between the selected targets is reached.

4.4.1 Formulation of the Optimization Problem

In this implementation, suspension synthesis is formulated as a constrained nonlinear optimization problem. The aim is to find a set of design variables that reduces the difference between the responses calculated by the Excel models and the target values selected by the user for the selected optimization case.

The optimization problem can be written in the following general form:

$$\min_{\mathbf{x}\in\Omega} J(\mathbf{x})$$

where $\mathbf{x}$ is the design variable vector, Ω is the feasible design space defined by the imposed bounds, and $J(\mathbf{x})$ is the objective function to be minimized.

In most runs, the design vector is made of the Cartesian coordinates of the hardpoints selected in the Configuration Workbook. Since each coordinate can be activated separately, the number of design variables changes from one setup to another. Coordinates not selected by the user are kept fixed and are not part of the vector.

The same structure can also include additional variables if a specific model requires them. In steering optimization, for instance, the Reference Average Wheel Angle may be treated as one of the design variables.

Each design variable has a lower and an upper limit. These limits are not entered as final absolute coordinates. Instead, the Configuration Workbook stores the allowed variation from the initial geometry, and during initialization the script converts these values into the following absolute lower and upper bounds:

$$x_i^{\min} = x_i^0 + \Delta_i^{\min}$$

and

$$x_i^{\max} = x_i^0 + \Delta_i^{\max}$$

where x_i^0 is the initial value of the i-th design variable, while $\Delta_i^{\min}$ and $\Delta_i^{\max}$ are the lower and upper variations allowed by the user. These limits define the admissible search interval for each active coordinate and prevent the optimizer from exploring geometries that are outside the user-defined design space.

The initial geometry is used as the reference configuration for the optimization process. It represents the starting point from which the optimizer begins to modify the selected hardpoint coordinates. For this reason, the admissible range of each design variable is not defined as an absolute interval chosen independently from the vehicle geometry, but as a variation around its initial value. This approach makes the definition of the design space more intuitive for the user, since the allowed movements can be specified directly with respect to the original suspension layout. It also helps to keep the optimization process within a realistic region of the design space.

The resulting feasible region is:

$$\mathbf{x}^{\min} \leq \mathbf{x} \leq \mathbf{x}^{\max}$$

At each iteration, the optimizer updates the active variables while remaining inside these bounds. The resulting suspension geometry is then evaluated by the Excel-based kinematic models, and the computed responses are used to calculate the objective function.

4.4.2 Objective Function

The optimizer needs one scalar value to compare different candidate geometries. The objective function provides this value by measuring the mismatch between the computed kinematic responses and the target values entered by the user.

For each selected output, the error is calculated as the difference between the target and the response obtained for the current geometry. Since several outputs may be active at the same time, the individual errors are combined into a single objective value.

The implemented objective function is:

$$J(\mathbf{x}) = \sum_{i=1}^{N} w_i \, s_i \left(y_i^{target} - y_i(\mathbf{x}) \right)^2$$

Here, y_i^{target} is the target value of the i-th output and $y_i(\mathbf{x})$ is the corresponding response read or interpolated from the Excel-based model for the current design vector $\mathbf{x}$. The coefficient w_i is the user-defined weight, while s_i is the sensitivity factor used to scale the squared error. The value $y_i(\mathbf{x})$ is also written into the Output Values column of the Configuration Workbook. In this way, each term in the objective function can be traced back to an individual kinematic output.

Figure 7 explains how the objective function is calculated from kinematic outputs that are read from the Excel models. These responses are compared to target values that are defined in the Configuration Workbook. The difference between them is weighted by the priority set for each output. Each output's contribution is summed to create a scalar value that the optimizer uses to rank each candidate geometry.

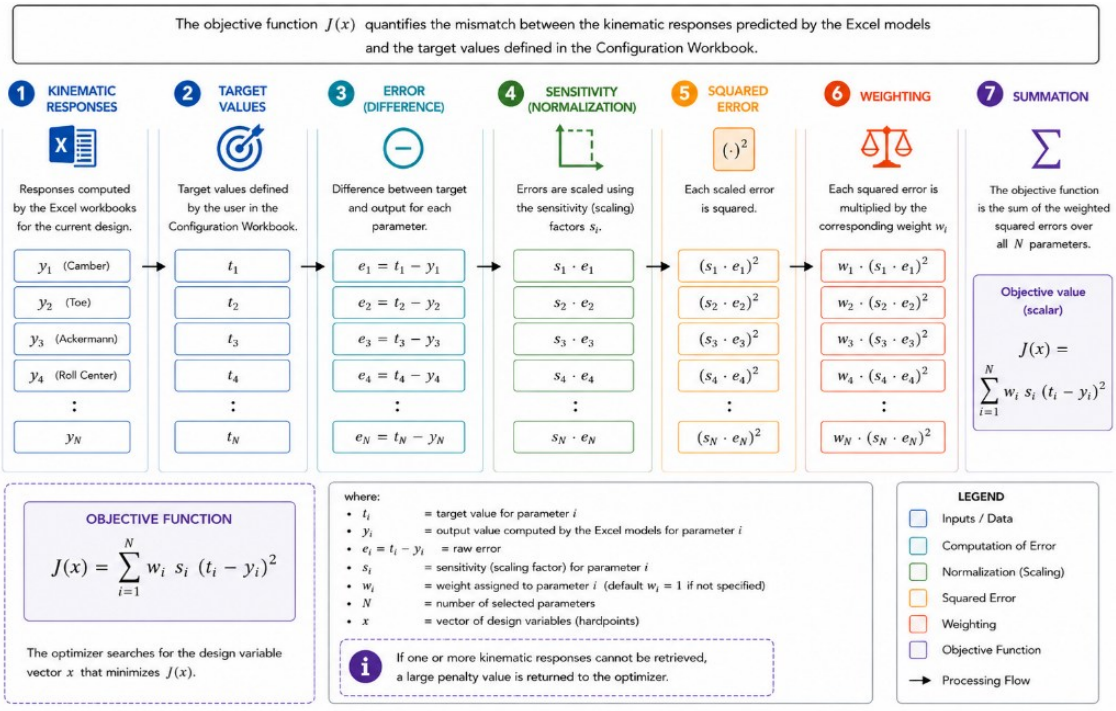

Figure 7 - Objective Function Computation

Using squared errors means that positive and negative deviations are treated in the same way, while larger errors receive a stronger penalty. As the computed responses move closer to their targets, the objective value decreases.

Weights are optional inputs defined by the user. If no value is entered for an output, the default weight is one. Increasing a weight gives that output more influence inside the objective function and therefore prioritizes that output target during optimization process.

Sensitivity factors scale responses that have different units or different numerical orders of magnitude. They are stored in a dedicated worksheet and are normally kept fixed during the use of the tool. Their purpose is to avoid a situation where one response dominates the objective function only because its numerical value is larger.

If a required response cannot be read or interpolated from the Excel files, the objective function returns a large penalty value. This makes invalid configurations unattractive to the optimizer and prevents incomplete output data from being treated as acceptable results. In this way, numerical errors or missing workbook outputs do not interrupt the optimization logic, but are converted into unfavourable candidate solutions.

During the run, a monitoring plot is also updated. It reports the absolute difference between each target and the corresponding response, so the user can follow how the selected outputs evolve during the optimization.

4.4.3 Selection of the Optimization Algorithm

The optimization problem is nonlinear and includes design variables bounds. The numerical solver therefore has to deal with constrained nonlinear optimization while also tolerating the fact that the objective function is evaluated through external Excel workbooks.

For this reason, the trust-constr algorithm available in SciPy was selected as the solver [6]. It supports constrained optimization and allows bounds to be imposed on the active variables. In the framework, every candidate vector proposed by the optimizer must respect the limits defined in the Configuration Workbook.

At each iteration, the updated geometry is evaluated through Excel. The resulting objective value, together with derivative information, is then used by the solver to choose the next candidate vector. The process continues until a convergence criterion is satisfied.

Because the objective function is evaluated by running the workbooks, no analytical gradient is available. During development, a custom central finite-difference gradient was implemented. For each design variable, the derivative can be approximated as:

$$\frac{\partial J(\mathbf{x})}{\partial x_i} \approx \frac{J(\mathbf{x} + \varepsilon \mathbf{e}_i) - J(\mathbf{x} - \varepsilon \mathbf{e}_i)}{2\varepsilon}$$

where ε is a small perturbation and $\mathbf{e}_i$ is the unit vector associated with the i-th variable. Only one component of the design vector is perturbed at a time, while all others remain unchanged. In the implementation, the perturbation size can be adjusted according to the norm of the current gradient to improve numerical stability, and reduce the sensitivity of the derivative estimate to numerical noise.

This custom gradient is expensive. It requires two additional objective-function evaluations for every active variable. Since a single evaluation already involves writing hardpoints to Excel, recalculating the models and reading the outputs, the central finite-difference approach can increase the execution time considerably.

For the final case studies, the finite-difference gradient provided by SciPy was used through the jac='2-point' option. This reduces the number of objective-function evaluations needed for gradient estimation and remains compatible with the black-box nature of the Excel-based models.

The Hessian required by trust-constr is approximated through a BFGS update. This allows the solver to use first-order derivative information without an analytical Hessian, which would not be available with the present Excel-based evaluation process.

The code was also prepared to test additional constraints through the SciPy NonlinearConstraint interface. In the final case studies, however, only the bounds on the active design variables were used. The auxiliary nonlinear-constraint functions were kept as development features for future testing.

4.4.4 Optimization Workflow

The optimization is carried out as an iterative loop. For each candidate geometry, the Excel models calculate the corresponding response and the optimizer uses the new objective value to decide the next step. The loop stops when one of the convergence criteria is met for the candidate geometry.

The process starts from the initial suspension geometry read from the selected Excel-based workbook. Based on the Configuration Workbook settings, only the coordinates selected by the user are included in the design vector. All remaining coordinates stay fixed.

At each iteration, the optimizer proposes a new candidate vector. The values are mapped back to their hardpoint coordinates and written into the required workbook cells. If both vertical and steering models are active, the same geometry is written to both files at every iteration. If the steering model uses the Upright configuration, the VBA procedure is executed after the hardpoint update.

After the workbooks complete their calculations, the program reads or interpolates the updated kinematic responses from the stored output locations. These values are written into the Output Values column and assembled into the response vector for the current geometry. The objective function then compares this vector with the target values.

The monitoring plot gives feedback during the run by showing the absolute difference between each target and the corresponding response. This helps identify which outputs are already close to their targets and which ones still have larger errors.

At the end of the optimization, the final design vector returned by the solver is mapped back to the complete hardpoint matrix. The optimized geometry is written to the workbooks and stored in the Configuration Workbook. The final responses are recalculated so that the saved output values correspond to the optimized geometry.

The workflow therefore combines Excel data exchange, numerical optimization and engineering-model evaluation in a single process. The result is an automated way to test hardpoint configurations against several kinematic objectives without manually editing the Excel files during the run.

4.4.5 Convergence Criteria

The optimization stops when one of the solver termination conditions is reached. These criteria prevent the process from continuing when improvements become negligible or when the maximum allowed number of iterations has been reached.

The final implementation uses the stopping criteria provided by trust-constr in SciPy. The relevant settings are the gradient tolerance (gtol), the variable tolerance (xtol), the barrier tolerance (barrier_tol) and the maximum number of iterations (maxiter) [6].

Table 2 summarizes the convergence criteria used in the optimization procedure.

Table 2 - Convergence criteria used by the optimization procedure

Criterion	SciPy option	Purpose
Gradient tolerance	gtol	Detects stationary conditions of the objective function
Variable tolerance	xtol	Stops when design-variable changes are negligible
Barrier tolerance	barrier_tol	Controls constrained barrier-subproblem convergence
Maximum iterations	maxiter	Limits the computational effort

The gradient tolerance is used to detect a stationary condition of the objective function. In practical terms, it indicates that further changes in the design variables are unlikely to reduce the objective value significantly. The variable tolerance stops the optimization when consecutive updates of the design vector become very small. Since trust-constr uses an interior-point strategy for constrained optimization, the barrier tolerance controls the convergence of the barrier subproblem associated with the imposed constraints.

A maximum iteration number is also imposed to avoid excessively long runs when the prescribed tolerances cannot be satisfied. Once the process terminates, the final design vector is interpreted as the optimized suspension geometry and is written back into the Excel workbooks and into the Configuration Workbook.

The convergence behaviour can be followed with the real-time monitoring plot developed in the framework. The plot shows the absolute error between each target and the corresponding response at every iteration, making it easier to see which outputs have approached their targets and which ones still require improvement.

A custom stopping callback was implemented during development for monitoring and testing purposes. The final case studies, however, rely on the stopping criteria provided directly by trust-constr, because this gives a more consistent termination strategy across the different scenarios without adding case-specific logic.

4.5 Chapter Summary

This chapter described the software architecture and implementation of the Python-based optimization framework. The implementation was designed to automate suspension kinematic synthesis while keeping the existing Excel-based models inside the workflow.

The Configuration Workbook was first described as the user interface of the tool. Through it, the user defines the optimization scenario, selects the Excel workbooks, specifies outputs and targets, assigns weights and sensitivity factors and defines the admissible hardpoint variations before starting the optimization run.

The Python-Excel interaction was then presented. The framework reads the initial geometry, creates working copies of the selected workbooks, updates the active coordinates, runs the model evaluation and retrieves or interpolates the resulting kinematic responses at each function evaluation of the optimization loop.

The optimization engine was also introduced. The selected hardpoint coordinates form the design variable vector, while the admissible coordinate variations define the bounds of the problem. The objective function combines weighted and scaled squared errors between target values and computed responses.

The numerical strategy was then discussed. The framework uses the trust-constr algorithm available in SciPy, with bound constraints on the active variables and finite-difference derivative information. The same workflow can be used for vertical-only suspension kinematics and for combined vertical and steering-related optimization.

The subsequent chapter introduces a series of illustrative examples which are analyzed using the developed framework. It presents the associated kinematic changes, the sensitivity to weighting choices and the computational cost of the chosen optimizations in practical engineering applications and representative suspension design tasks.

Chapter 5 – Case Studies and Results

This chapter applies the developed optimization framework to a set of numerical case studies. The focus is on vertical suspension kinematics, on the effect of the weighting strategy, on the additional cost introduced by the steering model, and finally on a combined optimization in which vertical and steering-related responses are considered together.

5.1 Introduction

The framework described in the previous chapters is applied here to a series of suspension kinematic optimization cases. The aim is not only to show the final numerical results, but also to understand how the Python-Excel workflow behaves when selected hardpoint coordinates are changed and the resulting kinematic responses are computed through the existing Excel models.

All the cases are based on a front suspension model and are presented in a progressive order. The first runs use only the vertical kinematic workbook, so that the basic capability of the framework can be assessed on quantities such as camber gain, toe variation and roll centre height. These outputs were chosen because they are directly affected by hardpoint movements and are commonly considered during the kinematic synthesis of a racing suspension for this purpose.

The first case is deliberately set up with a performance-oriented weighting strategy. In this configuration the roll centre is monitored, but it is not penalized in the objective function. This allows the optimizer to focus mainly on the camber and toe targets and makes the effect of target prioritization easier to observe more clearly. A second vertical-only case is then analysed with a more balanced set of weights, where roll centre height is also included in the objective function. Comparing these two cases is useful for showing the compromise between improving camber and toe behaviour and keeping the roll centre in a more acceptable range in this case.

Once the vertical-only behaviour has been assessed, the steering kinematic workbook is added to the workflow. This step is mainly used to measure the computational impact of the steering model and of the associated VBA macro, which is needed for this steering configuration. A benchmark is therefore carried out by comparing a vertical-only optimization with an equivalent run in which the steering workbook is also evaluated. In this way, the additional cost due to the Excel/VBA steering calculation can be separated from the

cost of the numerical optimization itself and considered during the evaluation of the tool. This also helps to understand how much the steering model affects the total run time. For this reason, the benchmark is useful before moving to the combined optimization case.

The last case combines vertical and steering-related targets in the same objective function. In this setup, both the vertical suspension outputs and selected steering outputs are evaluated, and additional toe-link coordinates are activated as design variables. This case is used to check whether the framework can deal with coupled vertical and steering requirements inside a single optimization run.

The results reported in this chapter are not meant to define a final suspension design. They are instead used to evaluate the behaviour of the tool, the effect of the selected design variables, the influence of weights and sensitivity factors, and the computational limits introduced by the repeated interaction between Python, Excel and VBA.

5.2 Case Study Definition

The case study is based on the front suspension of a high-performance vehicle. The optimization is carried out by changing selected hardpoint coordinates and evaluating the corresponding kinematic responses with the Excel-based models integrated in the framework.

The initial suspension geometry is read from the selected Excel-based kinematic workbook and is used as the reference configuration for the optimization. The admissible design space is defined by means of relative coordinate variations specified in the Configuration Workbook. These variations are then converted by the framework into absolute lower and upper bounds with respect to the initial hardpoint coordinates.

The vertical targets were defined at representative wheel travel values inside the available travel range of the model. Camber and toe variation were evaluated at +10 mm and -10 mm of wheel travel, while roll centre height was evaluated at 0 mm. The selected outputs, together with the initial values, targets, weights and sensitivity factors, are listed in Table 3.

Table 3 - Vertical kinematic targets used in the case study

Output	Condition	Initial value	Target value	Weight	Sensitivity
Camber	+10 mm Travel	-0.1009 deg	-0.1500 deg	2	100
Camber	-10 mm Travel	0.0915 deg	0.1350 deg	2	100
Toe Out	+10 mm Travel	0.0233 deg	0.0100 deg	1	1000
Toe Out	-10 mm Travel	-0.0293 deg	-0.0100 deg	1	1000
Roll Centre	0 mm Travel	26.09 mm	26.09 mm	variable	10

The camber targets were chosen to increase camber gain in both bump and rebound with respect to the baseline suspension. The toe targets were chosen to reduce toe variation around the reference position. The roll centre target was kept equal to the initial value, so that the current configuration could be used as a reference. This made it possible to check how far camber and toe could be improved while still trying to limit excessive changes in roll centre height.

Different weighting strategies were then used across the case studies in order to give different priorities to the selected outputs. In particular, the roll centre weight was changed between the performance-oriented and the balanced vertical-only setups. Low weights were instead assigned to the steering-related outputs in the combined case. The complete weighting strategy is discussed in Section 5.4.

For the combined vertical and steering case, the same vertical targets were kept and two steering-related outputs were added. These were steering ratio and steer heave, both evaluated at +10 mm of rack travel. Their targets were set equal to the initial values, because the goal of this case was not to redesign the steering behaviour from zero, but to verify whether selected steering characteristics could be preserved while the vertical kinematic response was improved. This choice made the steering outputs act mainly as preservation targets within the combined optimization problem during the run.

The steering-related targets selected for the combined optimization case are listed in Table 4. These quantities are evaluated at prescribed rack travel conditions and are used to describe the steering response of the front suspension model. Including these targets makes it possible to extend the optimization from vertical wheel travel only to a case in which steering behaviour is also considered.

Table 4 - Steering kinematic targets used in the combined case

Output	Condition	Initial value	Target value	Weight	Sensitivity
Steering Ratio	+10 mm Rack Travel	15.25	15.25	0.01	100
Steer Heave	+10 mm Rack Travel	0.7322	0.7322	0.01	100

The active design variables were selected according to the objective of each case. For the vertical-only optimizations, the variables were chosen among the coordinates of the front suspension wishbone joints. In the performance-oriented setup, only the vertical coordinates of the lower wishbone front and rear joints were activated. In the balanced setup, the design space was enlarged by activating the lateral and vertical coordinates of the lower and upper wishbone front and rear joints.

The steering benchmark used the same active variables and the same bounds as the balanced vertical-only setup. This choice was made to compare the two runs without changing the vertical optimization problem. As a consequence, any increase in execution time could be mainly associated with the additional steering workbook evaluation and with the VBA macro execution.

In the combined vertical and steering case, the design space was extended further by activating the three Cartesian coordinates of the wheel-side toe-link joint. This was done because the toe link has a direct influence on steering-related kinematic quantities and therefore provides useful additional freedom when steering outputs are included in the objective function together with the vertical targets.

The admissible variations assigned to the active variables were expressed as relative offsets from the initial hardpoint coordinates. Table 5 summarizes the active hardpoint coordinates and the corresponding bounds used in the different optimization cases, showing which parts of the suspension geometry were allowed to change.

Table 5 - Active design variables and admissible variations used in the case studies

Optimization setup	Active hardpoint coordinates	Admissible variation
Performance-oriented vertical-only setup	Bottom Wishbone Front Joint, Z	0 mm to +12 mm
	Bottom Wishbone Rear Joint, Z	0 mm to +12 mm
Balanced vertical-only setup	Bottom Wishbone Front Joint, Y	±6 mm
	Bottom Wishbone Front Joint, Z	0 mm to +12 mm
	Bottom Wishbone Rear Joint, Y	±6 mm
	Bottom Wishbone Rear Joint, Z	0 mm to +12 mm
	Top Wishbone Front Joint, Y	±6 mm
	Top Wishbone Front Joint, Z	±12 mm
	Top Wishbone Rear Joint, Y	±6 mm
	Top Wishbone Rear Joint, Z	±12 mm
Vertical + steering benchmark	Same variables as balanced vertical-only setup	Same bounds as balanced vertical-only setup
Combined vertical and steering setup	Same variables as balanced vertical-only setup	Same bounds as balanced vertical-only setup
	Toe Link Joint – Wheel Side, X	±6 mm
	Toe Link Joint – Wheel Side, Y	±6 mm
	Toe Link Joint – Wheel Side, Z	±6 mm

The longitudinal coordinates of the wishbone hardpoints were kept fixed in all the optimization cases. This reduced the size of the design space and avoided unnecessary changes to the suspension layout. All hardpoint coordinates not listed in Table 5 were therefore kept constant during the optimization process.

The selected bounds were not intended to represent a complete packaging study. Their purpose was instead to define a controlled and physically meaningful design space for assessing the behaviour of the optimization framework. The optimizer was therefore allowed to explore only coordinate changes considered acceptable for the scope of these case studies.

With this setup, the performance-oriented vertical-only case involved two design variables. The balanced vertical-only case and the steering benchmark involved eight design variables, while the combined vertical and steering setup involved eleven. In the combined case, the objective function included seven kinematic outputs: five vertical outputs and two steering-related outputs in total.

This structure allows the framework to be assessed from three complementary points of view. First, the vertical-only cases show how the optimizer changes the suspension geometry when different target priorities are assigned. Second, the comparison with the steering workbook quantifies the additional computational cost introduced by the Excel/VBA steering model. Third, the combined vertical and steering case checks whether the framework can manage multiple kinematic models and different categories of outputs in the same optimization process.

5.3 Case 1 – Vertical-Only Optimization

The first case study deals only with the vertical kinematics of the front suspension. In this configuration the framework interacts only with the vertical kinematic workbook, while the steering workbook is not activated. Two weighting strategies are considered: a performance-oriented setup, where the roll centre contribution is not penalized, and a balanced setup, where a small roll centre weight is introduced to obtain a more realistic compromise between camber gain, toe variation and roll centre height.

5.3.1 Performance-Oriented Setup

The first optimization case used a performance-oriented weighting strategy. In this setup, only the camber and toe targets were effectively included in the objective function, while the roll centre term was not penalized. The purpose of the run was to check whether the framework could move the selected hardpoints so as to reduce the error on the main kinematic targets, without imposing any restriction on roll centre displacement.

Here performance- oriented does not mean that the resulting geometry corresponds to the globally best vehicle performance. It means that the optimization weights(cambers and toe targets) have priority over roll centre response which is only observed.

For this first test, the design space was deliberately kept small. Only the vertical coordinates of the lower wishbone front and rear joints were treated as design variables. The admissible range was defined relative to the initial geometry, and all the remaining hardpoint coordinates were fixed. This produced a simple and controlled case, useful for observing the behaviour of the optimization framework when only a limited number of variables is available to modify the geometry.

The roll centre target was kept equal to the value of the initial configuration, but its weight was set to zero. The roll centre response was therefore still recorded during the run, although it did not contribute to the objective function. This allowed the optimizer to concentrate on the camber and toe targets.

Table 6 gives the kinematic responses for the performance-oriented vertical-only case.

Table 6 - Results of the performance-oriented vertical-only optimization

Output	Condition	Initial value	Target value	Final value
Camber	+10 mm Travel	-0.1009 deg	-0.1500 deg	-0.1453 deg
Camber	-10 mm Travel	0.0915 deg	0.1350 deg	0.1361 deg
Toe Out	+10 mm Travel	0.0233 deg	0.0100 deg	0.0064 deg
Toe Out	-10 mm Travel	-0.0293 deg	-0.0100 deg	-0.0115 deg
Roll Centre	0 mm Travel	26.09 mm	26.09 mm	55.60 mm

The results indicate that the optimizer moved the camber and toe responses close to the prescribed targets. At +10 mm of wheel travel, camber changed from -0.1009 deg to -0.1453 deg, approaching the target of -0.1500 deg. At -10 mm of wheel travel, it changed from 0.0915 deg to 0.1361 deg, practically reaching the target of 0.1350 deg.

The toe responses also improved with respect to the initial configuration. At +10 mm of wheel travel, toe-out was reduced from 0.0233 deg to 0.0064 deg. At -10 mm, it moved from -0.0293 deg to -0.0115 deg. The final value at +10 mm slightly overshoots the target by a small margine, but the overall toe variation around the reference position is clearly reduced over the analyzed range.

The main drawback of this result is the roll centre displacement. Roll centre height increased from 26.09 mm to 55.60 mm. This large change is a direct consequence of the selected weighting strategy. Since the roll centre term was not active in the objective function, the optimizer was free to move the selected hardpoints in the direction that improved camber and toe, even if this produced an undesirable roll centre change.

The final objective function value was approximately 0.0200. The run required 93 function evaluations, with a solver execution time of about 55.7 s and a total execution time of about 89.8 s. These results show that the framework can reduce the objective function and identify a geometry that satisfies the prioritized targets. At the same time, the case also shows that leaving an important kinematic quantity out of the objective

function can produce a solution that works well for some outputs but is less suitable when the suspension behaviour is considered more globally. For this reason, the result should not be interpreted only from the final objective value, but also from the complete set of kinematic responses. This confirms the need to choose the weights according to the desired engineering compromise for this specific design case.

5.3.2 Balanced Setup

The second vertical-only case was set up to obtain a more balanced compromise between camber gain, toe variation and roll centre height. In contrast with the previous case, the roll centre term was included in the objective function through a small non-zero weight. The intention was not to keep the roll centre exactly at its initial value, but to penalize excessive displacement while still allowing improvements in camber and toe.

The design space was also larger than in the performance-oriented setup. The active variables included the lateral and vertical coordinates of the lower and upper wishbone front and rear joints. Eight hardpoint coordinates were therefore optimized, while the remaining suspension coordinates were fixed. This allowed the optimizer to explore more general changes of the wishbone geometry.

The camber and toe targets were the same as in the previous case. The roll centre target was again set equal to the initial value, while its weight was set to 0.00015. This value was chosen after preliminary tests: larger values limited camber and toe improvement too strongly, while smaller values led to solutions closer to the performance-oriented case.

Table 7 reports the final responses obtained in the balanced vertical-only optimization.

Table 7 - Results of the balanced vertical-only optimization

Output	Condition	Initial value	Target value	Final value
Camber	+10 mm Travel	-0.1009 deg	-0.1500 deg	-0.1215 deg
Camber	-10 mm Travel	0.0915 deg	0.1350 deg	0.1116 deg
Toe Out	+10 mm Travel	0.0233 deg	0.0100 deg	0.0056 deg
Toe Out	-10 mm Travel	-0.0293 deg	-0.0100 deg	-0.0152 deg
Roll Centre	0 mm Travel	26.09 mm	26.09 mm	36.96 mm

Compared with the performance-oriented run, the final camber values are farther from the target. At +10 mm of wheel travel, camber reached -0.1215 deg instead of -0.1500 deg. At -10 mm, it reached 0.1116 deg instead of 0.1350 deg. This shows that the optimizer could no longer move the wishbone geometry only in the direction of camber improvement, because roll centre displacement was now penalized.

The toe responses stayed reasonably close to the desired range. Toe-out at +10 mm of wheel travel was reduced to 0.0056 deg, while the value at -10 mm reached -0.0152 deg. The targets were not matched exactly, but toe variation was still reduced significantly compared with the initial configuration.

The most evident difference is observed in the roll centre response. In the performance-oriented setup the roll centre height increased to 55.60 mm, whereas in the balanced setup it was limited to 36.96 mm. The small roll centre weight therefore reduced the roll centre displacement by about 18.6 mm compared with the performance-oriented solution.

The final objective function value was approximately 0.4956 for this setup. The optimization terminated successfully with the xtol condition after 43 iterations and 423 function evaluations. The solver execution time was about 253.5 s, while the total execution time was about 291.4 s.

The higher computational effort compared with the first case is associated with the larger design space and with the more constrained compromise introduced by the roll centre term. From an engineering point of view, however, the result is more meaningful because it accounts for the interaction between several kinematic quantities. The comparison between the two vertical-only cases highlights a central aspect of suspension kinematic synthesis: camber gain, toe variation and roll centre height are strongly coupled, so improving one group of responses may worsen another.

The balanced setup therefore shows how the weighting factors define the compromise searched by the optimizer. The solution is not simply a weaker version of the performance-oriented result, but a different geometry in which camber and toe improvements are moderated by the need to limit roll centre displacement in the final configuration.

5.4 Discussion of the Weighting Strategy

The weighting coefficients used in the different cases were selected according to the role assigned to each output in the specific analysis. Since the objective function combines quantities with different units and numerical ranges, the final contribution of each output depends on both the sensitivity factor and the weighting coefficient. In this work, sensitivity factors were used to scale the squared errors, while weights were used to set the relative priority of each target within a given case.

Table 8 summarizes the weighting coefficients used in the three main optimization setups discussed in this chapter. The values define the relative importance assigned to each target quantity within the objective function. They are therefore useful for comparing the different optimization strategies and for understanding how the final compromise was guided by the selected priorities.

Table 8 - Weighting coefficients used in the optimization cases

Output	Performance-oriented vertical setup	Balanced vertical setup	Combined vertical and steering setup
Camber, +10 mm Travel	2	2	2
Camber, -10 mm Travel	2	2	2
Toe Out, +10 mm Travel	1	1	1
Toe Out, -10 mm Travel	1	1	1
Roll Centre, 0 mm Travel	0	0.00015	0.00015
Steering Ratio, +10 mm Rack Travel	—	—	0.01
Steer Heave, +10 mm Rack Travel	—	—	0.01

The camber targets were given a higher weight than the toe targets because one of the main objectives of the vertical optimization was to increase camber gain in bump and rebound. Toe targets were still included with a non-zero weight, because reducing toe variation remained important, but they were assigned a slightly lower priority than camber.

The roll centre weight was changed between the preliminary and final vertical-only analyses to assess its influence on the optimized geometry. In the performance-oriented case it was set to zero, allowing the optimizer to focus almost entirely on camber and toe. This produced a large roll centre displacement. In the balanced setup, a small non-zero value was introduced to penalize excessive roll centre movement without preventing the optimizer from improving the other responses.

These preliminary runs were not meant to be a formal parametric study. The active variables were adjusted during the tuning process, so the runs are reported mainly to document the reasoning that led to the final weighting strategy. They should therefore be interpreted as exploratory tests, used to understand the effect of roll centre weighting before defining the final balanced setup.

Table 9 reports the final roll centre height obtained during the preliminary vertical-only runs with different roll centre weights. These results are useful to show how the weighting strategy changes the compromise found by the optimizer. As the roll centre weight is increased, the solution gives more importance to limiting roll centre displacement, even if this may reduce the freedom available for improving the other kinematic targets.

Table 9 - Influence of roll centre weighting on the optimized solution

Roll centre weight	Final roll centre height	Variation from initial value	Observed behaviour
0	55.60 mm	+29.51 mm	Camber and toe targets were approached closely, but roll centre displacement was not controlled
0.0001	40.4 mm	+14.39 mm	Roll centre displacement was reduced while still allowing significant improvement of the main targets
0.00015	36.96 mm	+10.87 mm	Selected compromise between camber, toe and roll centre preservation
0.0003	32.55 mm	+6.467 mm	Roll centre was more strongly constrained, but camber improvement became more limited

The results show a clear influence of the roll centre weight on the compromise found by the optimizer. With a zero roll centre weight, the optimizer can reduce the camber and toe errors without any penalty on roll centre displacement. Increasing this weight progressively reduces the final roll centre movement, but it also limits how far the camber responses can move toward their targets.

For the balanced vertical-only setup, the value 0.00015 was selected. It does not force the roll centre to remain exactly at the initial height, but it is sufficient to avoid the excessive displacement obtained in the performance-oriented case. The selected value therefore represents a practical compromise between roll centre preservation and meaningful camber and toe improvement.

In the combined vertical and steering setup, the same vertical weights used in the balanced case were retained. Two steering-related outputs, steering ratio and steer heave, were then added to the objective function. Their targets were set equal to the initial values, and both were assigned a low weight of 0.01. This reflects the purpose of the combined case: the steering behaviour was not being redesigned, but selected steering characteristics had to remain close to the baseline while the vertical kinematics were improved.

Objective function values obtained with different weighting configurations should not be compared in an absolute sense. Changing the weights changes the definition of the objective function itself. The comparison between cases should therefore be based mainly on the final kinematic behaviour and on the chosen target priorities, rather than only on the final scalar objective value.

5.5 Case 2 – Computational Cost of the Steering Model

After the vertical-only cases, the steering workbook was added to the workflow to evaluate its influence on computational cost. The purpose of this case was not to introduce new steering targets, but to isolate the effect of evaluating the steering model and running the corresponding VBA procedure.

For this reason, the same vertical targets and the same weighting strategy as in the balanced vertical-only setup were retained. The active design variables were also kept unchanged, so the optimization problem remained equivalent in terms of vertical objectives and design space. The only relevant change was the optimization mode: the framework now opened and updated both the vertical and steering workbooks at each objective-function evaluation.

In this configuration, the steering workbook was recalculated at every iteration, and the VBA macro associated with the Upright steering model was executed after each hardpoint update. No steering-related output was added to the objective function. Any increase in computational time can therefore be mainly attributed to the additional Excel/VBA operations required by the steering model, rather than to a larger objective function.

Table 10 compares the balanced vertical-only optimization with the equivalent run performed with the steering workbook activated. The two runs use the same vertical targets and the same active design variables, but the second case also evaluates the steering model at each function evaluation. This comparison makes it possible to isolate the additional computational cost introduced by the steering workbook and by the associated VBA calculation.

Table 10 - Computational impact of the steering workbook and VBA macro

Optimization setup	Number of outputs in the objective function	Number of design variables	Function evaluations	Solver execution time	Total execution time
Balanced Vertical-only	5	8	423	253.5 s	291.4 s
Vertical + steering workbook	5	8	423	1508 s	1559 s

The number of function evaluations was the same in the two runs. Both optimizations required 423 objective-function evaluations, meaning that the numerical optimizer followed an equivalent convergence process. Even so, activating the steering workbook increased the total execution time from about 291.4 s to 1559.0 s, corresponding to approximately 5.4 times the original value.

This corresponds to an increase of approximately 5.4 times in total execution time. The solver execution time increased from about 253.5 s to 1508.0 s, which corresponds to about 6.0 times. Since the number of function evaluations did not change, the increase is related to the higher cost of each individual objective-function evaluation.

The average total time per function evaluation increased from approximately 0.69 s/evaluation in the vertical-only case to about 3.69 s/evaluation when the steering workbook was active. This confirms that the dominant contribution to computational cost is not the numerical optimizer itself, but the repeated interaction with the steering workbook and the VBA macro required by the Upright steering model.

This benchmark is useful because it separates the effect of the optimizer from the effect of the external engineering models. The objective function still contained only vertical kinematic outputs, and the number of design variables was unchanged. The increase in execution time can therefore be attributed to the additional overhead introduced by the Excel/VBA steering evaluation, rather than to a more complex numerical problem.

This result highlights one of the main limitations of the current implementation. The framework can integrate the validated steering workbook into the optimization loop and preserve the existing workflow, but the repeated execution of the steering macro increases computational time significantly. Future improvements should therefore focus on reducing the cost of steering evaluation in future applications, for example by optimizing the VBA procedure, limiting unnecessary Excel recalculations or moving part of the steering kinematic calculations to Python.

5.6 Case 3 – Combined Vertical and Steering Optimization

The final case study includes both the vertical and steering kinematic models in the same optimization process. Unlike the benchmark case, steering-related outputs are now part of the objective function. The purpose was to check whether the framework could handle vertical suspension targets and steering targets at the same time, while also updating a larger set of suspension hardpoint coordinates.

The vertical targets were the same as in the balanced vertical-only setup. Camber gain, toe variation and roll centre height were therefore still considered in the objective function. Two steering-related outputs were then added: steering ratio and steer heave, both evaluated at +10 mm of rack travel. These steering targets were kept equal to the initial values, since the aim was to preserve selected steering characteristics while improving the vertical kinematic response.

The vertical outputs used the same weighting strategy as in the balanced vertical-only case. The steering-related outputs were assigned low weights, because their main role was to prevent excessive deviations from the initial steering behaviour. As discussed in Section 5.4, the steering ratio and steer heave weights were both set to 0.01.

The set of active design variables was extended with respect to the balanced vertical-only case. In addition to the lateral and vertical coordinates of the lower and upper wishbone front and rear joints, the three Cartesian coordinates of the wheel-side toe-link joint were activated. The resulting optimization problem therefore included eleven design variables and seven kinematic outputs in the objective function.

Table 11 reports the final kinematic responses after the optimized geometry was written back to the Excel workbooks and both the vertical and steering models were recalculated. This makes it possible to compare the optimized outputs with the initial values and with the targets used in the combined case. In this way, the table shows not only the improvement of the vertical responses, but also how the selected steering quantities were preserved during the optimization.

Table 11 - Results of the combined vertical and steering optimization

Output	Condition	Initial value	Target value	Final value
Camber	+10 mm Travel	-0.1009 deg	-0.1500 deg	-0.1287 deg
Camber	-10 mm Travel	0.0915 deg	0.1350 deg	0.1194 deg
Toe Out	+10 mm Travel	0.0233 deg	0.0100 deg	0.0083 deg
Toe Out	-10 mm Travel	-0.0293 deg	-0.0100 deg	-0.0139 deg
Roll Centre	0 mm Travel	26.09 mm	26.09 mm	35.62 mm
Steering Ratio	+10 mm Rack Travel	15.25	15.25	15.21
Steer Heave	+10 mm Rack Travel	0.7322 mm	0.7322 mm	0.7336 mm

The combined optimization improved the vertical kinematic behaviour while keeping the selected steering outputs close to the initial values. The camber values moved toward the targets, although not all the way to them. At +10 mm of wheel travel, camber changed from -0.1009 deg to -0.1287 deg. At -10 mm, it changed from 0.0915 deg to 0.1194 deg. These values are a clear improvement compared with the initial configuration, while still satisfying the additional roll centre and steering requirements included in the objective function.

The toe responses also improved. At +10 mm of wheel travel, toe-out moved from 0.0233 deg to 0.0083 deg, close to the target value of 0.0100 deg. At -10 mm, the final value was -0.0139 deg, still close to the target of -0.0100 deg and significantly lower than the initial toe variation.

Roll centre height increased from 26.09 mm to 35.62 mm. This variation is smaller than the one obtained in the performance-oriented setup and comparable with the balanced vertical-only result. This confirms that the roll centre penalty remained effective even when the steering model and the additional toe-link variables were included.

The steering-related outputs stayed close to their target values. Steering ratio changed from 15.25 to 15.21, while steer heave changed from 0.7322 to 0.7336. Since both targets were defined as current values, the optimizer was able to change the selected hardpoints without causing a significant alteration of the monitored steering behaviour.

The optimization terminated successfully with the xtol condition. The final objective function value was approximately 0.2953 for this combined case. The run required 32 iterations and 432 function evaluations. Solver execution time was about 1205 s, while total execution time was about 1261 s.

Table 12 compares the computational performance of the combined optimization with the steering benchmark introduced in the previous section. Both cases include the steering workbook, but the combined optimization also adds steering-related targets to the objective function and uses a larger set of active design variables. This comparison helps to understand how much of the computational cost is linked to the steering model itself and how much is related to the larger optimization problem.

Table 12 - Computational comparison between steering benchmark and combined optimization

Optimization setup	Number of outputs in the objective function	Number of design variables	Function evaluations	Solver execution time	Total execution time
Vertical + steering workbook	5	8	423	1508 s	1559 s
Combined vertical and steering optimization	7	11	432	1205 s	1261 s

The combined optimization did not produce a significant increase in computational cost compared with the steering benchmark. Although two steering outputs were added to the objective function and three toe-link coordinates were activated as design variables, the total execution time did not increase. It was actually lower than in the benchmark case, despite the slightly larger number of function evaluations.

This result supports the conclusion drawn in the previous section. The main contribution to computational cost is not the number of objective-function terms or the number of active design variables, but the repeated execution of the steering workbook calculations and of the associated VBA macro. Once the steering model is inside the optimization loop, the cost of each objective-function evaluation is mainly governed by

the Excel/VBA evaluation time. For this reason, reducing the cost of the steering calculation would have a direct effect on the efficiency of future combined optimization runs. This point becomes crucial if the framework is intended to be used on larger design spaces or for multiple optimization studies.

The average total time per function evaluation in the combined case was approximately 2.92 s/evaluation. This is much higher than in the vertical-only case, but still comparable with the steering benchmark. Therefore, adding steering targets and additional toe-link variables did not fundamentally change the computational behaviour of the framework. The main limiting factor remains the repeated interaction with the steering workbook for this test case.

Overall, this final case shows that the framework can manage a coupled vertical and steering optimization problem. The optimizer handled different types of kinematic outputs, different weight levels and an extended set of design variables in the same run. At the same time, the result confirms the need to select steering targets and weights carefully, especially when the objective is to preserve steering behaviour while modifying hardpoints to improve vertical kinematics.

5.7 Discussion of Results

The case studies highlight both the capabilities and the current limitations of the developed optimization framework. Overall, the Python-Excel tool was able to modify selected hardpoint coordinates automatically, update the corresponding Excel-based kinematic models and reduce the discrepancy between computed responses and target values.

The vertical-only cases showed that selected kinematic responses can be improved when the design space and the weighting strategy are defined appropriately. In the performance-oriented setup, camber and toe responses moved very close to their targets. However, since roll centre height was not penalized, the resulting geometry produced a large roll centre increase. This confirms that optimizing only a subset of the relevant outputs can lead to solutions that are effective locally, but not necessarily suitable when the whole suspension behaviour is considered.

The balanced vertical-only setup gave a more representative example of suspension kinematic synthesis. Introducing a non-zero roll centre weight allowed the optimizer to find a compromise between camber gain, toe variation and roll centre height. The final camber values were farther from the targets than in the performance-oriented case, but the roll centre displacement was much smaller. This result confirms the strong coupling between the selected kinematic quantities and shows how strongly the final geometry depends on the priorities assigned to the targets.

The weighting analysis showed that the roll centre weight plays a key role in the final optimized geometry. With a zero roll centre weight, the optimizer focused almost entirely on camber and toe. Increasing the roll centre weight reduced the final roll centre displacement, but also limited the camber improvement. The selected balanced value should therefore be seen as a practical compromise, not as an absolute optimum. This is important from an engineering point of view: the tool does not replace the designer's judgement, but provides a structured way to explore different design priorities.

The comparison between the vertical-only case and the equivalent steering-workbook case identified the main computational limitation of the current implementation. When the steering workbook and its VBA macro were included, execution time increased significantly even though the number of design variables and objective-function outputs remained unchanged. The computational cost is therefore dominated by repeated Excel/VBA steering evaluation rather than by the optimizer itself.

The final combined optimization confirmed the same conclusion. Two steering-related outputs and three additional toe-link coordinates were added, but total execution time did not increase with respect to the steering benchmark. Once the steering workbook is included in the loop, the main cost is associated with executing the steering model at each function evaluation. Additional objective terms and design variables have a secondary influence on total computational time.

From a kinematic point of view, the combined case showed that the framework can deal with vertical and steering outputs at the same time. The vertical responses improved with respect to the baseline, while steering ratio and steer heave remained close to their initial values. The tool can therefore be used not only to optimize one category of kinematic outputs, but also to preserve selected characteristics while other responses are modified.

Overall, the results suggest that the framework is suitable for supporting suspension kinematic synthesis. Its main advantage is the automation of a task that would otherwise require repeated manual hardpoint modifications and checking of the resulting kinematic responses. At the same time, the case studies also show that the optimized geometry depends strongly on the selected variables, target values, sensitivity factors and weights. The framework should therefore be considered as an engineering support tool for exploring the design space, not as an automatic generator of a definitive suspension layout.

The main limitation observed in the case studies is the computational cost associated with the Excel/VBA steering model. Future work should therefore focus on reducing the time required for each objective-function evaluation. Possible improvements include optimizing the VBA procedure, avoiding unnecessary workbook recalculations or progressively transferring part of the steering kinematic calculations from Excel/VBA to Python. This would allow the same optimization strategy to be used more efficiently on larger design spaces and more complex objective functions.

5.8 Chapter Summary

This chapter applied the developed optimization framework to a set of front suspension case studies. The analyses were arranged progressively: first vertical-only optimization, then optimization with the steering workbook active, and finally a combined vertical and steering optimization problem.

The vertical-only cases showed that the framework can modify selected hardpoint coordinates automatically and improve the prescribed responses. The performance-oriented setup showed that camber and toe targets can be approached effectively when roll centre height is not penalized. However, the resulting increase in roll centre height also showed why all relevant kinematic quantities should be included in the objective function when a more balanced suspension behaviour is required.

The balanced vertical-only setup introduced a non-zero roll centre weight and produced a compromise between camber gain, toe variation and roll centre height. The comparison between different roll centre weights confirmed that the final geometry depends strongly on the relative priority assigned to each output. In this sense, the weighting coefficients act as engineering tuning parameters, not just numerical constants.

The steering benchmark showed that the main limitation of the current implementation is the repeated evaluation of the Excel/VBA steering model. When the steering workbook and its VBA macro were included in the loop, execution time increased significantly even though no steering-related targets were added. This confirmed that the computational cost is dominated by external model evaluation rather than by the numerical optimizer itself.

The combined vertical and steering optimization demonstrated that the framework can handle multiple kinematic models and different categories of outputs within the same run. The optimizer improved the vertical responses while keeping the selected steering outputs close to their initial values. The addition of steering targets and extra toe-link variables did not significantly increase the computational cost compared with the steering benchmark, confirming that the VBA macro remains the main bottleneck.

Overall, the case studies confirmed the functionality and flexibility of the proposed Python-Excel optimization framework. The tool preserves the existing Excel-based engineering workflow while automating hardpoint modification and response evaluation. At the same time, the optimized solution is strongly influenced by design variables, targets, sensitivity factors and weights. For this reason, the framework should be seen as a support tool for suspension kinematic synthesis, useful for exploring different design compromises rather than replacing the engineering judgement required to define a final suspension layout in practical applications.

Chapter 6 – Conclusions and Future Developments

This chapter summarizes the main outcomes of the work and its application, discusses the limitations identified during the case studies, and outlines possible future developments of the proposed optimization framework. The aim is to highlight how the implemented workflow can support suspension kinematic development while also identifying the aspects that could be improved in future versions of the tool.

6.1 Conclusions

This thesis presented the development and application of a Python-based optimization framework for the kinematic synthesis of suspension hardpoints. The main objective of the work was to automate a process that is traditionally performed through repeated manual modifications of suspension geometry and subsequent evaluation of the resulting kinematic responses within the existing workflow.

The framework was developed with the specific aim of preserving the existing Excel-based engineering workflow. Instead of replacing the validated kinematic models already available within the adopted engineering workflow, the proposed tool interacts directly with them through an automated Python–Excel interface. In this way, the kinematic calculations remain performed by the original Excel workbooks, while Python manages the optimization process, updates the hardpoint coordinates, retrieves the computed responses and evaluates the objective function.

A dedicated Configuration Workbook was introduced as the user interface of the framework. This workbook allows the engineer to define the optimization setup without modifying the Python source code. Through this interface, the user can select the suspension configuration, specify the Excel workbooks involved, activate or exclude individual hardpoint coordinates, define admissible coordinate variations, assign target values, and set the corresponding weighting and sensitivity factors. This structure makes the framework flexible and reusable for different suspension optimization scenarios.

The optimization problem was formulated by considering the selected hardpoint coordinates as design variables. Each coordinate can be independently activated, allowing the design space to be adapted according to the specific requirements of the analysis. This makes it possible to keep fixed the coordinates that are constrained by packaging,

modelling assumptions or design choices. The objective function combines the squared errors between the target values and the corresponding kinematic responses, scaled through sensitivity factors and prioritized through weighting coefficients. Bound constraints are automatically generated from the admissible coordinate variations defined by the user in the Configuration Workbook. In this way, the optimization remains limited to a controlled and physically meaningful region around the initial suspension geometry.

The numerical optimization process was implemented using the trust-constr algorithm available in the SciPy optimization library [6]. The final implementation uses the numerical finite-difference gradient approximation provided by SciPy, while a custom central finite-difference gradient function was also developed as an alternative strategy. This choice preserved compatibility with the black-box nature of the Excel-based kinematic models, for which analytical derivatives are not available.

The developed framework was applied to a set of front suspension case studies. The vertical-only optimization cases demonstrated that the tool is able to automatically modify selected hardpoint coordinates and improve the prescribed kinematic responses. In the performance-oriented setup, camber and toe targets were approached effectively when the roll centre contribution was not penalized. However, the resulting roll centre displacement showed that optimizing only a subset of kinematic outputs can lead to solutions that are not necessarily balanced from a global suspension design perspective.

A more balanced vertical-only setup was then considered by introducing a non-zero roll centre weight. This case showed that the framework can identify a compromise between camber gain, toe variation and roll centre height. The results highlighted the strong coupling between the selected kinematic quantities and confirmed the importance of weighting factors in defining the final optimized geometry.

The steering workbook was subsequently introduced into the optimization workflow. A dedicated computational benchmark showed that the activation of the steering model and of the associated VBA macro significantly increases the execution time, even when no steering-related target is included in the objective function. This result demonstrated that the main computational limitation of the current implementation is associated with the repeated evaluation of the Excel/VBA steering model, rather than with the numerical optimizer itself in the analysed case studies.

Finally, a combined vertical and steering optimization case was performed. In this configuration, both vertical suspension outputs and steering-related outputs were included in the objective function, while additional toe-link coordinates were activated as design variables. The results showed that the framework can handle multiple kinematic models and different categories of outputs within the same optimization process. The vertical kinematic responses were improved with respect to the initial configuration, while the monitored steering outputs remained close to their reference values.

Overall, the results confirm that the proposed framework can support the suspension kinematic synthesis process by automating the exploration of different hardpoint configurations. The tool does not replace the engineering judgement required to define targets, design constraints and acceptable compromises, but it provides a systematic and repeatable method to evaluate the effect of hardpoint modifications. Its main contribution is the integration of numerical optimization into an existing Excel-based design workflow, reducing the need for manual trial-and-error iterations while preserving compatibility with validated engineering models.

6.2 Main Achievements

The work carried out in this thesis led to the development of a configurable optimization framework able to automate the suspension kinematic synthesis process while preserving the use of the existing Excel-based engineering models. The main achievement of the proposed approach is the integration of numerical optimization techniques into a workflow that was originally based on manual hardpoint modifications and repeated evaluation of kinematic responses.

A first important result is the development of a Python–Excel interface capable of reading, updating and evaluating suspension geometries automatically. The framework can retrieve the initial hardpoint coordinates from the selected kinematic workbook, modify only the coordinates activated by the user, write the updated geometry back into the Excel models and retrieve the resulting kinematic outputs. This automated data exchange makes it possible to perform complete optimization loops without requiring manual interaction with the workbooks during the iterative process.

Another relevant achievement is the introduction of the Configuration Workbook as a centralized user interface. Through this workbook, the user can define the complete optimization problem, including the model paths, the suspension configuration, the active design variables, the admissible coordinate variations, the target values, the weighting coefficients and the sensitivity factors. This allows different optimization scenarios to be set up without modifying the Python source code, improving the usability and flexibility of the tool for non-programming users.

The framework also allows each Cartesian coordinate of each hardpoint to be activated independently. This represents an important feature from an engineering point of view, because it enables the user to define a design space that reflects the actual design freedom available for a specific suspension system. Instead of moving complete hardpoints as rigid design variables, the engineer can decide which individual coordinates are allowed to vary and which coordinates must remain fixed.

The developed tool supports different optimization configurations. It can be used for vertical-only suspension optimization and can also manage combined vertical and steering analyses when the steering workbook is included. In the latter case, the framework is able to update both workbooks consistently and execute the required VBA procedure for the evaluation of the steering model. This confirms that the architecture is sufficiently flexible to handle multiple Excel-based kinematic models within the same optimization process during a single run.

The implementation also includes automatic handling of the optimization bounds. The admissible coordinate variations defined by the user as relative offsets are converted into absolute lower and upper limits with respect to the initial suspension geometry. This allows the optimization to remain constrained within a feasible design space while preserving the original suspension configuration as reference.

A further achievement is the automatic retrieval and interpolation of the selected kinematic responses. The framework identifies the output quantities required by the user, retrieves the corresponding values from the Excel tables and performs interpolation when the requested operating condition is not directly available. This makes it possible to define targets at specific wheel travel or rack travel values without requiring those exact values to be explicitly present in the workbook output tables.

The case studies confirmed the practical functionality of the developed framework. The tool was able to reduce the objective function, modify the selected hardpoint coordinates and generate optimized suspension geometries according to the assigned targets and weights. The results also showed that the framework can be used to investigate different design priorities, such as performance-oriented optimization or more balanced solutions involving camber gain, toe variation and roll centre height.

Overall, these achievements resulted in a reusable optimization environment that connects a numerical optimizer with existing Excel-based suspension models. This provides a practical engineering tool capable of accelerating the exploration of suspension hardpoint configurations while maintaining compatibility with the validated calculation procedures already used in the design workflow.

6.3 Limitations

Although the developed framework proved to be effective in automating the suspension kinematic optimization process, some limitations were identified during its implementation and application to the case studies.

The first limitation is related to the computational cost of the optimization process, especially when the steering workbook is included. The vertical-only optimization cases showed acceptable execution times, while the activation of the steering workbook and of the associated VBA macro produced a significant increase in total computational time. This behaviour is mainly due to the repeated interaction between Python, Excel and VBA at each objective-function evaluation. Therefore, the efficiency of the current implementation is strongly influenced by the execution time of the external Excel-based models.

A second limitation is the dependence on the structure and robustness of the existing Excel workbooks. Since the framework does not replace the kinematic models but interacts with them, the optimization process relies on the correct functioning of the workbook formulas, worksheet organization and VBA procedures. Any modification to the structure of the Excel models, such as changes in the location of hardpoint coordinates or output tables, may require corresponding updates in the Python interface or in the Configuration Workbook to preserve tool compatibility.

Another limitation concerns the selection of weighting and sensitivity factors. In the current implementation, these parameters are defined by the user according to engineering judgement and preliminary testing. The case studies showed that the final optimized geometry can change significantly depending on the relative priority assigned to each output. Therefore, the choice of weights and sensitivities plays an important role in the final solution, but it is not yet based on a fully automated or systematic calibration procedure.

The optimized result also depends strongly on the selected design variables and on the admissible bounds assigned to them. The framework can only search within the design space defined by the user. If the selected hardpoint coordinates do not provide sufficient freedom to satisfy the prescribed targets, or if the imposed bounds are too restrictive, the optimizer may only identify a partial compromise. For this reason, the definition of the active variables remains an important engineering decision.

In addition, the current implementation mainly considers kinematic targets evaluated at selected operating conditions, such as specific wheel travel or rack travel values. Although this approach is useful for controlling relevant points of the kinematic curves, it does not directly guarantee the behaviour of the complete response over the entire operating range unless additional target points are included. Extending the optimization to a larger number of operating conditions would improve the control of the full kinematic curves, but would also increase the size of the objective function and the computational effort.

A practical example of this limitation was observed in the vertical-only case studies. When the optimization was focused mainly on increasing camber gain and reducing toe variation, the roll centre height changed significantly if it was not included in the objective function with an appropriate weight. This result does not represent a failure of the optimization algorithm, but rather the consequence of the physical coupling between suspension kinematic quantities. Modifying hardpoint coordinates to improve camber gain and toe behaviour inevitably affects other suspension characteristics, such as the roll centre position. Therefore, the optimizer can only identify the best compromise within the imposed design space and according to the selected objective function; it cannot violate the physical relationships imposed by the suspension geometry. For this reason, the framework must be used consciously by an engineer, who remains responsible for defining meaningful targets, selecting appropriate design variables and interpreting the optimized solution from a complete vehicle-design perspective.

Finally, the framework should not be interpreted as an automatic generator of a definitive suspension layout. The tool is able to reduce the discrepancy between selected targets and computed responses within the imposed design space, but the final solution must still be evaluated from a broader engineering perspective. Packaging constraints, structural requirements, manufacturability, compliance effects and vehicle-level performance considerations are not fully included in the current optimization formulation. Therefore, the optimized geometry should be considered as a candidate solution to be further assessed and refined by the engineer.

6.4 Future Developments

Several future developments can be considered to improve the usability, efficiency and engineering completeness of the proposed optimization framework.

A first possible improvement concerns the reduction of the computational cost associated with the interaction between Python, Excel and VBA. The case studies showed that the execution time increases significantly when the steering workbook and the associated VBA macro are included in the optimization loop. For this reason, future work should focus on reducing the time required for each objective-function evaluation. This could be achieved by optimizing the VBA procedure, minimizing unnecessary workbook recalculations, improving the data exchange between Python and Excel, or reducing the number of operations performed at each iteration.

A second development could consist in progressively transferring part of the kinematic calculations from Excel/VBA to Python. In particular, the steering model represents the most critical part from a computational point of view, since the repeated execution of the VBA macro was identified as the main bottleneck of the current implementation. Reimplementing selected parts of the steering calculation directly in Python would reduce the dependency on Excel, improve execution speed and provide greater control over the numerical evaluation of the kinematic responses.

Another possible improvement concerns the deployment of the optimization framework as a standalone executable application. In the current implementation, the Python script is executed from a development environment, such as Spyder, which requires Python and the necessary libraries to be installed on the user's machine. Although this solution is suitable during development and testing, it may limit the usability of the tool in an industrial environment, especially for users who are not familiar with Python.

For this reason, the framework could be packaged into an executable file, allowing the optimization process to be launched without directly interacting with the Python source code. This would make the tool easier to distribute and use, while reducing the risk of accidental modifications to the implementation. The executable could be packaged together with the required Python environment and dependencies, so that the end user would only need to interact with the Configuration Workbook and the Excel-based kinematic models on a standard workstation.

A further improvement could consist in integrating the executable launch directly inside the Configuration Workbook. For example, a dedicated VBA macro could be added to the workbook in order to start the executable automatically. In this way, the engineer would define the optimization setup in the Configuration Workbook and then launch the complete optimization process through a button inside Excel, without opening Spyder or any other Python development environment. This would make the Configuration Workbook the complete user interface of the optimization process, while the executable would operate in the background as the computational core.

Future developments could also focus on a more systematic definition of weighting and sensitivity factors. In the current implementation, these parameters are selected by the user according to engineering judgement and preliminary tests. A possible improvement would be to define sensitivity factors based on admissible engineering errors or target tolerances for each kinematic quantity. This would make the objective-function contributions more consistent and reduce the empirical component involved in the tuning of the optimization setup.

The optimization could also be extended to consider a larger number of operating conditions. In the case studies presented in this work, the targets were defined at selected wheel travel and rack travel values. Future applications could include additional points along the travel range or even complete kinematic curves. This would allow the optimizer to control the overall shape of the camber, toe, roll centre and steering response curves, rather than focusing only on a limited number of operating conditions.

Finally, the optimized geometries could be validated through vehicle-level simulations. The current framework evaluates the suspension geometry from a kinematic point of view, but the final effect of the optimized hardpoints should also be assessed in terms of vehicle dynamics and performance. Future work could therefore include the export of the optimized geometries to a vehicle simulation environment, in order to evaluate their influence on handling behaviour, lap time or other performance indicators. This would create a stronger connection between suspension kinematic synthesis and complete vehicle performance assessment.

6.5 Final Remarks

The work presented in this thesis demonstrated that numerical optimization can be effectively integrated into an existing Excel-based suspension design workflow. By combining Python automation with validated kinematic workbooks, the developed framework provides a practical way to reduce manual trial-and-error iterations and to explore different suspension hardpoint configurations in a systematic and repeatable manner within a controlled design space.

The proposed tool demonstrated sufficient flexibility to manage different optimization scenarios, including vertical-only analyses and combined vertical and steering evaluations. The case studies showed that the framework can improve selected kinematic responses, preserve prescribed characteristics when appropriate weights are assigned, and highlight the trade-offs that naturally arise between coupled suspension quantities.

At the same time, the results confirmed that suspension kinematic synthesis cannot be reduced to a purely automatic numerical task. The final solution depends on the selected targets, design variables, bounds, sensitivity factors and weighting coefficients. Therefore, the role of the engineer remains essential in defining a meaningful optimization problem, interpreting the results and selecting the most appropriate compromise from a complete vehicle-design perspective.

Overall, the developed framework represents a useful support tool for suspension kinematic development. It preserves compatibility with the existing engineering models while introducing an automated optimization capability that can accelerate the design process and support more informed decision-making during the synthesis of suspension hardpoints within the existing design workflow.

References

[1] W. F. Milliken and D. L. Milliken, Race Car Vehicle Dynamics. Warrendale, PA: SAE International, 1995.

[2] X. Liu, M. Wang, X. Wang, C. Li, H. Guo, and J. Luo, “Hardpoint correlation analysis and optimal design for front suspension of a Formula SAE car,” Australian Journal of Mechanical Engineering, vol. 13, no. 2, pp. 67–76, 2015. DOI: 10.7158/M13-061.2015.13.2.

[3] A. A. G. Saravana Kumar and S. Bandyopadhyay, “Optimisation of Double Wishbone Suspension System using Multi-Objective Genetic Algorithm,” 2010.

[4] G. Ağakişi and F. Öztürk, “Kinematics & Compliance Validation of a Vehicle Suspension and Steering Kinematics Optimization Using Neural Networks,” Mechanika, vol. 29, no. 3, pp. 243–251, 2023. DOI: 10.5755/j02.mech.31983.

[5] xlwings, “xlwings Documentation.” Accessed: July 1, 2026. [Online]. Available: https://docs.xlwings.org/en/latest/

[6] SciPy Developers, “minimize(method='trust-constr') — SciPy Documentation.” Accessed: July 1, 2026. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/optimize.minimize-trustconstr.html

[7] C. Longhurst, “The Car Suspension Bible”, archived/republished online article. Available: http://www.mikedeff.in/double_wishbone.htm. Accessed: 7 July 2026.