Automated Target Cascade

vorgelegt von
M.Sc.
Martín O. Taboada

Von der Fakultät V
Verkehrs- und Maschinensysteme
der Technischen Universität Berlin
zur Verleihung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)

genehmigte Dissertation

Promotionsausschuss

Vorsitzender:	Prof. Dr. rer. nat. Volker Schindler
Erster Gutachter:	Prof. Dr. rer. nat. Wolfgang H. Müller
Zweiter externer Gutachter:	Dr.-Ing. Pascal Wohlenberg, ZF Lemförder

Tag der Aussprache: 25.10.2006

Berlin, 2006
D 83

PROMOTIONSAUSSCHUSS

Vorsitzender:	Prof. Dr. rer. nat. Volker Schindler Institut für Land- und Seeverkehr Leiter des Fachgebietes Technische Universität Berlin
Erster Gutachter:	Prof. Dr. rer. nat. Wolfgang H. Müller Lehrstuhl für Kontinuumsmechanik und Materialtheorie, Technische Universität Berlin
Zweiter externer Gutachter:	Dr.-Ing. Pascal Wohlenberg Virtuelle Tests und Simulation (F-DA2) Vorentwicklung und Fahrverhalten ZF Lemförder Fahrwerktechnik GmbH & CO. KG

Tag der Abgabe:	28.09.2006
Tag der Aussprache:	25.10.2006

To my mother,
who invested half of her life to get me here

and my friends,
who managed to make me feel accompanied, even at distance.

ACKNOWLEDGEMENTS

This thesis was carried out at the Fakultät V, Verkehrs- und Maschinensysteme der Technische Universität Berlin, Berlin, Germany; and at ZF Lemförder Fahrwerktechnik GmbH & CO. KG, Lemförde, Germany. It is the result of a work that would have proved intractable should I have faced it alone. Luckily, I have been accompanied and supported by many good people. It feels nice to have now the opportunity to express my gratitude for all of them. I would like to thank all the people who were involved in it, specially:

Felix Häusler, Team Leader of F-DA5, ZF Lemförder, my supervisor in the company, who trusted me and gave me the opportunity to be in this project. His supervision and guidance during the whole period, together with his enormous support, proved to be priceless.

Prof. Wolfgang H. Müller, my supervisor in the university, for his help and assistance through the period of this thesis. His comments, questions and suggestions made possible to really improve the work and my own skills to develop it.

Dr. Pascal Wohlenberg, F-DA2 Simulations, ZF Lemförder, who with his calculations on the last part of the thesis and his contributions both on my work and knowledge, enhanced the quality of the results. Even with the tight time constraints that I sometimes put on him, he always responded beyond my expectations. In short, somebody great to work with.

All ZF Lemförder staff in F-DA5, in particular to Dr. Simon Aramah and Knut Heidsieck, for their valuable help, tolerance and humorous comments. On top of this, Dr. Aramah's endless patience was indispensable at the time of working on my English grammar and spelling for the text of the thesis.

Last, but not least, to my mother. She is the foundation stone of whatever good I may ever accomplish.

Martín Taboada

Lemförde, October 2006

ABSTRACT

During the development of a suspension assembly for a road vehicle, often conflicting targets must be satisfied. Components, assemblies and, in the end, products are becoming increasingly complex, sophisticated and perfect. Cost, weight, reliability, safety, environment and even style constrain and put new challenges on the engineers. In particular, when focusing on the design of a suspension component, one of the most important factors is its weight.

For passenger cars, reducing the mass improves the acceleration of the vehicle and returns lower fuel consumption. At the same time, lighter components means less raw material used during production and eventually, depending on the processes involved, less waste. Therefore, when a suspension component that is already designed and in production goes through an optimisation process, if the design space and the hardpoints are kept, any mass that can be taken out of it without worsening its performance or changing to more expensive materials, methods, etc, can usually be translated to lower product cost.

Bearing this in mind, it looks appealing to try to optimise a suspension assembly by finding a way to reduce the weight of the components without falling in problems such as increased compliances that could affect the handling characteristics of the vehicle. The aim of this thesis is, then, to design and develop an optimisation method able to cope with existent suspension assemblies, their requirements and targets both at assembly and component level, and systematically look for the best way to weaken and lighten the components without compromising suspension performance.

The thesis focuses on a new approach based on the stiffness properties of each component by looking at its mathematical representation, the stiffness matrix. Once this entity is parameterised, a set of modifications is found by means of a genetic algorithm optimisation process. These modifications are tested on a virtual model and, if satisfactory from the suspensions performance point of view, used to guide the engineer in the redesign of lighter, more efficient components.

Keywords: optimisation, stiffness matrix, suspension assembly, weight, genetic algorithm

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

FOREWORD

In order to facilitate its reading, the present thesis is organised in fourteen chapters. Some of them are divided in sections which, when the scope of the topic broadens too much, are organised in paragraphs. The overall aim of the writing style is to walk through the same steps in which the actual work was performed, with the natural (and beneficial) omissions that could mislead and discourage the reader. As any other task in which the method, and not so much the results is what it is researched, the dead-end roads taken consumed quite a precious time that, in most cases, made no contribution to the final result, except for the lessons taught to the researcher.

In the first chapter, an introduction to the genesis of this thesis is presented, in order to outline the need that has to be fulfilled and the ideas behind the solution proposed. Basic notions of the current state of suspension technology are also discussed, together with the aspects that must be taken into account when assessing the performance of a suspension assembly. In the second chapter, the mathematical background that supports those ideas is thoroughly discussed. Although the concepts introduced therein are important to understand the whole process, it can be skipped by the reader not interested in each and every mathematical procedure.

Chapter 3 presents a first, general view of the process to implement in order to tackle the optimisation of a suspension assembly, based on the available software packages and the files used by them. The following two chapters introduce the reader with some of the optimisation algorithms in use nowadays: Gradient based methods, Response Surface Methods, and Evolutionary Search Strategies, explaining why the genetic algorithm is chosen for this thesis, and the particulars of its implementation. A brief introduction to the analogy with the biological counterpart is discussed, and the vocabulary introduced there is used through the rest of the thesis in order to explain the different effects that changes in the implementation might have on the results.

Given the choice of the genetic algorithm and the need of running time consuming simulations, Chapters 6 and 7 explain two ways to reduce the total time needed to run an optimisation case, first, by reducing the size of the problem by eliminating irrelevant parameters and, second, by approximating the results of the simulations with the help of a knowledge database.

The intricacies of a multi-body simulation and the way to obtain the necessary measurements out of it are discussed in Chapter 8. In particular the three cases used to analyse a suspension assembly are described and discussed in terms of their importance for a suspension performance: acceleration force, braking force, and lateral force. As the last part of the optimisation process stands the static analysis of

the optimised components, which is thoroughly explained in Chapter 9, together with the relationship between the different load cases applied to the components and the behaviour of the suspension assembly.

In Chapter 10 the developed software is introduced to the reader and every step executed is thoroughly explained, keeping track of the file manipulation, the inputs and outputs of each stage, the role of the user within the process, etc. Those stages in which, through dialog boxes, the user introduces data, chooses or filters results according to his criteria or the goals pursued are highlighted. In the following chapter, an optimisation case is run with the help of a suspension assembly used as a test mule, where the results are discussed in order to explain the different aspects of the process developed. The outcome of this process is the optimised component stiffness targets.

Once the virtual components are redesigned by the FE calculations department based on the new targets, they must be used in the suspension assembly in order to verify that the outcome is as expected. This is examined in Chapters 12 and 13.

The conclusions of the author are presented in Chapter 14. The results obtained in Chapters 11 and 13 are discussed, and the limitations of the procedure developed. Finally, in order to are dedicated to analyse the further improvements and lines of development to take, in order to perfect the optimisation process, or even widen its application field.

Obviously in a task comprising all the mentioned topics it would be impossible to develop each aspect of it in the time frame foreseen for this work. However, the topic researched, its nature and field of application (the automotive industry) implies that any substantial achievement in the area is not published, turning a literature review in a futile exercise. Hence, a reduced but thorough bibliography has been used, which is listed at the end of the thesis. Each of the books and web sites has been cited throughout the text in the respective chapters at least once. However, they are primarily mentioned when a particular paragraph was quoted with barely or no change, but they have been a source of information and inspiration for most of the chapters where they have been used, in particular Chapters 2 to 5, where the theoretical background in mathematics and optimisation algorithms is described. Most statements in Chapter 4 are based in one way of another on the sources mentioned along its text.

Finally, a comment regarding the pictures is necessary. In order to comply with confidentiality agreements, every suspension assembly shown in this thesis is made up of parts belonging to different assemblies. The objective is academical, so no actual assembly is present in the figures. The same can be said about the tables where target values for measurements, component weight, etc. are displayed. The values have been altered by different factors in order to preserve the real ones. Of course, in order to allow comparability and a systematic study of the results, there is consistency within and between the tables regarding the factors applied.

1 INTRODUCTION

1.1 Background

Like anywhere else in the modern industry developing a suspension assembly for a road vehicle is a complex and multidisciplinary task where multiple, often conflicting targets must be met. Components, assemblies and, in the end, products are becoming increasingly complex, sophisticated and (hopefully) perfect. Ever more demanding constraints regarding cost, weight, reliability, safety, environment and even style put new challenges on the engineers at the time of designing and developing new items. In a nutshell, the three most important considerations and the variables that affect them are:

- *Performance* is the ability of the design to fulfil its role, measured by the appropriate variables, provided that the components fit in the design space. In the case of a suspension assembly, targets regarding compliances, fatigue resistance, NVH[1], weight and energy absorption (crash behaviour) are among the most important.
- *Cost* is the amount of money to pay for the component or assembly. Materials and the processes (casting, machining, etc.) to build it, its complexity, production scale, development costs, production facilities location (which affects the taxes, labour costs, raw material, and so forth), logistics, etc. have to be factored in.
- *Manufacturability*: the design must be technologically feasible.

In particular, when focusing on the design of a suspension component, one of the most important factors is its weight. In the past, it was believed that the role of the suspension was to keep a minimum level of comfort, but nowadays it is known that vehicle dynamics and safety are also important.

Fundamentally, the sprung/unsprung mass ratio must be as high as possible, so that the suspension is more capable of keeping the wheel in contact with the road, especially during braking and turning. This means improving the dynamic characteristics and increasing safety since the limit lateral and longitudinal forces are raised. For a vehicle with a given mass, the unsprung mass has to be engineered to be kept as low as possible. In turn, the total mass is reduced improving the acceleration of the vehicle and returning lower fuel consumption figures. At the same time, a lighter component means less raw material used during its production and eventually, depending on the process, less waste. There

[1] *NVH: Noise, Vibration, Harshness*, is the term used in automotive industry that describes the quality of the drive in a vehicle. Reduced levels of NVH mean a higher comfort for the passengers and lower stresses on the components, allowing for increased life span.

fore, when a component that is already designed and in production goes through an optimisation process, if the design space and the hardpoints are kept, any mass that can be taken out of it without worsening its performance or changing to more expensive materials, methods, etc., can generally be translated to lower product cost.

Bearing this in mind, it looks appealing to try to optimise a suspension assembly by finding a way to reduce the weight of the components without falling in problems such as increased compliances that could affect the handling characteristics of the vehicle. The aim of this thesis is, then, to design and develop an optimisation method able to cope with existent suspension assemblies, their requirements and targets both at assembly and component level, and systematically modify them in order to obtain a weight reduction without compromising suspension performance.

1.2 Introductory case

In a mechanical assembly comprising several parts interconnected, one of the criteria to measure its performance is in terms of compliances. **Figure 1.1** shows a simple mechanism in which three levers, A, B and C, are interconnected by bushings b and c. In addition, levers A and C are attached to ground by means of bushings a and d, respectively. In this case, let the focus be on the compliance of point 4 with respect to point 1, measured as the rotation in degrees of point 4 when a given force is applied at the same point, keeping point 1 fixed. Such compliance is obviously dependent on the elastic properties of the three levers and the stiffness of the four bushings.

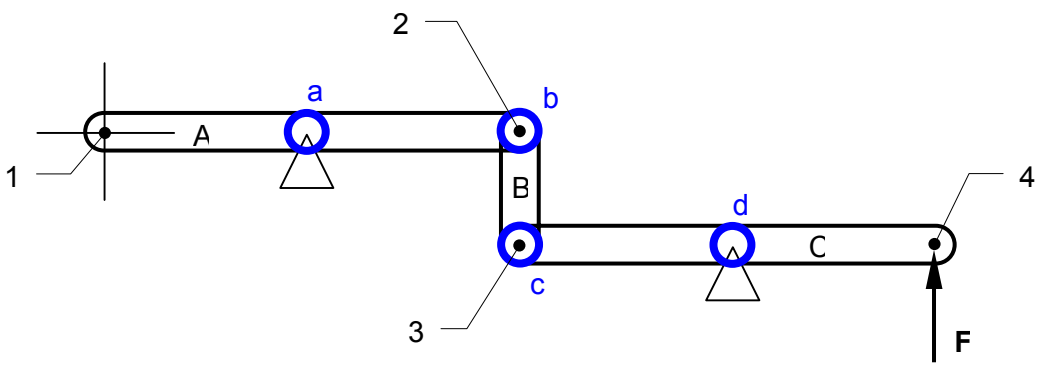


Figure 1.1 – Three levers mechanism

In this case there is a target for the mechanism that, under certain boundary conditions, the rotation of hardpoint 4 should not exceed a given value. The boundary conditions are:

- no displacement allowed in the vertical direction in point 1,
- no displacement of the bushings a and d,

- a force **F** in the vertical direction is applied at point 4.

Clearly, each of the three components has to have its own set of targets to ensure the performance of the mechanism as a whole. Lever A and C will have a given target for their stiffness in the bending mode, while lever B will have to fulfil a target for the compression/extension mode. In a typical design loop a first set of targets for each component is taken from previous designs or by educated guess of the designer. The challenge is, then, to envisage a systematic approach to optimise the design that both keeps the targets set for the mechanism and makes its components as light as possible. Perhaps there is a set of targets for the three levers that, even when it makes some of them weaker and more flexible in certain ways, it still fulfils the targets for the mechanism altogether.

1.3 Suspension assembly

A suspension assembly is assessed by its ability to cope with the task of placing the wheel in contact with the road as long as possible and with the best possible contact patch in every situation: turning, braking, accelerating, one-side/two-sides compression/extension. From a quick research in the literature it can be seen that there are an important number of possible suspension designs, from simple solid axles (as used in trucks) to complex multi-link layouts **[1.1]**. For the purposes of this thesis, it suffices to understand at least one of the most common and widely used suspensions in passenger cars, the *double wishbone* lay-out, shown in **Figure 1.2**. Each main component is labelled and the four attachment hardpoints to the subframe are numbered.

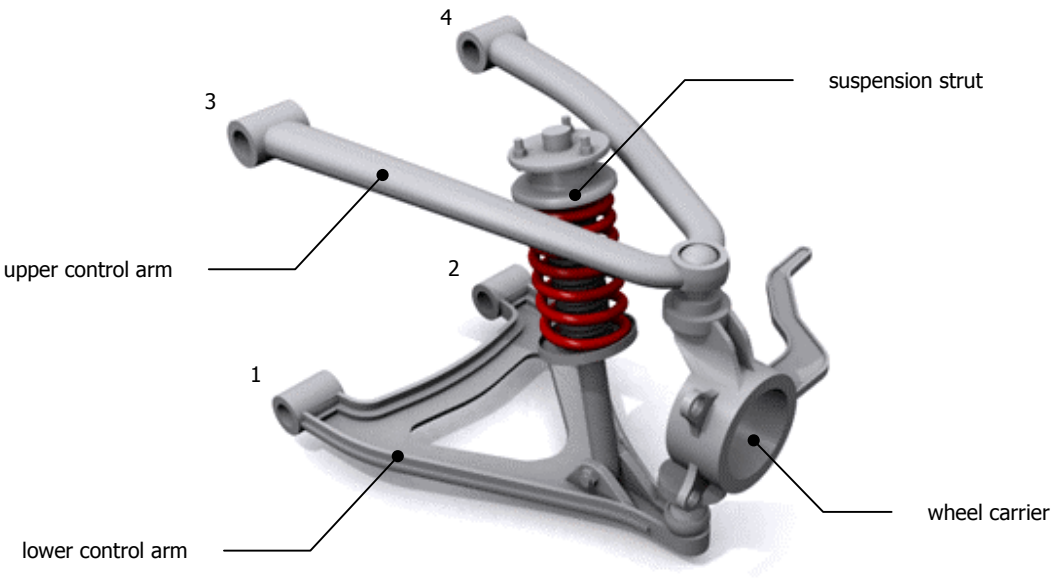


Figure 1.2 – Double wishbone suspension assembly

In this assembly, the wheel is supported by the wheelcarrier, which is attached to the lower and upper control arms by means of balljoints. The suspension is attached to the subframe of the vehicle by means of bushings, at hardpoints 1 to 4.

In a passenger vehicle the components in the suspension can be attached by several means, bushings and balljoints being the most common. **Figure 1.3** shows a control arm where the inner hardpoint is attached to the subframe by means of a bushing, while the outer hardpoint is attached to the upper part of the wheel carrier by means of a balljoint.

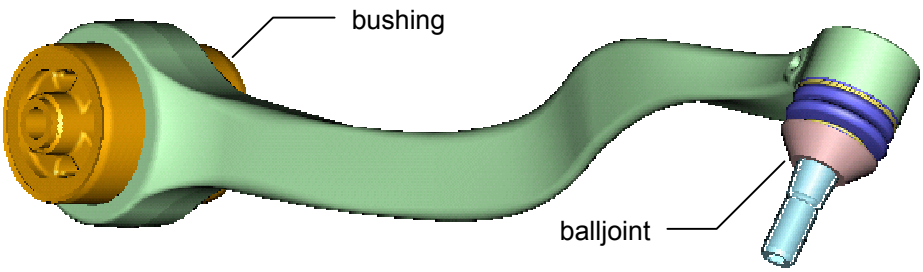


Figure 1.3 – Control arm

Balljoints consist of a sealed metal ball and a socket (**Figure 1.4**), usually with a cup made of plastic, in which the metal ball, attached to one arm, allows rotation in the three directions with respect to the socket, attached to the other arm or component.

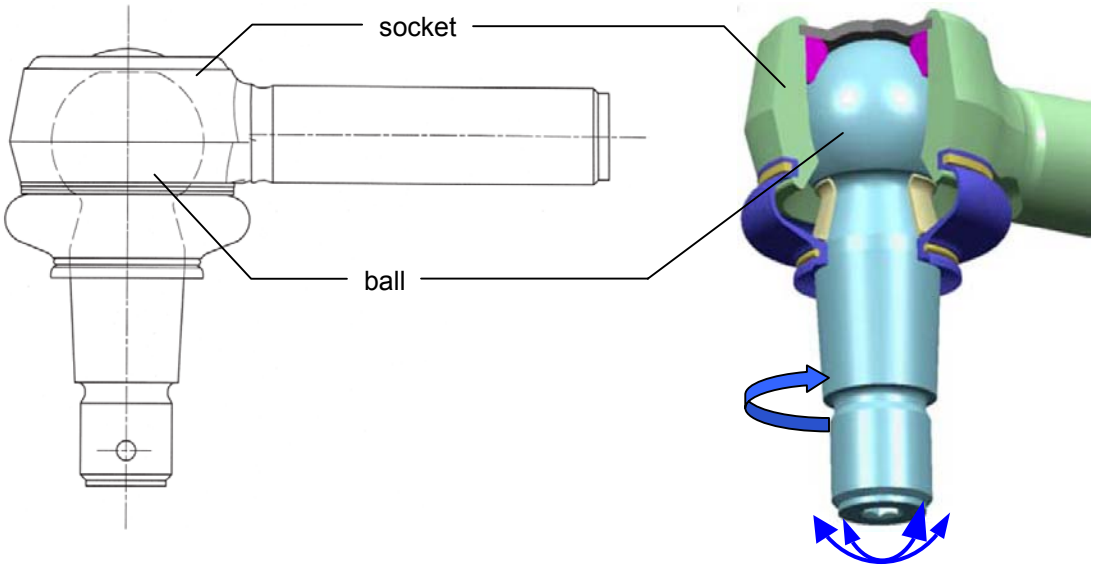


Figure 1.4 – Balljoint

In theory, balljoints do not transmit rotational loads between the two connected parts. However, there is in practice a small amount of forces being transmitted due to the friction between the ball and the plastic cup containing it, depending on the pre-load of the particular balljoint and its lubrication characteristics. Another factor is stiction (short for *st*atic fr*iction*), which is an increase of the static friction coefficient that happens after long periods in which the joint as not been moved.

Bushings are elastic components (**Figure 1.5**) that are necessary in order to add some degree of comfort to the vehicle. However, precisely because the material used to produce the component is not perfectly rigid, they also add same elasticity to the assembly. Typically, a bushing is five to ten times more elastic than the control arm it supports. These elasticities altogether are lumped in the so called compliance, which is the deflection of the suspension when a force is applied but the spring/damper unit is not compressed or extended. In particular, it is useful to consider the lateral and longitudinal forces applied at the contact patch (the area of the tyre's tread that is in actual contact with the ground) or at the wheel centre. These forces produce changes from the design position of the wheel with respect to the body of the vehicle and the road, so the compliance of a suspension is measured as the deviation (taken in angular degrees or in millimetres) per unit force.

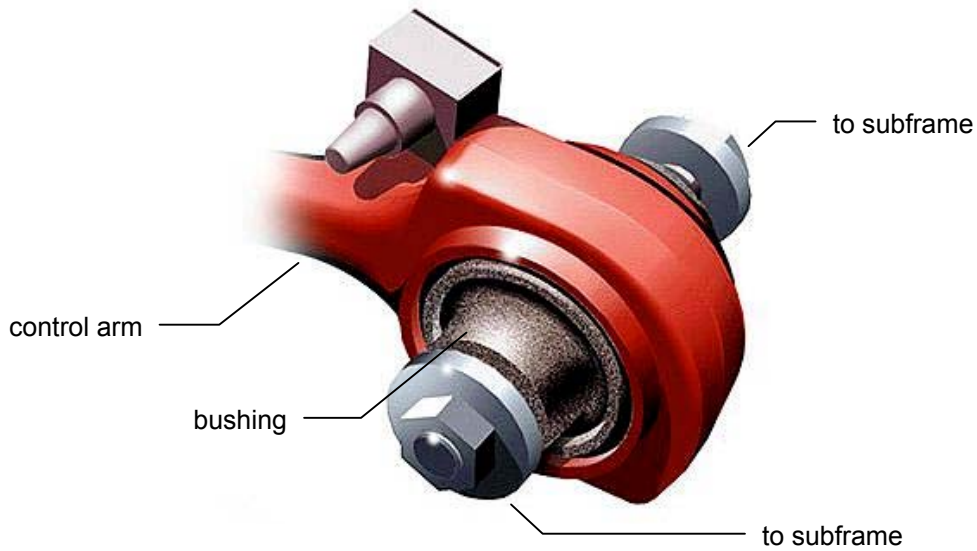


Figure 1.5 – Bushing

1.4 Definitions

There is a whole universe of terminology and variables related with the field of road vehicles suspension technology that is extensively used through out this thesis. Therefore, it is important and worth while to spend some time to familiarise with it. Although there exists an extensive literature about

most of the topics discussed in this chapter, from the perspective and needs of this thesis the basic definitions and a brief introduction to each concept are sufficient to start off.

Two of the most important variables to be measured in a suspension set up are camber angle and toe angle. Camber angle (**Figure 1.6**) is the angle the wheel plane makes with the vehicle's vertical axis. It is positive when the top of the wheel leans outward from the vehicle body. Toe angle (**Figure 1.7**) is the angle the wheel plane makes with the vehicle's longitudinal axis. When the front of the wheel leans outward from the vehicle body it is called toe out. A third variable to keep record is the wheel centre position, as it will be seen in the next section.

In summary, the most important compliances and their units are:

- *camber compliance* [°/kN] is the change in camber angle, measured in degrees, when a lateral force of 1 kN is applied at the contact patch.
- *toe compliance* [°/kN] is the change in toe angle, measured in degrees, when a force of 1 kN is applied at the contact patch, either longitudinally (positive = acceleration, negative = braking) or laterally (cornering).
- *lateral wheel centre compliance* [mm/kN] is the change in the position of the wheel centre in the lateral direction (change in track width), measured in millimetres, when a lateral force of 1 kN is applied at the contact patch.
- *longitudinal wheel centre compliance* [mm/kN] is the change in the position of the wheel centre in the longitudinal direction (change in base), measured in millimetres, when a longitudinal force (positive = acceleration, negative = braking) of 1 kN is applied at the contact patch.

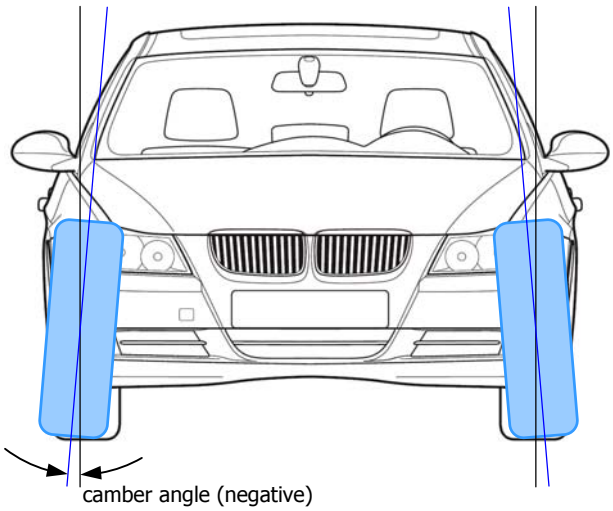


Figure 1.6 – Camber angle

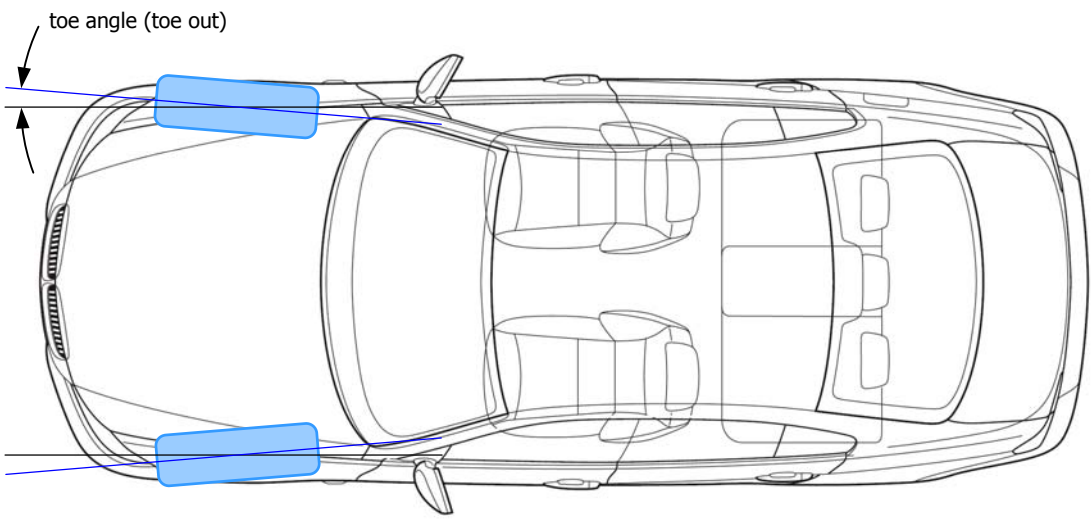


Figure 1.7 – Toe angle

1.5 Suspension parameters

Although the three major alignment parameters on a car are camber, toe, and caster, only the first two are of interest for this thesis. It is central to understand what camber and toe involve from the vehicle dynamics standpoint, why a particular setting is called for, and how it affects performance, which explains the tight requirements from manufacturers regarding design settings and compliances. Wheel centre compliance is another key tuning element for comfort and handling, and it will also be considered. In this section these concepts are only briefly discussed in order to grasp their significance, and not to strive in the search for the best solution for each aspect of the suspension design.

1.5.1 Camber angle

The lateral force that a tyre can develop is highly dependent on its angle relative to the road surface, so camber has a major effect on the road holding of a car. It is interesting to note that a tire develops its maximum cornering force at a small negative camber angle, typically around –0,5°. This fact is due to the contribution of camber thrust, which is an additional lateral force generated by elastic deformation as the tread rubber pulls through the contact patch **[1.2] [1.3]**.

Since the wheel is connected to the chassis by links which must rotate to allow for the wheel deflection, the wheel can be subject to large camber changes as the suspension moves up and down (**Figure 1.8**). Consequently, the more the wheel must deflect from its static position, the more difficult it is to maintain an ideal camber angle. The relatively large wheel travel and soft roll stiffness needed to provide a smooth ride in passenger cars during straight line travelling presents a difficult design challenge

since it has to be matched with the small wheel travel and high roll stiffness needed while taking a curve to reduce roll angle and increase safety and performance.

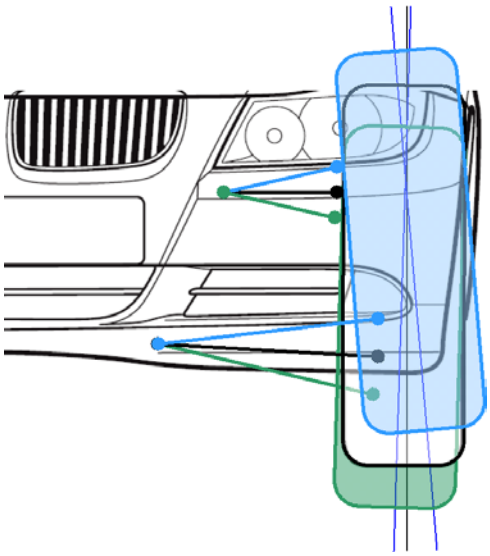


Figure 1.8 – Camber angle change due to wheel travel (front view)

It is important to draw the distinction between camber relative to the road, and camber relative to the chassis. To maintain the ideal camber relative to the road, the suspension must be designed so that wheel camber relative to the chassis becomes increasingly negative as the suspension deflects upward. If the suspension was designed so as to maintain no camber change relative to the chassis, then body roll would induce positive camber of the wheel relative to the road. Thus to negate the effect of body roll, the suspension must be designed so that it pulls in the top of the wheel (i.e., gains negative camber) as it is deflected upwards.

While maintaining the ideal camber angle throughout the suspension travel assures that the tyre is operating at peak efficiency, designers often configure the front suspensions of passenger cars so that the wheels gain positive camber as they are deflected upward. The purpose of such a design is to reduce the cornering power of the front end relative to the rear end, so that the car will understeer in steadily greater amounts up to the limit of adhesion. Understeer is inherently a much safer and more stable condition than oversteer, and thus is preferable for cars intended for the public.

1.5.2 Toe angle

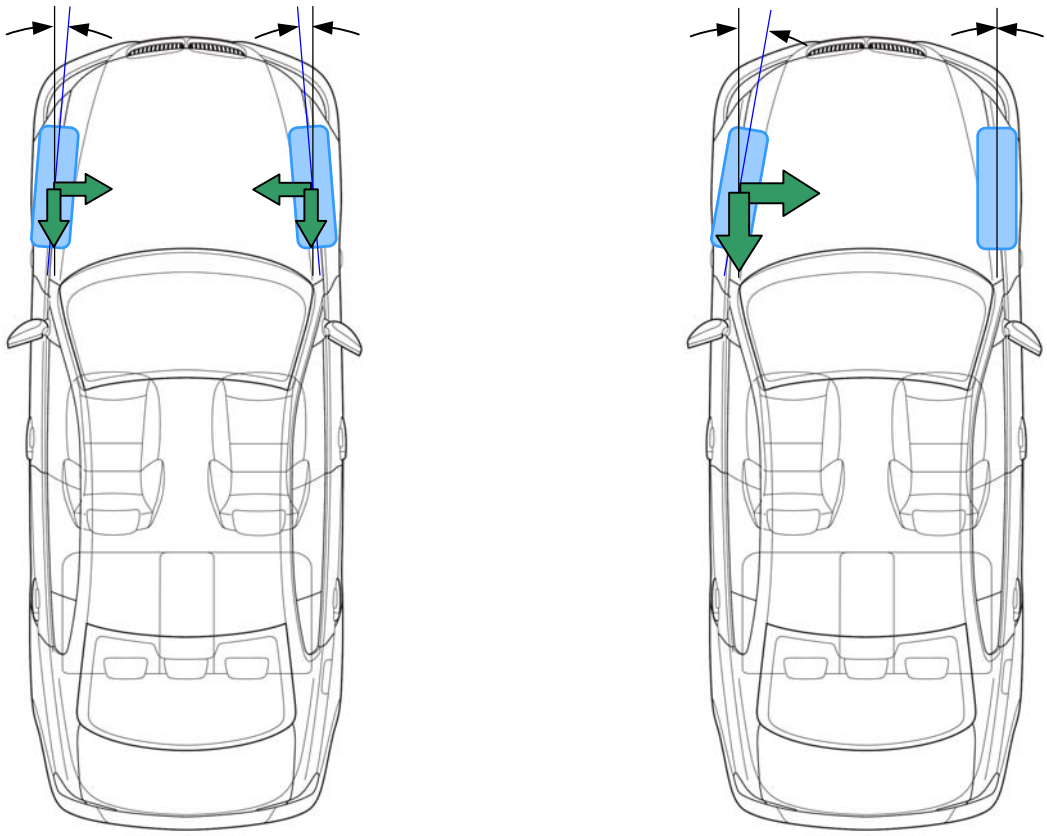
In theory, the wheels on a given axle of a car should point directly ahead when the vehicle is running in a straight line for minimum tyre wear and power loss due to scrub. Toe-in or toe-out causes the tyres to scrub and wear at the edges, outer or inner, respectively, since they are always turned relative

to the direction of travel. However, in reality zero toe is seldom wanted because its settings have a major impact on directional stability **[1.3]**.

With the steering wheel centred (**Figure 1.9.a**), toe-in causes the wheels to tend to roll along paths that intersect each other. Under this condition the wheels are at odds with each other developing lateral forces pointing inwards, so they mutually cancel and therefore no turn results. They also increase the rolling resistance since their rotation planes are at an angle with the direction of travel.

Now, when the wheel on one side of the car encounters a disturbance (on the right side in **Figure 1.9.b**), that wheel is pulled rearward about its steering axis. This action also pulls the other wheel in the same steering direction.

For a minor disturbance the right wheel steers only a little, perhaps so that it is rolling straight ahead instead of toed-in slightly. This eliminates the forces at the right wheel and increases the forces developed at the left wheel: the lateral force induces a torque to the right, but the longitudinal force a torque to the left, counteracting each other. Thus the wheels have absorbed the irregularity without significantly changing the direction of the vehicle. In this way, toe-in enhances straight-line stability.



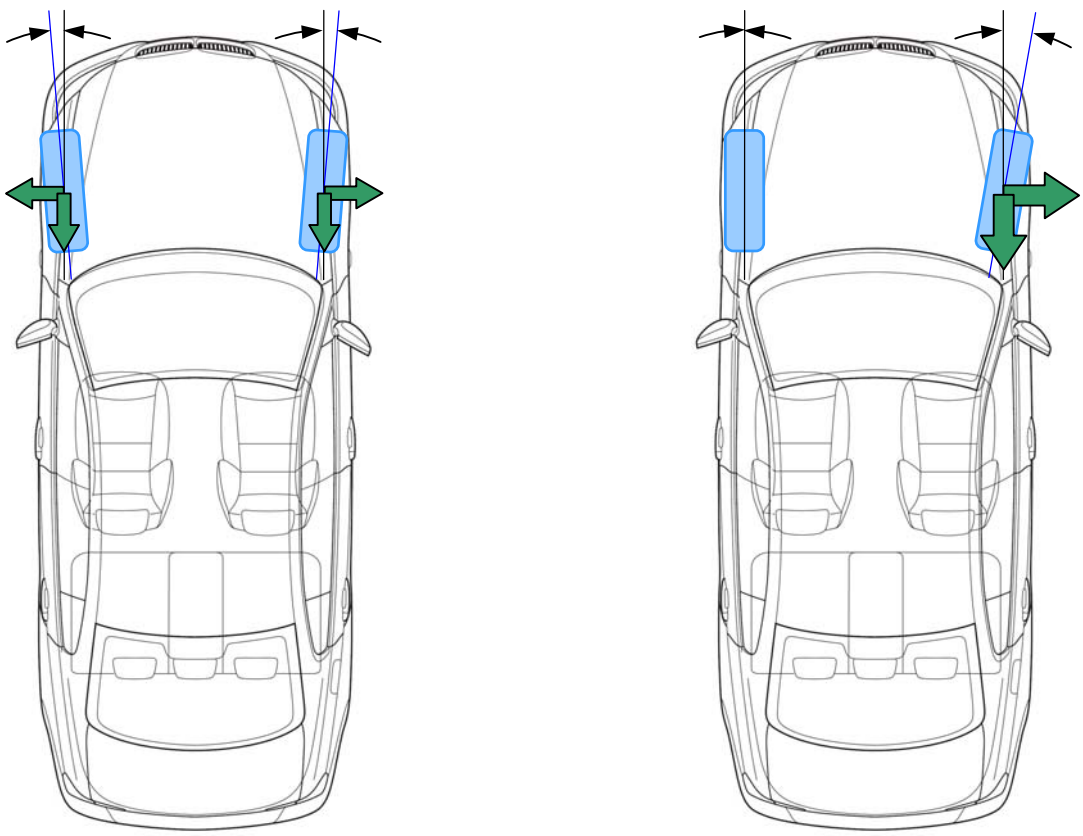
a **b**

Figure 1.9 – Analysis of the effect of toe-in in contact patch forces

On the other hand, with the steering wheel centred (**Figure 1.10.a**), toe-out causes the wheels to tend to roll along paths that diverge from each other. This time the wheels develop lateral forces pointing outwards and mutually cancelled, so no turn results. They also increase the rolling resistance since the rotation plane of each wheel is at an angle with the direction of travel. Now, a disturbance on the right side (**Figure 1.10.b**) pulls the right wheel rearward about its steering axis, rotating it to the right, and also pulls the other wheel in the same steering direction.

For a minor disturbance, the wheels will steer only a small amount, so that the left wheel it is rolling straight ahead instead of toed-out slightly, thus eliminating the forces at the left wheel, but increasing those at the right wheel: both lateral and longitudinal force induce a torque to the right, making the vehicle more prone to change direction. In this way, toe-out encourages directions changes.

Not only disturbances on the road have an effect on the toe settings. During breaking the wheels tend to "open" (increasing the toe-out or reducing the toe-in) due to the compliances in the suspension. During acceleration, for a rear wheel drive car, the front suspension will tend to toe-out and the back to toe-in, and vice versa on a front wheel drive car.



a **b**

Figure 1.10 – Analysis of the effect of toe-out in contact patch forces

1.5.3 Wheel centre compliances

Ideally a suspension system should be able to cope with the road, absorbing the irregularities instead of transmitting them to the vehicle body and its occupants. Proper compliances allow the wheels to move rearward slightly as they hit bumps or brake but do not allow them to move laterally during cornering, maintaining the handling characteristics. The solution is always a trade-off between vehicle dynamics requirements and comfort since it is very difficult to achieve a stiff suspension in the lateral direction while allowing for longitudinal elasticity. Every manufacturer has its own philosophy that determines the solution to adopt, which variables to prioritise and which ones to sacrifice. Vehicles with an orientation to comfort will allow for some detriment in the road holding characteristics in exchange for a smoother ride, while more sporty cars with higher dynamic limits will suffer from a harsher ride.

In most cases a solution for these two opposite targets is by the use of bushings with different compliances, depending on the direction of deformation. For example, in the transversal direction of travel they behave stiffer than in the longitudinal direction.

1.6 Design process of a suspension assembly

As shown in **Figure 1.11**, the development of a suspension assembly consists of five steps, which can be sorted out in system (assembly) level and component level:

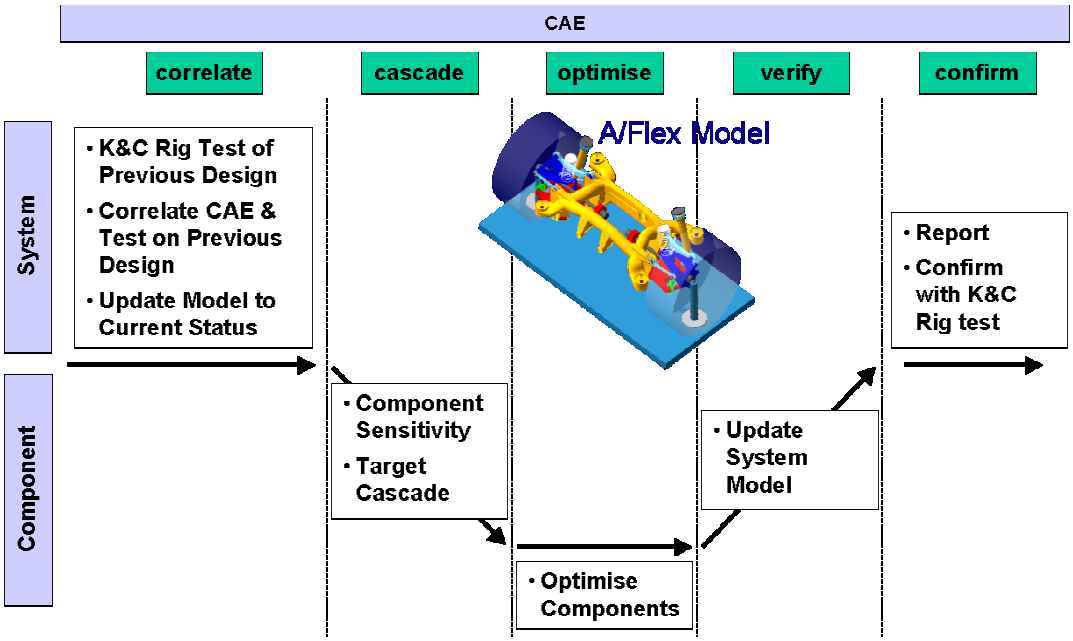


Figure 1.11 – Target Cascade System Component

1. Correlate:
   - K&C (kinematic and compliant) rig test of previous designs: a previous design, available in physical form, is used as a starting point for the development of the new assembly;
   - Correlation of CAE and test on previous designs: a virtual model of the previous design must be correlated to the results obtained on the physical model;
   - Update model to current status: the virtual model accurately represents the physical one.
2. Cascade:
   - Component sensitivity: the performance of the model is more dependant on some components than others;
   - Target cascade: once the targets for the new suspension are met, the target for each of the components in the assembly must be determined.
3. Optimise:
   - Component optimisation: each of the components is optimised from a set of different criteria (fatigue, static analysis, etc.).
4. Verify:
   - Update system model: the virtual model is updated by including to the optimised components, and new simulations are performed.
5. Confirm:
   - Report: all the changes for the suspension are reported;
   - Confirm with K&C rig test: the changes are implemented and tested in the physical prototype for validation.

The thesis focuses its attention on the steps 2 and 3 in order to tackle the problem of optimising the individual components while keeping the targets for the suspension assembly as a whole. While there exist system targets such as the compliances seen in Section **1.3**, there is also a set of component targets that has to be fulfilled by every component of the system, in order to make the system altogether fulfil its own targets. Of special interest for this thesis, among the system targets are the compliances of the suspension, which are set by the car maker. Once the system has been refined and performs as required, the targets for each component are calculated and given to the CAD department for its design. These component targets are basically determined using static analyses where each component is subjected to different load cases under certain boundary conditions. The deflection for each case is measured and reported, usually expressed as mm/kN.

1.7 Proposed methodology

Applying a new perspective, the virtual model available from the updated model in the step **correlate** of **Figure 1.7** is translated to a mathematical model and the sensitivity of the compliances to changes in the model is studied. An optimisation tool is then implemented to find the best set of modifications to every component in the assembly, using the system targets (compliances) as constraints.

What the new approach proposes is a methodology that focuses on the stiffness matrix of each of the components in the assembly. Instead of manually performing an educated trial and error process in which changes are implemented in the model in the search of lightening and improving it, a new approach is developed where an intermediate stage needs to be introduced. In this stage, the stiffness matrix representing the elastic properties of each component is produced, parameterised and optimised. Then, the components are recreated based on the optimised stiffness matrices.

Next the optimised component is subjected to a static analysis and the component targets re-issued. To perform the static analysis and calculate the component targets, a static load is applied to the component at a given node in a certain direction, and the displacement of the node in that direction is measured. This procedure is repeated in as many directions as needed according to the component. Software packages for FEM analysis such as MSC.Nastran provide the means to run this operation (which Nastran calls SOL101, linear static analysis), based on this method.

Finally, the new set of component targets is fed into the CAD department and the components are redesigned to meet the new targets, at which point the process is completed.

2 THE STIFFNESS MATRIX

It could be said that the stiffness matrix is the core of the present thesis. It is the fundamental stone from which the optimisation process was conceived in the first place and then built. The stiffness matrix is the mathematical representation of a component and opens possibilities to investigate, access and modify it. Thus it is of the highest importance to have a thorough knowledge of what it represents from a physical standpoint and also what are the mathematics behind it, its properties and its limitations. Therefore, in the present chapter, three main topics are presented and discussed: what the stiffness matrix is, its mathematical properties and physical meaning, and what can be done with it.

2.1 What is the stiffness matrix?

Before attempting to explain what the stiffness matrix is, it is better to start with a basic definition. *Stiffness* is the resistance of an elastic body to deflection by an applied force. The stiffness k of a body that deflects a distance D under an applied force R is: $k = R / D$. Therefore, stiffness is typically measured in N/m. As both the applied force and displacement are vectors, in general their relationship is characterised by a stiffness matrix, $[K]$ where: $[K]\cdot\vec{D}=\vec{R}$ or, in explicit form:

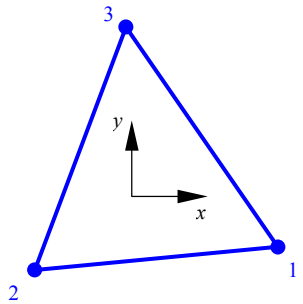
$$\begin{bmatrix} k_{1,1} & k_{1,2} & k_{1,3} & \cdots & k_{1,n} \\ k_{2,1} & k_{2,2} & k_{2,3} & \cdots & k_{2,n} \\ k_{3,1} & k_{3,2} & k_{3,3} & \cdots & k_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ k_{n,1} & k_{n,2} & k_{n,3} & \cdots & k_{n,n} \end{bmatrix} \cdot \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_n \end{pmatrix} \tag{2.1}$$

The displacement can, in general, refer to a point distinct from that where the force is applied, and a complicated structure will typically not deflect purely in the same direction as an applied force. The stiffness matrix enables such systems to be characterised in straightforward terms **[2.1]**. The n displacements contained in vector $\vec{D}$ are usually the primary unknowns, and they are also called *state variables* or *degrees of freedom* (DOF) of the system[2] **[2.2]**.

[2] *Primary unknowns* is the correct mathematical term whereas *degrees of freedom* has a mechanical idea: "any of a limited number of ways in which a body may move or in which a dynamic system may change" (Merrian-Webster). The term *state variables* is used more often in non-linear analysis and statistics.

2.1.1 Stiffness coefficients

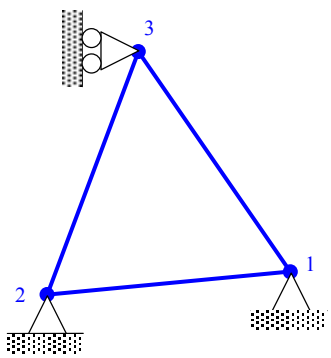
Figure 1 shows a 2-dimensional three-member truss with 3 joints or nodes, labelled 1, 2 and 3, and its corresponding stiffness matrix. The rows in the matrix represent the stiffness of the nodes in each direction, and the columns the direction of displacement **[2.2]**. For instance, the coefficient $k_{x1,y3}$ represents the contribution to the stiffness of node 1 in its x direction, when the node 3 is subjected to a unit displacement along its y direction. Notice that the model is presented without any boundary condition.



$$\begin{bmatrix} k_{x1,x1} & k_{x1,y1} & k_{x1,x2} & k_{x1,y2} & k_{x1,x3} & k_{x1,y3} \\ k_{y1,x1} & k_{y1,y1} & k_{y1,x2} & k_{y1,y2} & k_{y1,x3} & k_{y1,y3} \\ k_{x2,x1} & k_{x2,y1} & k_{x2,x2} & k_{x2,y2} & k_{x2,x3} & k_{x2,y3} \\ k_{y2,x1} & k_{y2,y1} & k_{y2,x2} & k_{y2,y2} & k_{y2,x3} & k_{y2,y3} \\ k_{x3,x1} & k_{x3,y1} & k_{x3,x2} & k_{x3,y2} & k_{x3,x3} & k_{x3,y3} \\ k_{y3,x1} & k_{y3,y1} & k_{y3,x2} & k_{y3,y2} & k_{y3,x3} & k_{y3,y3} \end{bmatrix}$$

Figure 2.1 – Three-member truss and its corresponding stiffness matrix

To clarify the meaning of the stiffness coefficients, an example is presented in **Figure 2.2** in which all displacements are prescribed, and only a unit displacement at node 3 in the y direction is allowed. To proceed, node 1 is subjected in x direction to a force of a magnitude given by the stiffness coefficient $k_{x1,y3}$ (actually, a unit force which in turn is multiplied by this coefficient), and in y direction to a force of a magnitude given by the stiffness coefficient $k_{y1,y3}$ (ditto). Likewise, the forces at node 2 in x and y direction are given by the stiffness coefficients $k_{x2,y3}$ and $k_{y2,y3}$, respectively, and the forces at node 3 in x and y direction are given by the stiffness coefficients $k_{x3,y3}$ and $k_{y3,y3}$, respectively.



$$\vec{D} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \quad \vec{R} = \begin{pmatrix} k_{x1,y3} \\ k_{y1,y3} \\ k_{x2,y3} \\ k_{y2,y3} \\ k_{x3,y3} \\ k_{y3,y3} \end{pmatrix}$$

Figure 2.2 – Constraints (displacements and forces) on the 3-member truss

2.1.2 Building the stiffness matrix

A good approach to understand how the stiffness matrix is built and the meaning of each component across rows and columns is by an example. In what follows, each truss member is assumed to be uniform, pin- and frictionless-connected at its ends, linearly elastic, and axially loaded. Displacements shown in sketches are greatly exaggerated; actual displacements are assumed to be small **[2.2]**. In the three-bar truss of **Figure 2.3**, nodes and elements (members) have been numbered arbitrarily. Let the stiffness of the elements be k_1, k_2 and k_3; that is, for any element i:

$$r_i = k_i e_i = \frac{A_i E_i}{L_i} e_i \tag{2.2}$$

where

r_i = force in element i

e_i = elongation of element i

A_i = cross-sectional area of element i

E_i = elastic modulus of element i

L_i = length of element i

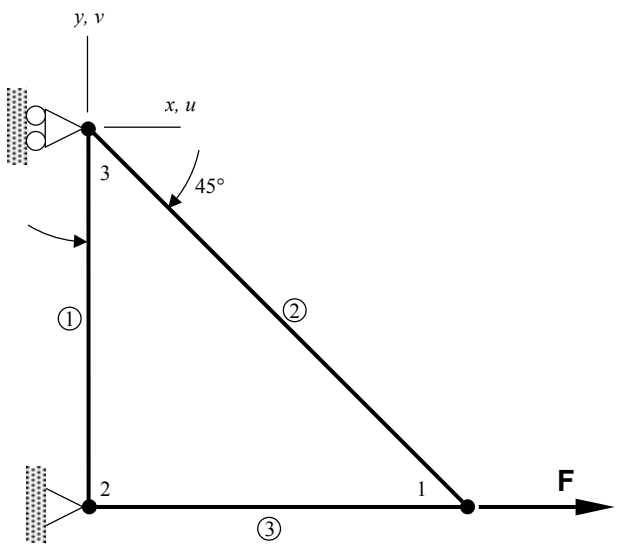
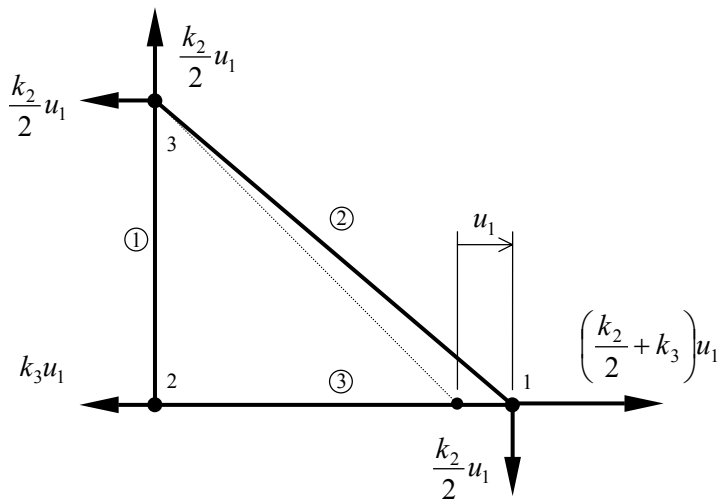


Figure 2.3 – Plane truss

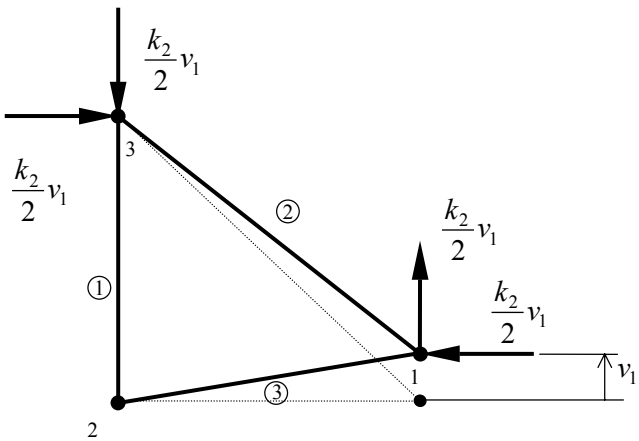
Let each of the three nodes be displaced a small amount, first in the x direction and then in the y direction, while all other nodal displacements are prohibited. In each of these six cases the external forces

that must be applied to maintain static equilibrium in the displaced configuration can be calculated. The first two cases (displacements in *x* and *y* directions for node 1) are shown in **Figures 2.4** and **2.5**.



$$\vec{D}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Figure 2.4 – Forces induced by displacement of node 1 in the *x* direction



$$\vec{D}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Figure 2.5 – Forces induced by displacement of node 1 in the *y* direction

As shown in **Figure 2.4**, when node 1 is displaced by u_1 (a distance *u* in the horizontal direction) the elongation at element 3 is also u_1 and the force produced by this element is simply k_3u_1. The elongation of element 2 is $u_1 \cdot \cos 45° = u_1 / \sqrt{2}$, so the force produced by this element is $k_2u_1 / \sqrt{2}$. Both forces are produced in the axial direction of the respective element. Since the angle between element 2 and 3 is 45°, the *x* and *y* direction components of the force produced by element 2 are:

$$k_2 \frac{u_1}{\sqrt{2}} \cdot \cos 45° = k_2 \frac{u_1}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}} = \frac{1}{2} k_2 u_1 . \tag{2.3}$$

As a result, if no other boundary conditions are applied to the truss, the forces induced by the members at every joint in each direction can be determined by simple observation, so:

$$\vec{R}_1 = \begin{pmatrix} \left(\frac{k_2}{2} + k_3\right) u_1 \\ -\frac{k_2}{2} u_1 \\ -k_3 u_1 \\ 0 \\ \frac{k_2}{2} u_1 \\ -\frac{k_2}{2} u_1 \end{pmatrix} . \tag{2.4}$$

As shown in **Figure 2.5**, the node 1 is displaced v_1 (a distance v in the vertical direction) and the resultant forces at every joint are:

$$\vec{R}_2 = \begin{pmatrix} -\frac{k_2}{2} v_1 \\ \frac{k_2}{2} v_1 \\ 0 \\ 0 \\ \frac{k_2}{2} v_1 \\ -\frac{k_2}{2} v_1 \end{pmatrix} . \tag{2.5}$$

As it can be seen from these first two cases, from the collection of six free-body diagrams it can be determined the equilibrating force required at any node in any displacement state. In equation form, the individual forces are:

$$p_1 = \left(\frac{k_2}{2} + k_3\right) u_1 - \frac{k_2}{2} v_1 - k_3 u_2 - \frac{k_2}{2} u_3 + \frac{k_2}{2} v_3 , \tag{2.6}$$

$$q_1 = -\frac{k_2}{2} u_1 + \frac{k_2}{2} v_1 + \frac{k_2}{2} u_3 - \frac{k_2}{2} v_3 , \tag{2.7}$$

$$p_2 = -k_3 u_1 + k_3 u_2 , \tag{2.8}$$

$$q_2 = k_1 v_2 - k_1 v_3 , \tag{2.9}$$

$$p_3 = -\frac{k_2}{2}u_1 + \frac{k_2}{2}v_1 + \frac{k_2}{2}u_3 - \frac{k_2}{2}v_3\,, \tag{2.10}$$

$$q_3 = \frac{k_2}{2}u_1 - \frac{k_2}{2}v_1 - k_1 v_2 - \frac{k_2}{2}u_3 + \frac{k_2}{2}v_3\,, \tag{2.11}$$

Finally, in matrix form these equilibrium equations are

$$\begin{bmatrix} \frac{k_2}{2}+k_3 & -\frac{k_2}{2} & -k_3 & 0 & -\frac{k_2}{2} & \frac{k_2}{2} \\ -\frac{k_2}{2} & \frac{k_2}{2} & 0 & 0 & \frac{k_2}{2} & -\frac{k_2}{2} \\ -k_3 & 0 & k_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & k_1 & 0 & -k_1 \\ -\frac{k_2}{2} & \frac{k_2}{2} & 0 & 0 & \frac{k_2}{2} & -\frac{k_2}{2} \\ \frac{k_2}{2} & -\frac{k_2}{2} & 0 & -k_1 & -\frac{k_2}{2} & k_1+\frac{k_2}{2} \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{Bmatrix} = \begin{Bmatrix} p_1 \\ q_1 \\ p_2 \\ q_2 \\ p_3 \\ q_3 \end{Bmatrix}. \tag{2.12}$$

Or, as it can be abbreviated these structural equations,

$$[K]\{D\} = \{R\}. \tag{2.13}$$

Each column of [K] in (2.13) sums to zero, because it represents an equilibrium set of nodal forces produced by unit displacement of one nodal degree of freedom, as shown in the two examples presented in **Figures 2.4** and **2.5**.

The above analysis may be applied to any structure, regardless of how many elements it has and its degree of static indeterminacy. It is always obtained as many equations as there are independent nodal displacements. If the structure is not a truss or frame, the stiffness properties of the elements are to be somehow approximated.

2.1.3 Using the stiffness matrix

As it can be noticed, (2.12) was obtained without taking any boundary conditions into account, that is, the data that tells which components of the vectors $\vec{D}$ and $\vec{R}$ are true unknown and which ones are known *a priori*.

In structural analysis procedures of manual approaches such information is used immediately by the analyst to discard unnecessary variables and thus reduce the amount of data that had to be carried along by hand, as it will be shown in Section **2.3**. The computer oriented philosophy is radically different and boundary conditions can wait until the last moment. This is because on the computer the sheer volume of data may not be as important as the efficiency with which it is organised, accessed and processed **[2.2]**.

To solve the system of equations represented by (2.13), the following procedure can be followed. It can be seen in **Figure 2.3** that the imposed boundary conditions are:

- forces
  - $p_1 = F$ load in horizontal positive direction at node 1
  - $q_1 = 0$ no force in vertical direction at node 1
  - $q_3 = 0$ no force in vertical direction at node 3
- displacements
  - $u_2 = 0$ no displacement in horizontal direction at node 2
  - $v_2 = 0$ no displacement in vertical direction at node 2
  - $u_3 = 0$ no displacement in horizontal direction at node 3

The remaining three forces (p_2, q_2 and p_3) and three displacements (u_1, v_1 and v_3) are as yet unknown. Formal solution for the unknowns may proceed as follows. Let the subscript *o* designate the known quantities and subscript *s* the remaining unknown quantities. By rearrangement of terms, (2.13) can be partitioned as follows:

$$\left[\begin{array}{c:c} K_{11} & K_{12} \\ \hdashline K_{21} & K_{22} \end{array}\right]\left\{\begin{array}{c} D_s \\ \hdashline D_o \end{array}\right\} = \left\{\begin{array}{c} R_o \\ \hdashline R_s \end{array}\right\}. \tag{2.14}$$

Consequently,

$$[K_{11}]\{D_s\} + [K_{12}]\{D_o\} = \{R_o\}, \tag{2.15}$$

$$[K_{21}]\{D_s\} + [K_{22}]\{D_o\} = \{R_s\}. \tag{2.16}$$

(2.15) and (2.16) yield to the unknown displacements and forces, respectively:

$$\{D_s\} = [K_{11}]^{-1}(\{R_o\} - [K_{12}]\{D_o\}) \tag{2.17}$$

$$\{R_s\} = [K_{21}]\{D_s\} + [K_{22}]\{D_o\} \tag{2.18}$$

Notice that to solve (2.16) $[K_{11}]^{-1}$ must exist, that is, K_{11} must not be singular. This condition is fulfilled as long as the component under study is correctly constrained, i.e., no free body motion is possible, and there are no spurious constraints, in which case K_{11} is rank deficient[3].

[3] When a square matrix *K* is supposed to be of rank *r*, but in fact has a smaller rank $\bar{r} < r$, the matrix is said to be *rank deficient*. The number $\bar{r} - r < 0$ is called the *rank deficiency*. As mentioned, in such a case the matrix *K* is *singular* and its determinant is 0.

2.2 Properties of the stiffness matrix

A general structure or component stiffness matrix, before specification of any boundary conditions, has four general characteristics or properties that can be used to check the formulation of a particular stiffness matrix **[2.2]**:

- Square - The number of rows is equal to the number of columns in the matrix.
- Symmetric - It implies that $k_{ij} = k_{ji}$. This is always the case when the displacements are directly proportional to the applied loads **[2.3]**. For a comprehensive discussion and demonstration of this property, see **Appendix A**.
- Singular - When the rank of a matrix is lower than its order, at least one of the equations is linear combination of the others. Therefore, the matrix will be singular and its determinant will be zero (see **Appendix B** for a presentation of some properties of matrices and determinants **[2.4]**). The physical reason is that, as mentioned, no displacement boundary conditions have yet been imposed and the structure is free to move as a rigid body. For any plane structure there are three independent rigid body motions (3 DOF); two may be translation and one rotation. Each is associated with zero forces {R}. Thus, for the three-bar truss, with displacement vectors {D} of {1 0 1 0 1 0}, {1 1 1 1 1 1} or {0 0 0 1 1 1}, for example, the product [K]{D} is zero. As suggested by these examples, an infinite number of rigid body motions are possible but only three are independent.
- Positive Diagonal Terms - All the terms in the main diagonal must be positive. If k_{ii} is negative then the force and its corresponding displacement would be oppositely directed, which is physically unreasonable. If $k_{ii} = 0$, then the displacement would produce no reaction force resisting it, which would imply that the structure is unstable.

As an example of the third property, the stiffness matrix of (2.13) is singular: its order is 6, but its rank only 3. By visual inspection in can be seen that rows 1, 5 and 6 can be expressed as linear combinations of rows 2, 3 and 4, as follows:

- row 1 = - (row 2 + row 3)
- row 5 = row 2
- row 6 = - (row 2 + row 4)

2.3 Modification of the stiffness matrix

On the basis of keeping the properties of the stiffness matrix intact, a simple set of rules can be withdrawn to allow the modification of the matrix coefficients without affecting the four properties dis

cussed above. As explained in Section **1.1**, the purpose is to work on the stiffness coefficients within the stiffness matrix in order to mathematically manipulate the elastic properties of the component and therefore its behaviour under load. The stiffness matrix itself keeps its external characteristics, namely to be square. Obviously, the order of the matrix (its size) is also kept because it depends purely on the number of DOF of the component.

Out of the list of properties of a determinant presented in **Appendix B**, two are particularly useful and will play a central roll in the proposed method:

- when at least one row (or column) of a matrix is a linear combination of the other rows (or columns) the determinant is zero. For example:

$$\begin{vmatrix} 5 & 2 & -1 \\ 6 & 2 & 8 \\ 1 & 0 & 9 \end{vmatrix}.$$

  The determinant is zero because row 2 = row 1 + row 3. The matrices with this characteristic have a rank lower than its order, and are so called singular.

- if the elements in any row (or column) have a common non-zero factor α, then the determinant equals the determinant of the corresponding matrix in which $\alpha = 1$, multiplied by α. For example:

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ \alpha \cdot a_{21} & \alpha \cdot a_{22} & \alpha \cdot a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = \alpha \cdot \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}.$$

As outlined in Section **2.2**, in a stiffness matrix representing a free body some of the equations are linearly dependent on the others. According to the first property of the determinants discussed above, this matrix is singular and its determinant is zero. But according to the second property presented, any of the rows (or columns) can be multiplied by a factor α and the determinant is multiplied by the same factor. In particular, if the determinant is zero, then it will still be zero after multiplying any row (or column) of the matrix by any factor. Moreover, if the factor is a positive number, the sign of the coefficients in the resulting matrix will be the same as before. Finally, instead of multiplying just an ith row by a factor, also the ith column is multiplied. In the next matrix, row and column 2 are multiplied by a factor α:

$$\begin{bmatrix} k_{1,1} & \alpha \cdot k_{1,2} & k_{1,3} & \cdots & k_{1,n} \\ \alpha \cdot k_{2,1} & \alpha^2 \cdot k_{2,2} & \alpha \cdot k_{2,3} & \cdots & \alpha \cdot k_{2,n} \\ k_{3,1} & \alpha \cdot k_{3,2} & k_{3,3} & \cdots & k_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ k_{n,1} & \alpha \cdot k_{n,2} & k_{n,3} & \cdots & k_{n,n} \end{bmatrix}$$

Therefore, it can be seen that multiplying a pair row$_i$/column$_i$ of a stiffness matrix corresponding to a free-body by a positive factor gives as a result a matrix with the four properties presented in the previous section:

- *Square*: operating on the stiffness coefficients does not change the dimensions of the matrix.
- *Symmetric*: multiplying both row and column ensures that $k_{ij} = k_{ji}$.
- *Singular*: the resulting matrix is still singular after multiplying a row (or column) by a positive modification factor.
- *Positive diagonal terms*: the original stiffness matrix has positive diagonal terms, and they remain so as long as the factor α is a positive number.

Although mathematically correct this procedure has no physical counterpart. In the modified stiffness matrix each column will no longer sum to zero because one of the terms has been modified when multiplying the ith row by α. For example, if the pair row/column 2 in the stiffness matrix has been multiplied by a factor α, when calculating the sum of column 3:

- before the modification, it is known that: $\sum_{i=1}^{n} k_{i3} = k_{13} + k_{23} + k_{33} + \ldots + k_{n3} = 0$, (2.19)
- after the modification: $\sum_{i=1}^{n} k_{i3} = k_{13} + \alpha \cdot k_{23} + k_{33} + \ldots + k_{n3}$, (2.20)

Rewriting the modified term as $\alpha \cdot k_{23} = k_{23} + (\alpha - 1) \cdot k_{23}$ and substituting in (2.20):

$$\sum_{i=1}^{n} k_{i3} = \underbrace{k_{13} + k_{23} + k_{33} + \ldots + k_{n3}}_{\substack{0 \\ (\text{Equation 9})}} + (\alpha - 1) \cdot k_{23} = (\alpha - 1) \cdot k_{23} \neq 0 \,. \tag{2.21}$$

Therefore, this procedure is simply a mathematical tool to perturb the DOF of a component, and to be able to analyse the changes in the performance of the suspension assembly as a whole. In this way it is possible to pinpoint those DOF that are critical to a specific measurement and to tune the performance of the suspension assembly.

However, in the particular case that a force is applied at a DOF and the displacement is measured in the same DOF, this procedure does have a physical correspondence. This displacement is exactly what is needed when the targets for the components must be issued for the topology optimisation phase, as explained in Section **1.1**.

To illustrate this physical correspondence, the simple 2-dimensional truss member of **Figure 2.3** is taken as a study case. Allowing for the boundary conditions in (2.12) and rearranged to suit (2.14), the situation can be formulated as:

$$\begin{bmatrix} \frac{k_2}{2}+k_3 & -\frac{k_2}{2} & \frac{k_2}{2} & -k_3 & 0 & -\frac{k_2}{2} \\ -\frac{k_2}{2} & \frac{k_2}{2} & -\frac{k_2}{2} & 0 & 0 & \frac{k_2}{2} \\ \frac{k_2}{2} & -\frac{k_2}{2} & k_1+\frac{k_2}{2} & 0 & -k_1 & -\frac{k_2}{2} \\ -k_3 & 0 & 0 & k_3 & 0 & 0 \\ 0 & 0 & -k_1 & 0 & k_1 & 0 \\ -\frac{k_2}{2} & \frac{k_2}{2} & -\frac{k_2}{2} & 0 & 0 & \frac{k_2}{2} \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ v_3 \\ 0 \\ 0 \\ 0 \end{Bmatrix} = \begin{Bmatrix} F \\ 0 \\ 0 \\ p_2 \\ q_2 \\ p_3 \end{Bmatrix}. \tag{2.22}$$

According to (2.17) the displacement at node 1 in x direction can be determined as:

$$\begin{Bmatrix} u_1 \\ v_1 \\ v_3 \end{Bmatrix} = \begin{bmatrix} \frac{k_2}{2}+k_3 & -\frac{k_2}{2} & \frac{k_2}{2} \\ -\frac{k_2}{2} & \frac{k_2}{2} & -\frac{k_2}{2} \\ \frac{k_2}{2} & -\frac{k_2}{2} & k_1+\frac{k_2}{2} \end{bmatrix}^{-1} \cdot \left(\begin{Bmatrix} F \\ 0 \\ 0 \end{Bmatrix} - \begin{bmatrix} -k_3 & 0 & -\frac{k_2}{2} \\ 0 & 0 & \frac{k_2}{2} \\ 0 & -k_1 & -\frac{k_2}{2} \end{bmatrix} \cdot \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \right). \tag{2.23}$$

For a square matrix of order 3, its inverse can be calculated by

$$[K] = \begin{bmatrix} k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,1} & k_{3,2} & k_{3,3} \end{bmatrix} \Rightarrow [K]^{-1} = \frac{1}{|K|} \cdot \begin{bmatrix} \begin{vmatrix} k_{2,2} & k_{2,3} \\ k_{3,2} & k_{3,3} \end{vmatrix} & \begin{vmatrix} k_{3,2} & k_{3,3} \\ k_{1,2} & k_{1,3} \end{vmatrix} & \begin{vmatrix} k_{1,2} & k_{1,3} \\ k_{2,2} & k_{2,3} \end{vmatrix} \\ \begin{vmatrix} k_{3,1} & k_{3,3} \\ k_{2,1} & k_{2,3} \end{vmatrix} & \begin{vmatrix} k_{1,1} & k_{2,3} \\ k_{3,1} & k_{3,3} \end{vmatrix} & \begin{vmatrix} k_{2,1} & k_{2,3} \\ k_{1,1} & k_{1,3} \end{vmatrix} \\ \begin{vmatrix} k_{2,1} & k_{2,2} \\ k_{3,1} & k_{3,2} \end{vmatrix} & \begin{vmatrix} k_{3,1} & k_{3,2} \\ k_{1,1} & k_{1,2} \end{vmatrix} & \begin{vmatrix} k_{1,1} & k_{1,2} \\ k_{2,1} & k_{2,2} \end{vmatrix} \end{bmatrix}. \tag{2.24}$$

So,

$$\begin{bmatrix} \frac{k_2}{2}+k_3 & -\frac{k_2}{2} & \frac{k_2}{2} \\ -\frac{k_2}{2} & \frac{k_2}{2} & -\frac{k_2}{2} \\ \frac{k_2}{2} & -\frac{k_2}{2} & k_1+\frac{k_2}{2} \end{bmatrix}^{-1} = \frac{1}{|K|} \cdot \begin{bmatrix} \frac{k_2}{2}\cdot k_1 & \frac{k_2}{2}\cdot k_1 & 0 \\ \frac{k_2}{2}\cdot k_1 & \frac{k_2}{2}\cdot k_1 + k_3\cdot k_1 + \frac{k_2}{2}\cdot k_3 & \frac{k_2}{2}\cdot k_3 \\ 0 & \frac{k_2}{2}\cdot k_3 & \frac{k_2}{2}\cdot k_3 \end{bmatrix}. \tag{2.25}$$

Therefore,

$$\begin{Bmatrix} u_1 \\ v_1 \\ v_3 \end{Bmatrix} = \begin{bmatrix} \frac{k_2}{2}+k_3 & -\frac{k_2}{2} & \frac{k_2}{2} \\ -\frac{k_2}{2} & \frac{k_2}{2} & -\frac{k_2}{2} \\ \frac{k_2}{2} & -\frac{k_2}{2} & k_1+\frac{k_2}{2} \end{bmatrix}^{-1} \cdot \begin{Bmatrix} F \\ 0 \\ 0 \end{Bmatrix}. \tag{2.26}$$

And

$$\begin{Bmatrix} u_1 \\ v_1 \\ v_3 \end{Bmatrix} = \frac{1}{|K|} \cdot \begin{Bmatrix} F \cdot \frac{k_2}{2} \cdot k_1 \\ F \cdot \frac{k_2}{2} \cdot k_1 \\ 0 \end{Bmatrix}. \tag{2.27}$$

Next, the stiffness of node 1 in the horizontal direction is modified to investigate its influence in the displacement vector, according to the modification proposed. Therefore, its pair row/column in the stiffness matrix can be multiplied by a factor α, as follows:

$$\begin{bmatrix} \alpha^2 \cdot \left(\frac{k_2}{2} + k_3\right) & \alpha \cdot \left(-\frac{k_2}{2}\right) & \alpha \cdot \frac{k_2}{2} & \alpha \cdot (-k_3) & 0 & \alpha \cdot \left(-\frac{k_2}{2}\right) \\ \alpha \cdot \left(-\frac{k_2}{2}\right) & \frac{k_2}{2} & -\frac{k_2}{2} & 0 & 0 & \frac{k_2}{2} \\ \alpha \cdot \frac{k_2}{2} & -\frac{k_2}{2} & k_1 + \frac{k_2}{2} & 0 & -k_1 & -\frac{k_2}{2} \\ \alpha \cdot (-k_3) & 0 & 0 & k_3 & 0 & 0 \\ 0 & 0 & -k_1 & 0 & k_1 & 0 \\ \alpha \cdot \left(-\frac{k_2}{2}\right) & \frac{k_2}{2} & -\frac{k_2}{2} & 0 & 0 & \frac{k_2}{2} \end{bmatrix} \begin{Bmatrix} u_1' \\ v_1' \\ v_3' \\ 0 \\ 0 \\ 0 \end{Bmatrix} = \begin{Bmatrix} F \\ 0 \\ 0 \\ p_2 \\ q_2 \\ p_3 \end{Bmatrix}. \tag{2.28}$$

The modified stiffness matrix is called $[K']$, and its determinant is $|K'| = \alpha^2 \cdot |K|$. According to (2.17) the displacement at node 1 in x direction can be determined as:

$$\begin{Bmatrix} u_1' \\ v_1' \\ v_3' \end{Bmatrix} = \begin{bmatrix} \alpha^2 \cdot \left(\frac{k_2}{2} + k_3\right) & \alpha \cdot \left(-\frac{k_2}{2}\right) & \alpha \cdot \frac{k_2}{2} \\ \alpha \cdot \left(-\frac{k_2}{2}\right) & \frac{k_2}{2} & -\frac{k_2}{2} \\ \alpha \cdot \frac{k_2}{2} & -\frac{k_2}{2} & k_1 + \frac{k_2}{2} \end{bmatrix}^{-1} \cdot \left(\begin{Bmatrix} F \\ 0 \\ 0 \end{Bmatrix} - \begin{bmatrix} \alpha \cdot (-k_3) & 0 & \alpha \cdot \left(-\frac{k_2}{2}\right) \\ 0 & 0 & \frac{k_2}{2} \\ 0 & -k_1 & -\frac{k_2}{2} \end{bmatrix} \cdot \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \right), \tag{2.29}$$

$$\begin{Bmatrix} u_1' \\ v_1' \\ v_3' \end{Bmatrix} = \begin{bmatrix} \alpha^2 \cdot \left(\frac{k_2}{2} + k_3\right) & -\alpha \cdot \frac{k_2}{2} & \alpha \cdot \frac{k_2}{2} \\ -\alpha \cdot \frac{k_2}{2} & \frac{k_2}{2} & -\frac{k_2}{2} \\ \alpha \cdot \frac{k_2}{2} & -\frac{k_2}{2} & k_1 + \frac{k_2}{2} \end{bmatrix}^{-1} \cdot \begin{Bmatrix} F \\ 0 \\ 0 \end{Bmatrix}. \tag{2.30}$$

Which, after calculating the inverse,

$$\begin{Bmatrix} u_1' \\ v_1' \\ v_3' \end{Bmatrix} = \frac{1}{\alpha^2 \cdot |K|} \cdot \begin{bmatrix} \frac{k_2}{2} \cdot k_1 & \alpha \cdot \frac{k_2}{2} \cdot k_1 & 0 \\ \alpha \cdot \frac{k_2}{2} \cdot k_1 & \alpha^2 \cdot \left(\frac{k_2}{2} \cdot k_1 + k_3 \cdot k_1 + k_3 \cdot \frac{k_2}{2}\right) & \alpha^2 \cdot k_3 \cdot \frac{k_2}{2} \\ 0 & \alpha^2 \cdot k_3 \cdot \frac{k_2}{2} & \alpha^2 \cdot k_3 \cdot \frac{k_2}{2} \end{bmatrix} \cdot \begin{Bmatrix} F \\ 0 \\ 0 \end{Bmatrix}. \tag{2.31}$$

Therefore,

$$\begin{Bmatrix} u_1' \\ v_1' \\ v_3' \end{Bmatrix} = \frac{1}{|K|} \cdot \begin{Bmatrix} \frac{1}{\alpha^2} \cdot F \cdot \frac{k_2}{2} \cdot k_1 \\ \frac{1}{\alpha} \cdot F \cdot \frac{k_2}{2} \cdot k_1 \\ 0 \end{Bmatrix}. \tag{2.32}$$

2.4 Conclusions

Having understood the meaning of the stiffness matrix, its coefficients and its properties, and after having studied the consequences of the modification procedure proposed for the stiffness matrix from a mathematical standpoint, it is of particular interest to analyse the physical implication of these modifications. As it could be seen from (2.27) and (2.32), the displacement of node 1 in *x* direction for the reference and the modified case is:

$$u_1 = \frac{1}{|K|} \cdot F \cdot \frac{k_2}{2} \cdot k_1 , \tag{2.33}$$

$$u_1' = \frac{1}{\alpha^2} \cdot \frac{1}{|K|} \cdot F \cdot \frac{k_2}{2} \cdot k_1 = \frac{1}{\alpha^2} \cdot u_1 . \tag{2.34}$$

For this study case, both the force and the displacement are in the same DOF, which is the sort of study performed during the static analysis to calculate the targets for the component after the optimisation phase. As it can be seen, compared with the results from the unmodified matrix, the solution for the node is the same as the reference multiplied by $1/\alpha^2$. This is senseful since the DOF is stiffened when multiplied by a factor higher than 1, so the displacement of the node under the same conditions must be smaller.

In the following case the displacement is measured in a different DOF than that in which the force acts. From (2.12) and (2.13), the displacement of node 1 in *y* direction for the reference and the modified case are:

$$v_1 = \frac{1}{|K|} \cdot F \cdot \frac{k_2}{2} \cdot k_1 , \tag{2.35}$$

$$v_1' = \frac{1}{\alpha} \cdot \frac{1}{|K|} \cdot F \cdot \frac{k_2}{2} \cdot k_1 = \frac{1}{\alpha} \cdot v_1 . \tag{2.36}$$

According to the proposed method to modify the stiffness matrix, the coefficients in row *i* ($k_{i,1\text{-}n}$) are multiplied by α and also the coefficients in column *i* ($k_{1\text{-}n,i}$), so the coefficient k_{ii} is multiplied by α^2.

The meaning of this was analysed both from a mathematics standpoint and from the physical perspective. As it was established, the factor α^2 that multiplies the coefficient k_{ii} in the main diagonal has particular importance in the behaviour of a component, since it is what changes the stiffness of a node to move in a given direction, when a force is applied at that node in that direction, exactly what is tested when running a static analysis to produce the components targets.

Finally, since the purpose is to find the right set of modification factors to apply to the stiffness matrices representing the components in a suspension in order to lower their weight, the range of values that the factors can take go from a lower limit set by the user to 1. This upper limit is set so the stiffness of the component is never increased, which would mean an increment in its mass, which is against the aim of the optimisation process.

3 OVERVIEW OF THE PROCEDURE

Based on the possibilities that opens the manipulation of the stiffness matrix according to the method presented in Chapter 2, a process to optimise a whole suspension assembly is proposed and developed. The process can be described by a flux diagram as shown in **Figure 3.1**. The components available as FE models are translated into their stiffness matrices **(Figure 3.2)**, modified, and converted to a format useful for the simulations over the suspension assembly. Once the components have been optimised, a static analysis is performed and the new targets are calculated.

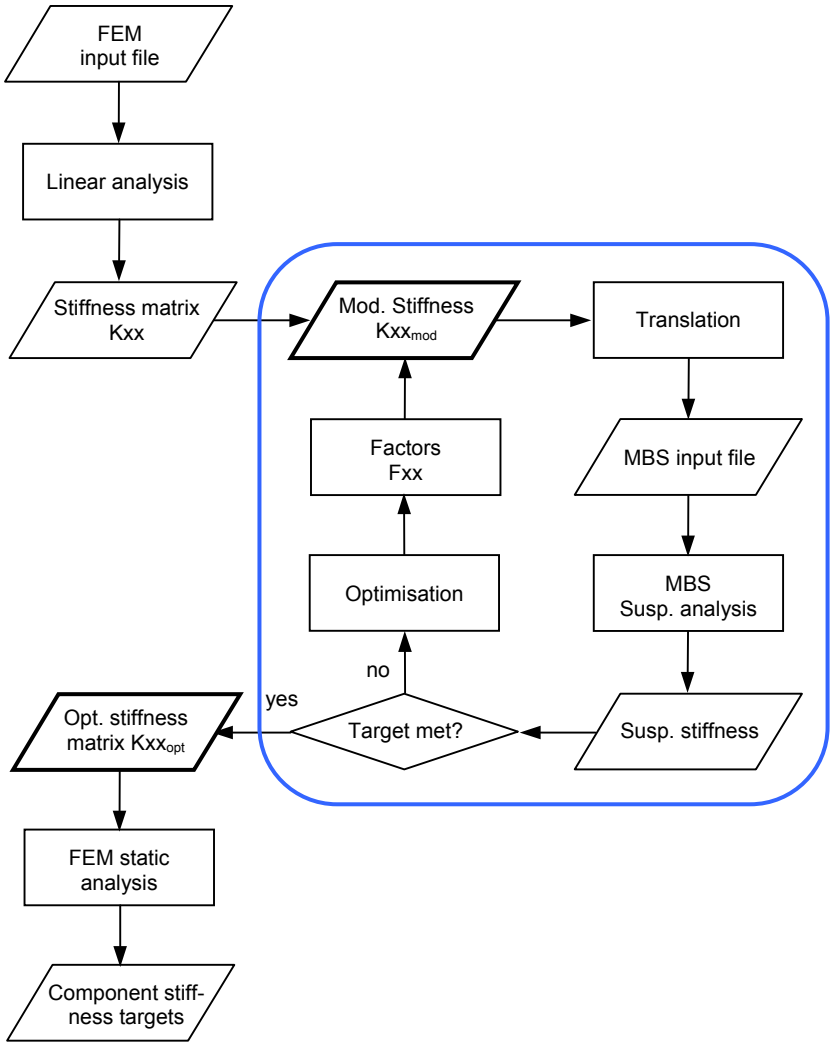
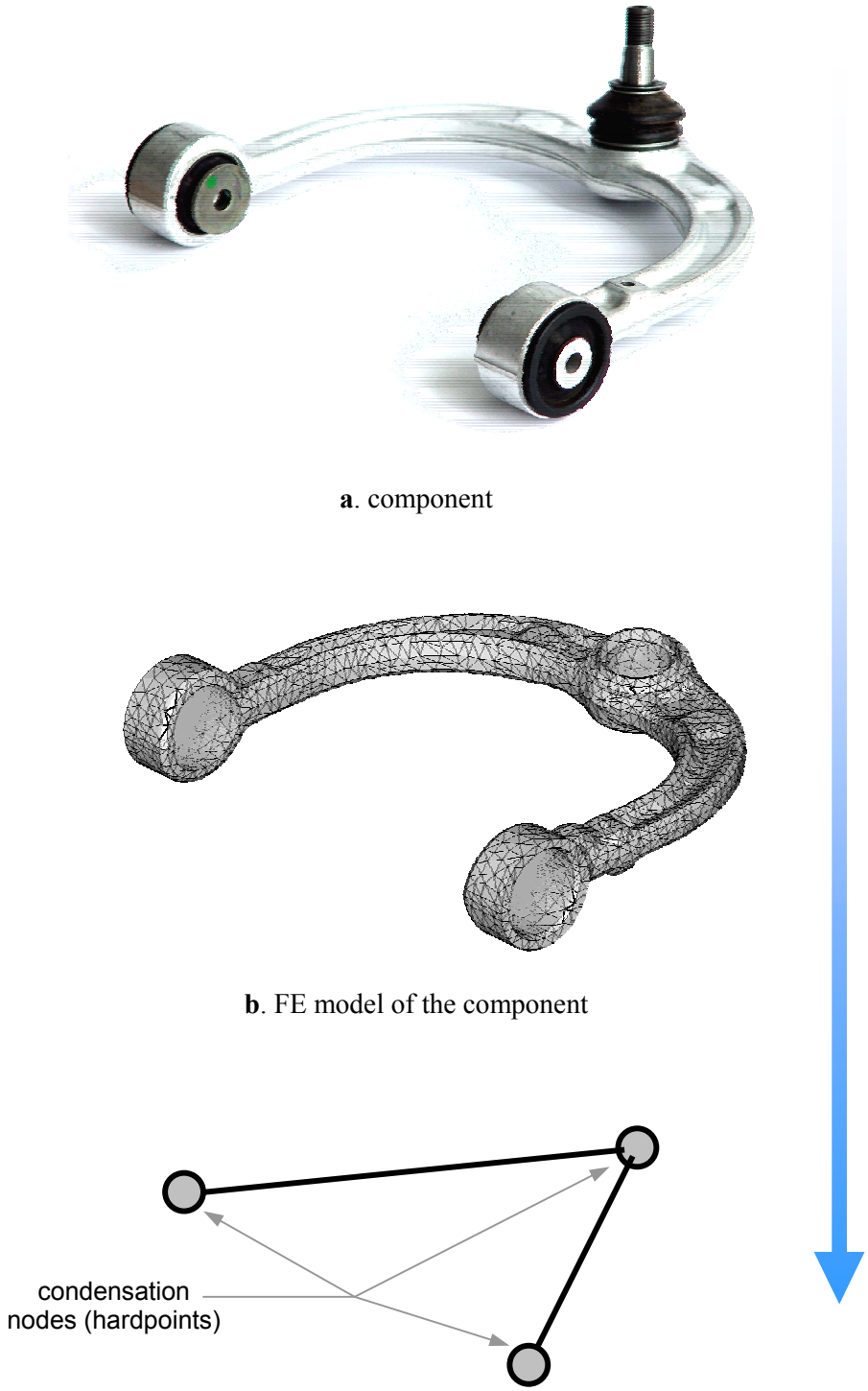


Figure 3.1 – Flux diagram of the process to optimise a suspension assembly



a. component

b. FE model of the component

c. condensed model of the component

Figure 3.2 – Condensation of a component for the optimisation process

As it can be seen, there is a fair amount of exchange of different files between various software packages, to be able to modify a flexible component created with a FE pre-processor and include it into a

multi-body simulation, replacing the original component. In the process, the FE model is manipulated not only to make it usable for the simulations, but also to provide the intermediate stage in which it is possible to make changes in the characteristics of the component by modifying its stiffness matrix.

The different software packages involved in these operations are:

- MSC.ADAMS/Cars is a multi-body motion simulation software, offering a specialised environment for modelling vehicles. It allows creating virtual prototypes of vehicle subsystems and analysing them much like the physical prototypes would be analysed.
- MSC.Nastran is a FEA solver, structural analysis program for analysing stress, vibration, dynamic, non-linear and heat transfer characteristics of structures and mechanical components. The two main features to be used from these software are:
  - SOL103, intended to find the normal modes of vibration of a component. When called, this solution internally calculates the stiffness matrix of the component and finds its eigenvectors. As an option it produces an output file which is used for the simulations in ADAMS. It is divided in two steps, first, the stiffness matrix is produced, with the particularity that it is condensed to the interface nodes (for an explanation of the condensation process, see **Appendix C [2.2]**) and, second, the file for the simulation in ADAMS is produced by translating the condensed stiffness matrix.
  - SOL101, linear static analysis, which allows imposing boundary conditions and loads to a component to find out its elastic behaviour under different situations.
- MATLAB is a language for technical computing. It integrates computation, visualisation and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. It also allows calling other programs and externally running processes such as FE analyses and ADAMS simulations, and reading and using the results.

4 OPTIMISATION ALGORITHMS

4.1 Introduction

The optimisation of a product **[4.1]**, i.e., improvement of its characteristics, has been an integral part of the development tasks for several years now, and numerous methods are available for this. Nevertheless, their suitability and cost effectiveness, the latter expressed in the number of solver runs, can not be guaranteed a priori. Besides considerable need of computational resources, the integration of these methods into the existing process is likely to demand a considerable effort. Nevertheless, there is a consensus on the fact that this integration is of enormous importance, taking into account that the development cycles are becoming shorter and shorter. A great innovation potential and competitive advantage is seen here.

In this chapter, methods of optimisation are introduced and discussed from a practical point of view. The focus lays less on the details of the methods but on demands, restrictions and application areas. Evidently, an evaluation as concrete as possible of the numerous algorithms that shall be comprehensible to the beginners in this field inevitably calls for simplifications and the concentration on so far successful applications. Because of the birth of hybrid approaches or the specialisation and enhancements of single methods, the areas of application of the different methods are no longer exactly outlined but appear to be movable. Nevertheless, it can not be expected for the near future that one single algorithm can efficiently solve the majority of the optimisation tasks.

Regardless of the application field and the particular case, an optimisation problem is posed using the following basic entities, as described in the on-line documentation of **[4.2]**:

- *Objective Function*: an objective function is the function to minimise, maximise or achieve a target value for. For instance, while designing a structure, the objective function might be to minimise the stress at a particular location in the structure, or the overall weight of the complete structure.
- *Design Variables* are a set of unknowns or variables affecting the value of the objective function, which the optimiser can use to modify its output.
- *Constraints* are the bounds on the response functions of a system that need to be satisfied in order for a design to be acceptable.

A typical optimisation problem will find values for design variables which minimise, maximise, or achieve a target value for the objective function while satisfying a set of constraints. Mathematically, it can be formulated as follows:

minimise $F_{(x_i)}$

subject to $g_{(x_i)} < 0$

and/or $x_L < x_i < x_U$

where

$F_{(x_i)}$ is the objective function,

$g_{(x_i)}$ is the constraint function,

x_i is the vector of design variables.

In optimisation tasks, the variation space or design space is defined by optimisation variables x_i. These can take continuous values between an upper (x_U) and lower (x_L) boundary as well as discrete values, and/or be under certain constraints expressed as, for example, $g_{(x_i)} < 0$. The desired properties of an optimal design are defined by objective functions such as $F_{(x_i)}$. Then, optimisation methods search the design space for as good an approximation as possible of both objective functions and constraints.

When this process involves more than one calculation discipline, the term *multidisciplinary optimisation* is applied. When more than one objective function is used, the term *multi-criteria optimisation* is used. As it will be seen when describing the different optimisation methods, particularly Evolutionary methods, it is usually better practice to combine all the objective functions into one, in order to simplify the optimisation algorithm.

An important aspect to consider is the way the objective function is defined and how multiple constraints are dealt with. When optimising a suspension, for instance, the objective function $F_{(x_i)}$ is intended to be the weight of the components in the assembly. Such quantity cannot be determined with the optimisation process proposed in Chapter 2 by any means, since only the stiffness matrix is available and modified, that is, only the elastic properties of the material are considered. Therefore, a way has to be found that gives at least an approximation to what is happening with the weight of the components through out the optimisation process. This is possible by keeping track of the optimisation factors that multiply the pairs row/column in the stiffness matrices. Intuitively, the lower these factors, the weaker the components will become, requiring less material to build them, hence, less weight. Therefore, the objective function is simply the aggregate of all the modification factors applied to the set of stiffness matrices that define the components of a suspension assembly. It could be argued that

such a measure can be somehow not accurate, since a modification of a given DOF can have more influence on the weight than other DOFs. However, this is still a good approximation and a valid approach to define the objective function since it provides the means to compare two possible solutions.

4.2 Fitness function

In order to allow an optimisation algorithm to take into account several constraints simultaneously, usually the best solution is to find a way to combine them into a single constraint or, even better, to include them into the objective function. For the optimisation task undertaken in this thesis, the latest is attempted. The procedure to achieve this aim is as follows:

1. measure the compliances of the original (not optimised) suspension assembly. These are the reference values. For example, there are two values to measure: toe and camber compliance.
2. apply some modification to the components.
3. measure the compliances of the modified assembly.
4. calculate the aggregate of the normalised variation of each measurement, that is:

   $$\frac{tc_{modified} - tc_{original}}{tc_{original}} + \frac{cc_{modified} - cc_{original}}{cc_{original}}$$

   where

   tc = toe compliance

   cc = camber compliance
5. multiply the "weight" of the assembly (as discussed before) by the result of the previous step.

In such a way, the objective function $F_{(x_i)}$ and the constraints are condensed in one so called fitness function $F'_{(x_i)}$ that considers improvements regarding both the weight of the components and closeness of the performance of the modified suspension assembly to the original design.

4.3 Optimisation methods

Generally, at least three method classes are available to perform the optimisation task:

- mathematical optimisation by means of gradients,
- response surface methods,
- stochastic search strategies.

In the following paragraphs, the three methods are introduced and some important characteristics are considered in order to evaluate how suitable is each of them for the particular optimisation task tackled in this thesis. These characteristics are:

- convergence,
- efficiency (measured in terms of computational time needed or runs),
- ability to cope with multi-parameter and multi-constraint optimisation tasks,

4.3.1 Gradient based methods

Mathematical optimisation methods determining search direction by means of gradient information possess the best convergence behaviour towards the optimum of the aforementioned methods. However, they do pose the strictest demands on the mathematical formulation of the problem regarding continuity, differentiability, smoothness, and scalability. Additionally, a highly accurate determination of the gradients is needed. The most critical aspect from a practical point of view is the determination of the gradients. For many tasks, gradients of important response variables can not be determined analytically or semi-analytically, and a numerical determination (such as finite differences) may fail, for example, for noisy or non differentiable tasks, or simply cannot be accurate enough, especially for very low gradient values.

Gradient based approaches require a study of every function under optimisation with respect to every variable. If the objective is to minimise a function, the scheme selectively modifies the variables that present the highest gradients in the direction that decreases the output of the function, until the gradients become zero or the constraints are violated. Obviously, as it can be seen from **Figure 4.1**, the disadvantage of gradient based methods is that they are more likely to find local rather than absolute minimum, which in turn is only depending on the seed values used for the design parameters (points *X*, *Y* and *Z* in the figure, which will make the algorithm to converge to point *F*, a local minimum, instead of the global minimum *D*).

Therefore, a successful practical application is restricted mainly to optimisation tasks with continuous optimisation variables and well posed mathematical problem formulations permitting to determine appropriate gradients. Gradient methods should if possible start in admissible design areas, i.e., those areas fulfilling all constraints, which often is possible if there is data available from previous designs. To avoid falling in local optima, a strategy of several optimisations starting from different points is recommended.

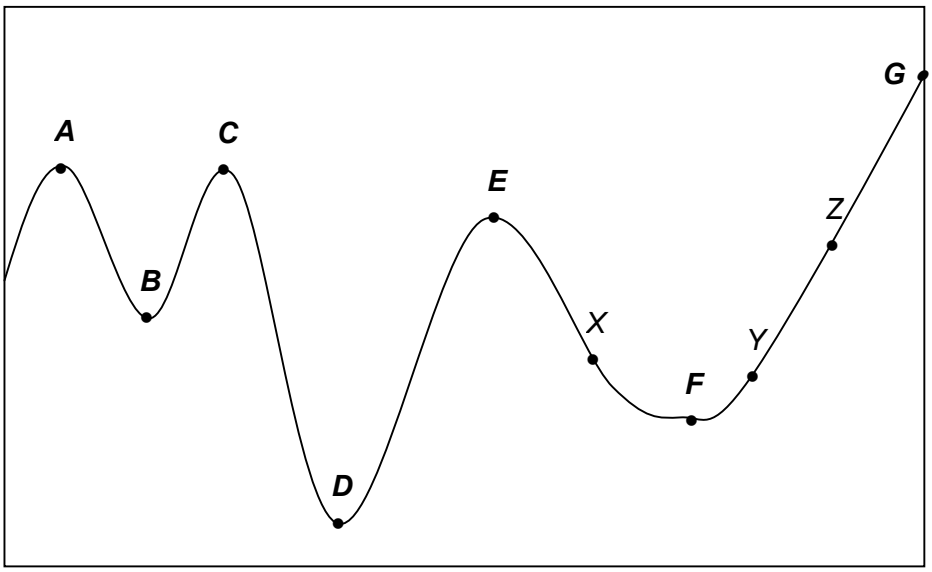


Figure 4.1 – Objective function with several peaks

4.3.2 Response Surface Methods

For tasks that are not appropriate for mathematical optimisation methods and do not involve more than 5 to 15 optimisation variables, *Response Surface Methods* (RSM) are an attractive alternative. These methods generate an approximation of the design space by means of approximation functions, based on an appropriate set of samples of the design space. The samples should be determined by means of sample patterns (design of experiments, DOE) that are fitted for the applied approximation functions.

Typically, the approximation functions possess good mathematical properties so mathematical optimisation tools can be used for the optimum search in the substitute space. However, it is not a trivial task to prove that the approximation is utilisable at the regions of interest within the design space, nor that they provide sufficient accuracy for the optimisation. Therefore, adaptation schemes are used to assure the approximation quality. Adaptive response surface methods, zooming and shifting the approximation space until the optimum converges on the response surface, prove to be most successful **[4.3]**.

A practical use of RSM is restricted mainly by the number of optimisation variables. Nowadays, adaptive response surface methods are successfully applied, e.g., for noisy tasks with up to 10, in some applications up to 15 optimisation variables. This problem class is frequently found in explicit finite element analyses, multi-body simulation, crash simulation, etc.

4.3.3 Evolutionary Search Strategies

When neither mathematical methods alone nor in combination with response surface methods succeed, stochastic search strategies remain to solve the optimisation task. Of all methods belonging to this

class, evolutionary methods in its two forms, genetic algorithms (GA) and evolutionary strategies (ES) are the most successful ones **[4.1]**. The term *stochastic search strategies* is applied here because are random events what lead to design modification. Frequently, the application of stochastic search strategies is called *design improvement* rather than optimisation. That is because these methods have a convergence much worse towards the optimum than mathematical methods and necessitate a very large amount of design evaluations to converge. Since there is no information about the effect of every design change on the performance throughout the optimisation process, it is the fitness function what filters the less appropriate individuals and allows the best ones to survive through the generations.

The main difference between genetic algorithms and evolutionary strategies lies in how the optimisation variables evolve. For genetic algorithms, the most important evolutionary process is the random interchange of genes, i.e., optimisation variables, between two parent designs in order to create descendants. On the other hand, for evolutionary search strategies, mutation, i.e., random modification, of single genes of a parent design in order to create one descendent is the most important process. This leads to different advantages and recommendable application areas of both methods. Genetic algorithms are especially suitable for a relatively wide search of the design space. That is why they are often employed for the search of different design areas of comparably good performance (island search) or for a design improvement without any previous knowledge entered into the evolution. In contrast, evolutionary strategies are most appropriate for a design improvement of pre-optimised design islands or for construction states for which previous knowledge can be integrated into the start generation or into the evolution operators.

The advantages of both genetic and evolutionary strategies can be combined by using hybrid algorithms, self-adjusting or adaptive evolutionary methods, and thus the speed of design improvement can be increased.

4.4 Comparative analysis of the optimisation methods

Typically, in order to compare the performance of the different optimisation methods, a test environment for the algorithms must be provided in the form of several objective functions f: $\mathbb{R}^n \rightarrow \mathbb{R}$. Finding an appropriate and representative set of test problems is not an easy task, since any particular combination of properties represented by a test function does not allow for generalised performance statements. Nonetheless, there is evidence **[4.4]** from a vast number of applications that both Gradient Based approaches and Evolutionary Algorithms are robust, in the sense that they give rather good performance over a wide range of different topologies, the later one having the advantage of being able to handle a higher number of optimisation parameters. This feature alone makes this method the most likely candidate to be adopted for the problem of optimising a suspension assembly by the

method discussed in Section **2.3**. A legitimate objection against Response Surface Methods is that they may not be representative of the "average complexity" of real-world problems.

In the test environment, the test functions are completely artificial and simple to be used, i.e., they are stated in a closed, analytical form, and have no direct background from any practical application. Instead, they allow for a detailed analysis of certain special characteristics of the topology, e.g., slope changes, convexity/concavity, etc. If any prediction is drawn up for the behaviour of the optimisation algorithms depending on such strong topological characteristics, the appropriate idealised test function provides a good instrument to test such hypotheses. Furthermore, since many known test sets have some functions in common, at least a minimal level of comparability of results is often guaranteed.

Finally, before an algorithm can be expected to succeed in the case of hard problems, it has to demonstrate that it does not fail to work on simple ones, such as the example shown in **Figure 4.2**, where the surface (vertical axis in the chart) represents the objective function, which depends on two variables represented by each horizontal axis of the chart. The horizontal plane represents a limit for the function. For example, the output of the function is the camber compliance of a given suspension assembly, and the two variables X and Y represent the stiffness of two DOF in the system.

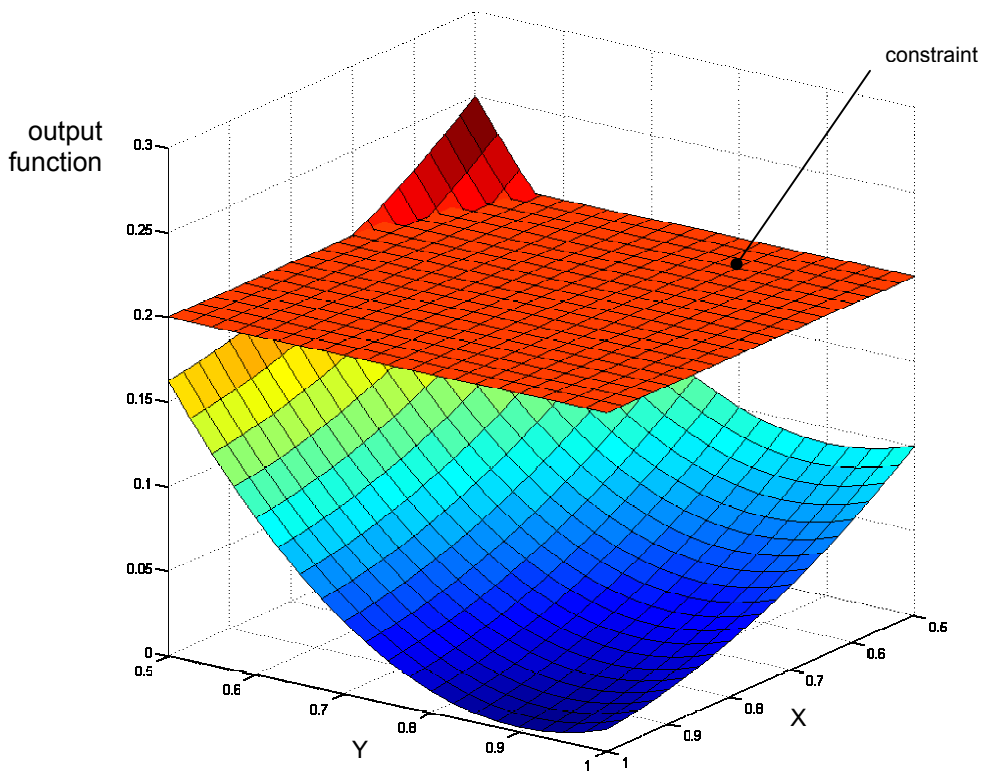


Figure 4.2 – Output function depending of the modification factors applied to X and Y

It can immediately be noticed that, even for this very simple, mono-criterion case, there is no unique solution. When the objective function reaches the constraint, there are several combinations of values that X and Y can take which keep the objective function reaching the constraint value. This set of points is called the Pareto curve.

However, it is the experience of the author that within the domain of the sort of problems of interest in this thesis, the functions describing the performance of the compliances in a suspension assembly perform in a rather smooth and simple way, such as depicted in **Figure 4.2**. Specifically, the behaviour of the DOF over the allowed range of values for the modification factors can be accurately approximated by a second order polynomial, as shown in **Figure 4.3**. This is due to the field of influence that a DOF can have on the compliances of a suspension assembly. When a DOF is weakened within the boundaries imposed (a lower limit for the modification factors set by the user, and an upper limit equal to 1), the suspension tends to modify its behaviour in a smooth, monotonous and predictable manner, either increasing one or more compliances (the common effect) or decreasing them.

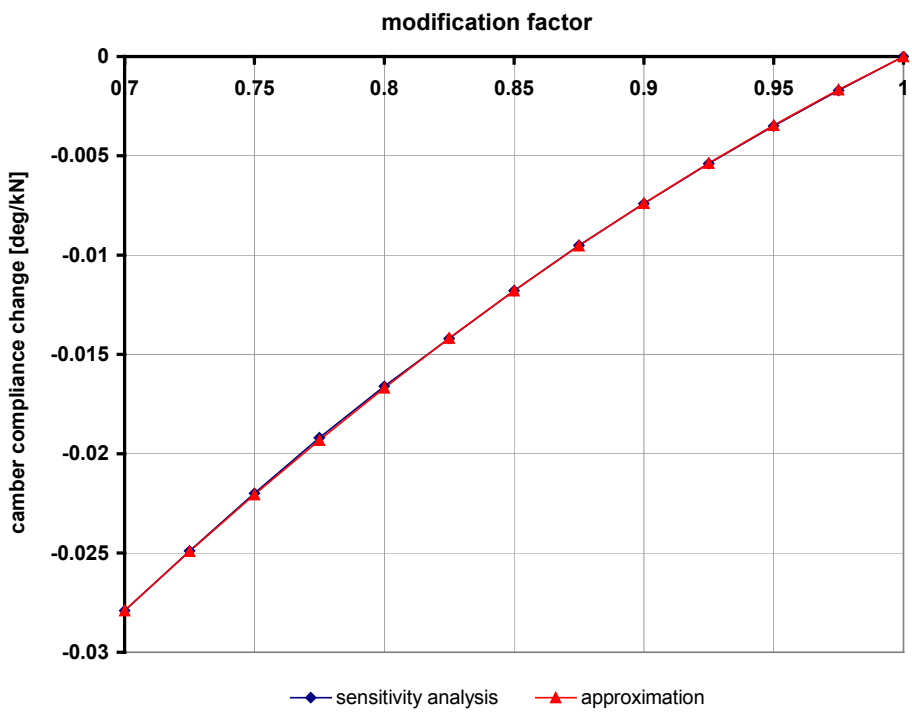


Figure 4.3 – Parabolic approximation of an objective function

Global optimisers are useful when the search space is likely to have many minima, making it hard to locate the true global minimum. In low dimensional or constrained problems it may be enough to apply a local optimiser starting at a set of possible start points, generated either randomly or systematically (for instance at grid locations), and choose the best result. However this approach is less likely to

locate the true optimum as the ratio of volume of the search region to number of starting points increases. Correctly applied the evolutionary search strategies can explore the search space better than a grid search for a given number of function evaluations, and is more likely to find the true global minimum. Notice that both these approaches involve a stochastic element and so may fail to find the true minimum. In addition, since they are better at global searching than local optimisation, it is usually worthwhile to "polish" any final solution using local optimisers such as a gradient based method.

In the case of the GA, they stand out for the easiness with which they can be implemented and the broad spectrum of cases they can handle. Very little information is required a priori and several solutions are tried simultaneously. However, they have the lowest rate of convergence and they present a huge disadvantage: for every single solution at each generation, an evaluation of the objective function must be performed. In the case of a suspension assembly, this means running one or more complete ADAMS/Cars simulations in order to check the constraints, that is, the evaluate if any of the compliance limits has been violated.

On the other hand, gradient based optimisation methods are useful when the search space shape is known and no multiple solutions or local minima are expected. They also are applied to improve the solutions found by global optimisers, when possible. Another approach is to run a quick, not so refined global optimisation method in order to pinpoint local minima, and then run a refined local optimiser; the results are then compared and the best solution chosen. Gradient based methods present, however, one big problem: how to introduce constraints into the optimisation algorithm. When these are included in the objective function as explained in Section **4.1**, there exist a high probability that the smoothness of the function is lost. Dividing by the distance to the limit values of the constraints means that, when the algorithm is close to a local solution, or at least one that achieves the constraints very accurately, the gradients may become very high or very low. Moreover, a solution with a good outcome from the weight standpoint but that violates the constraints can get as good fitness as a solution with a bad objective function result but good behaviour regarding the constraints.

Response Surface Methods allow to replace a complicated or even a partially unknown objective function of a search space by a well behaved function and with good mathematical properties. The optimisation is run over this function and the result is tried in the real objective function.

There is another case in which the use of RSM can be very profitable. When the evaluation of a possible solution is very time consuming and several evaluations must be performed, if the objective function can be approximated accurately enough with a few previous evaluations (randomly generated or by patterns, such as a grid), then RSM provide the answer to assess a possible solution without the need for an expensive evaluation in the real search space by using the objective function. This feature allows increasing the efficiency of the optimisation algorithm.

4.5 Sensitivity studies

As it has been pointed out insightful knowledge of the characteristics of the variation space is a prerequisite for the definition of constraints and objective functions, but –and most important– it is also essential for the choice of an appropriate optimisation method. If this knowledge is not available, sensitivity studies are recommended.

Ceteris paribus studies (varying single parameters at a time and looking on their effect) are a common task for the engineer. Similarly, in small parameter spaces, design of experiment (DOE) methods which systematically calculate single parameters and parameter combinations can be applied. With increasing dimension or non-linearity of the parameter space, stochastic sampling strategies are preferred to generate the sample set.

An additional advantage of the stochastic sampling strategies compared to the design of experiments is that they permit a statistical evaluation in the form of correlation and variation analysis of the sensitivities in the variation space. Consequently, unstable areas in the design space, hints on the variation potentials of the response variables, or global correlation structures showing which optimisation variable has which influence on which response variable, etc., can be identified by means of sensitivity analysis. Furthermore, it can help to determine that some variables, initially assumed to be part of the problem, have little or no influence on the objective function output. In this way, sensitivity studies may permit a reduction of the parameter space for subsequent optimisation tasks. Moreover, the previous knowledge drawn from the sensitivity studies regarding the properties of the design space often is most helpful for an appropriate formulation of the constraints and the objective functions.

4.6 Pareto optimisation strategies

It has been pointed out earlier that optimisation problems may possess more than one objective criterion (multi-criteria optimisation). When these criteria are not conflicting, weight strategies can be used to combine several criteria to one objective function. Mathematically, assigning different weights to a single criteria should only have influence on the convergence speed. In reality, even in cases where the objective criteria are not in conflict, different weights can indeed lead to different optimisation results. The reasons for this may be that a local optimum has been found, the optimal area has a very small gradient, or the solution is not converged.

In case the criteria are in conflict, even from a mathematical point of view, there is no longer one single optimum. Instead, a number of possible compromise solutions exist. In this case, the weights of the different objective criteria influence the optimal compromise in a much stronger way.

As an alternative Pareto optimisation methods can be applied to determine the compromise set. By means of a posterior weighting they permit to choose the optimal compromise solution not before but

after the optimisation. Successful methods of Pareto optimisation generally apply evolutionary methods such as Strength Evolutionary Pareto Algorithms **[4.5]**.

In practice a successful application is restricted to two- and three-dimensional multi-criteria tasks. This is not so much due to a restriction of the underlying algorithms but to the failure of representation possibilities of the compromise solution. However, a proper representation is essential for a good posterior choice.

Taking into account that instead of one optimal point a set of optimal points is to be determined, it is hardly surprising that generally the computational expense is considerably increased. This leads to the recommendation that Pareto optimisation strategies should not be used to start the work on an optimisation task. They should be applied at a work stage when the structure of the optimisation problem (important input and response variables) is well known, and it can be taken for sure that two or three important objective criteria are in conflict.

5 GENETIC ALGORITHM

Evolutionary search strategies are developed to work based on the fitness of the individuals of a population, without taking into account any set of constraints. However, solving *Constraint Satisfaction Problems* (CSPs) involves finding an assignment of values to variables that both improves the output of the function and satisfies a set of constraints **[5.1]**. This general problem has many real-life applications, such as time-tabling, resource allocation, pattern recognition, and machine vision. In the particular case dealt with in this thesis, a set of modification factors to apply to the stiffness matrices, one for every DOF of the condensed system, must be found; the constraints are the allowed camber compliance, toe compliance, etc., under different load cases: braking forces, lateral forces, etc. Usually, the methods developed to deal with CSPs assume that a maximisation case will be tackled, since handling minimisation problems only requires a trivial mathematical transformation **[5.2]**.

To solve CSPs, one may explore the search space in a systematic and complete way, until either a solution is found, or the problem is proven to have no solution. In order to reduce the search space, this kind of complete approach is usually combined with filtering techniques that narrow the variables domains with respect to some partial consistencies. Completeness is a very desirable feature, but it may become intractable on hard combinatorial problems. On the other hand, computational limitations have to be accounted. Too many variables or time consuming simulations (typically involving FE or non-linear mechanic systems) to evaluate the function under optimisation may set boundaries for the sort of algorithm to apply in the optimisation procedure. Hence, incomplete approaches, such as evolutionary and genetic algorithms have been proposed. These approaches differ only in technical details **[5.3]**; they do not perform an exhaustive search, but try to quickly find approximately optimal solutions in an opportunistic way. The search space is explored stochastically using heuristics to guide the search towards the most-promising areas. Therefore, under the constraints applied to the problem, several local solutions are likely to be found, while a post-processing of the results may lead to the best solution. The basic mechanism of these methods is as follows:

> In a formal setting, the environment is represented by a given quality function to be maximised (or minimised). The population is created by randomly generating a set of candidate solutions, i.e., elements of the function's domain, and the quality function is used as an abstract fitness measure → the higher the better. Based on this fitness, some of the better candidate solutions are chosen to seed the next generation by applying recombination and/or mutation to them. Recombination is an operator applied to two or more se

lected candidates (the so-called parents) and results one or more new candidates (the children). Mutation is applied to one candidate and results in one new candidate. Executing recombination and mutation leads to a set of new candidates (the offspring) that compete with the old ones, based on their fitness, for a place in the next generation. This process can be iterated until a candidate with sufficient quality (a solution) is found or a previously set computational limit is reached, e.g., no further improvement is noticed after a given number of generations **[5.2]**.

It is important to note that many components of such an evolutionary process are stochastic. During selection fitter individuals have a higher chance to be selected than less fit ones, but typically even the weak individuals have a chance to become a parent or to survive. For recombination of individuals the choice of which pieces will be recombined is random. Similarly for mutation, the pieces that will be mutated within a candidate solution, and the new pieces replacing them, are chosen randomly.

In the task of optimising a suspension assembly by means of applying factors to the coefficients of the stiffness matrices of it components, the abstract representation of each solution, that is, the vector containing the chromosome, is simply a vector with the value of each factor to apply to the pairs row/column.

The scheme of an evolutionary algorithm is given in **Figure 5.1** in a pseudo-code manner **[5.4]**:

```
BEGIN
    INITIALISE population with random candidate solutions;
    EVALUATE each candidate;
    REPEAT UNTIL (TERMINATION CONDITION is satisfied) DO
        1 SELECT parents;
        2 RECOMBINE pairs of parents;
        3 MUTATE the resulting offspring;
        4 EVALUATE offspring;
    END
END
```

Figure 5.1 – General scheme of an evolutionary algorithm in pseudo-code

As it can be seen, in any Genetic Algorithm implementation there are three basic operators:

- reproduction,
- crossover,
- mutation.

In the next paragraphs, the analogy with the biological background is explored. The evaluation of the individuals of a population and the three operators are presented and analysed for a thorough understanding of their modus operandi, as well as the contribution of each one of them to the optimisation algorithm. Lastly, some particularities about the management of the population and the details about the GA implementation and programming are introduced and discussed.

5.1 Biological background

Every organism has a set of rules, a sort of blueprint, describing how that organism is built up from the tiny building blocks of life. These rules are encoded in the genes of an organism, which in turn are connected together into long strings called chromosomes. Each gene represents a specific trait of the organism, like eye colour or hair colour, and has several different settings. For example, the settings for a hair colour gene may be blonde, black or auburn. These genes and their settings are usually referred to as an organism's genotype. The physical expression of the genotype - the organism itself - is called the phenotype. The environment, as the sum of all the threats to the organism survival probabilities, sets the fitness of the organism.

When two organisms mate they share their genes. The resulting offspring may end up having half the genes from one parent and half from the other. This process is called recombination. Very occasionally a gene may be mutated. Normally this mutated gene will not affect the development of the phenotype but very occasionally it will be expressed in the organism as a completely new trait. The process starts over again since the new born organism is released to the environment and assessed. Such a process is schematised in the loop presented in **Figure 5.2**.

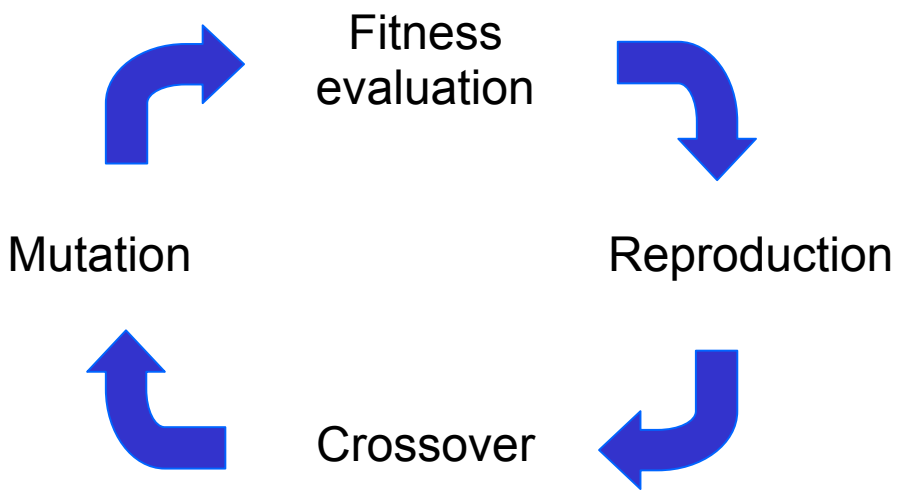


Figure 5.2 – Natural evolution process

Life on earth has evolved to be as it is through the processes of natural selection, recombination and mutation. In the same way, any phenomenon which depends on an encoded string and subjected to changes in the string that might lead to changes in the behaviour or expression of the phenomenon, is possible to under go an optimisation process based on this mechanism implemented by nature with living creatures. The three operators, reproduction, recombination and mutation, together with the assessment (in the case of nature, the own environment) are present and exploitable. However, and unlike nature, the parameters governing each of the four aspects of the process can be tuned in order to increase the pace at which evolution, hence perfecting of the population, takes place. Given certain conditions, an optimum can be attained, although the simple improvement is already a very desirable goal in the engineering field, being sometimes enough to gain little steps for a given investment of time and effort, rather than attaining the absolute perfect individual.

5.2 Representation

Similarly to nature, where the fittest individuals of a species possess the highest chances of survival, so genetic algorithms work. The very foundation stone of the genetic algorithms to perform the optimisation process is the string containing the genes or data that defines an individual: the chromosome. In the case of a suspension assembly expressed in terms of its stiffness matrices, and for the purposes of fitting with the optimisation schema, the chromosome is a vector or line matrix containing the modification factors to apply to each of the pairs row/column of the stiffness matrices.

As it was mentioned in Section **2.4**, the modification factors are allowed to take values between a lower limit set by the user and 1, and they represent the stiffness of the component, hence related with its weight.

5.3 Evaluation

Two elements are required for any problem before a genetic algorithm can be used to search for a solution: first, there must be a method of representing a solution in a manner that can be manipulated by the algorithm. Traditionally, a solution can be represented by a string of bits, numbers, characters or by a special structure (as seen in the previous paragraph). Second, there must be some method of measuring the quality of any proposed solution, using a so called fitness function.

In this context, a fitness function is a particular type of objective function that quantifies the optimality of a solution (that is, a chromosome) so that that particular individual may be ranked against all the other individuals in the population in that generation. Optimal chromosomes, or at least chromosomes which are better, are allowed to breed and mix their genes by any of several techniques (crossover, mutation) producing a new generation that is hopefully better.

An ideal fitness function correlates closely with the algorithm's goal, and yet may be computed quickly. Speed of execution is very important, as a typical genetic algorithm must go through a high number of iterations in order to produce a useable result for a non-trivial problem.

As discussed in Section **4.2**, the objective function represents the weight of the components in the assembly, defined as the average of all the modification factors (parameters) applied to the stiffness matrices. The need to introduce constraints in the algorithm due to targets for the suspension assembly (compliances) prompts to include them in the objective function, giving rise to the following fitness function (also known as *penalised objective function*):

$$\text{Fitness} = \frac{1}{\overline{\text{factor}} \cdot \text{error}} \tag{5.1}$$

where

$$\overline{\text{factor}} = \frac{\sum_{1}^{n_f} \text{factor}_i}{n_f} \tag{5.2}$$

$$\text{error} = \sum_{1}^{n_c} \sqrt{\left(\frac{\text{compliance}_{ref} - \text{compliance}_i}{\text{compliance}_{ref}} \right)^2} \tag{5.3}$$

That is, $\overline{\text{factor}}$ as in (5.2) represents the average of all the modification factors, where n_f is the number of modification factors or parameters for the optimisation. Conversely, error as in (5.3) is the sum of the absolute values of the normalised differences between each compliance of the modified individual and the reference compliances, where n_c is the number of constraints or measurements to take from the suspension assembly, i.e., toe compliance, camber compliance, etc.

In this way, the fitness function given in (5.1) effectively represents the quality of a proposed solution. The higher the weight of the proposed individual, that is, the higher the average value of the factor applied to the stiffness matrices of the components in the suspension, the lower it turns up to be the fitness. At the same time, when such individual has a performance too far away from the targets the error is increased, lowering the fitness.

5.4 Reproduction

During each successive generation a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population as this process may be very time-consuming.

Most functions are stochastic and designed so that a small proportion of less fit solutions are selected. This helps keep the diversity of the population large, preventing premature convergence on poorer, local solutions. Popular and well-studied selection methods include roulette wheel selection, tournament selection and elitism.

5.4.1 Roulette wheel selection

Parents are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have. If the individuals and their fitness are arranged in a pie chart, it can be seen as a roulette wheel (**Figure 5.3**), where the size of the sections are proportional to the value of the fitness function of every chromosome.

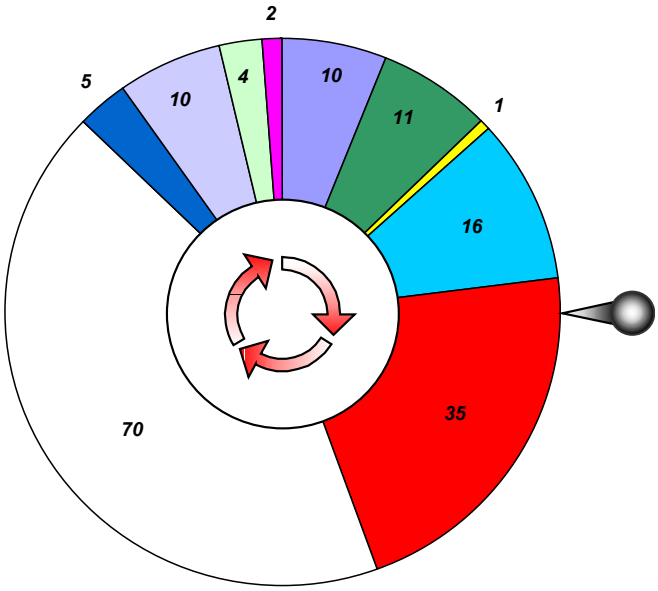


Figure 5.3 – Population's individual chromosomes arranged for the roulette wheel

The roulette wheel is turned and the chromosome corresponding to the final position is selected. Clearly, the those with higher fitness value will be selected more times. The process works as follows:

1. Sum: calculate the sum of all chromosome fitness in population (called *S*). This step is performed only once for each population.
2. Select: generate random number from the interval 0 to *S* (called *r*).
3. Loop: go through the population and summing the fitness. When the sum is greater than *r*, stop and return the chromosome.

5.4.2 Tournament selection

This selection method involves picking a number of individuals at random from the population to form a "tournament" pool. The two individuals of highest fitness from this tournament pool are then selected as parents. The target of this method is to speed up the process of selecting the parents for the next generation. However, it in fact overlooks some individuals that might have a higher fitness that those selected for the tournament pool. This method can be useful if the fitness function is very difficult to calculate, time consuming, or when the population is too large.

5.4.3 Elitism

The idea of the elitism is that, when creating a new population by randomly choosing the parents, there is a chance that the best chromosome will be lost. Elitism is the name of the method that first copies the best chromosome (or few best chromosomes) to the new population. The rest of the population is constructed in ways described above.

An inverted elitism can also be implemented, where the worst chromosome (or few worst chromosomes) of a population are deterministically eliminated before choosing the parents.

Typically, elitism can rapidly increase the performance of GA because it prevents a loss of the hitherto best found solution.

5.5 Crossover

Crossover is a genetic operator that combines or mates two chromosomes (parents) to produce a new chromosome (offspring). The idea behind crossover is that the new chromosome may be better than both parents if it takes the best characteristics from each of them.

The crossover mechanism trades blocks of genes between individuals, allowing the blocks to fully exploit a particularly profitable portion of the parameter space. A particular code sequence, called a *schema* or *building block* (for example, the first four positions of a chromosome) drives a specific parameter or parameter set (such as toe compliance) into a certain range. It will do this no matter what the other values within the chromosome are. Therefore, by combining this *schema* with different values of the rest of the chromosome the parameter space around those values can be thoroughly exploited.

5.5.1 One-point crossover

As it can be seen in **Figure 5.4**, a crossover point on the parent organism string is selected. All data beyond that point in the organism string is swapped between the two parent organisms. The resulting organisms are the offspring:

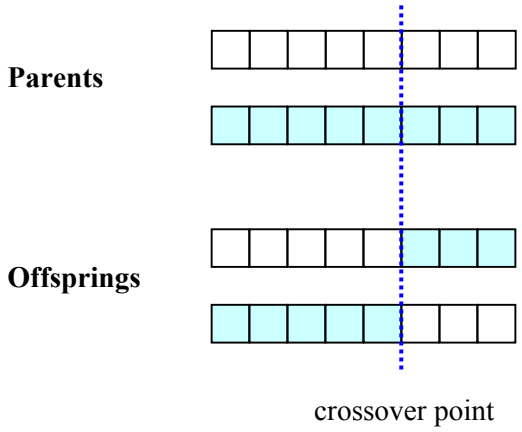


Figure 5.4 – One-point crossover

5.5.2 Two-points crossover

Two-points crossover calls for two points to be selected on the parent organism strings. Everything between the two points is swapped between the parent organisms, rendering two offspring organisms, as shown in **Figure 5.5**.

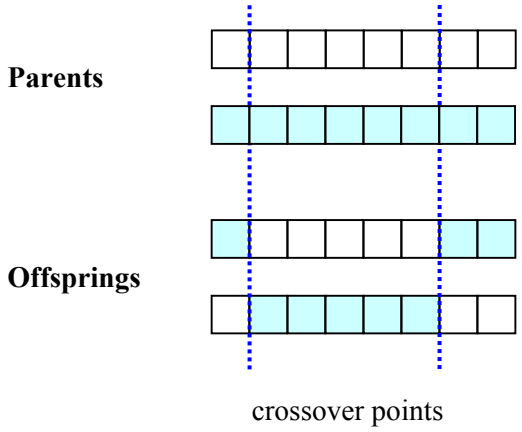


Figure 5.5 – Two-points crossover

This procedure is equivalent to see at the chromosome as in a circular arrangement as in **Figure 5.6**, where slices of the circles are exchanged.

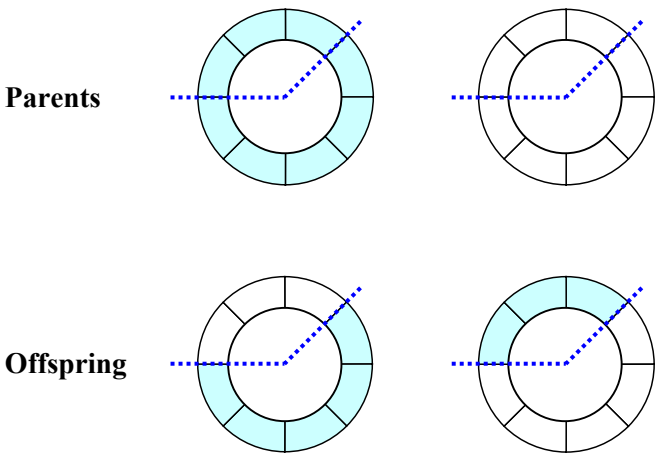


Figure 5.6 – Two-points crossover (circular representation)

The reason to do a two-points crossover instead of the simpler one-point crossover lays in the following statement: among the population of chromosomes, there might be useful strings of genes (called schema, plural schemata)to propagate. It is utterly obvious that a one-point crossover mechanism is likely to break up those strings whose elements lay at the ends of the chromosome. Those schemata grouped in the middle of the chromosome have better chances to survive since the crossover point is equally likely to fall between two given genes. For example:

- the schema **101*** (where the * can be any value) has 8 genes and 7 possibilities for the crossover point, but only 2 of them (28.57%) would destroy the schema: **1|01*** and **10|1***, where | is the crossover point. For the two-points crossover method, there are 28 different possible combinations, of which 5 would destroy the schema (17.86%).
- if the same schema is present but arranged next to the ends of the chromosome (1*****10) it makes no different where one crossover point falls; it always destroys the schema. But the two-points crossover method has again just 17.86% chances to break up the schema.

While much of the work is done by the sequential *building blocks*, the crossover effect can be seen with non-sequential *schemata* as well. For example, the binary sequence 11**1*01 is also exploiting a specific area of parameter space, but it is more likely to be broken up, whether one- or two-points crossover method is used. However, although there are indeed cases in which a two-points crossover worsens the possibilities, in the vast majority of them the chances for a schema to survive improve.

For a binary coding and a chromosome of length l, there are 2^l simultaneous schemata being tested by the GA, and the ones which contribute the most to increasing the fitness of the individual will survive and reproduce. This is known as “implicit parallelism”, and is one of the features that gives GAs their power **[5.4]**.

5.5.3 Uniform crossover

A certain number of genes are randomly selected (marked with ▼ in **Figure 5.7**) to be "swapped".

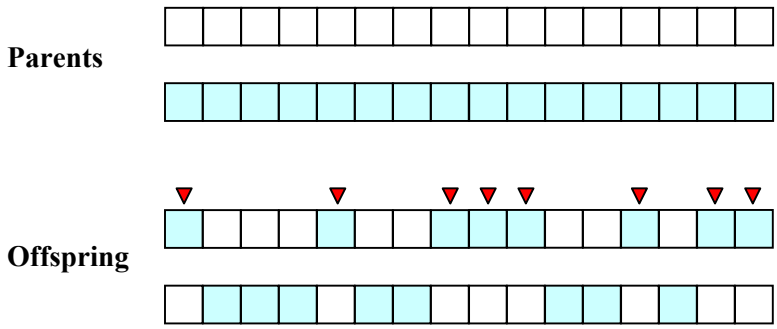


Figure 5.7 – Uniform crossover

Obviously, the main disadvantage of this method is that it does not allow schema survival. This may not have importance for certain code methods or optimisation problems, but when schemata are present it definitely means not using their potential. Besides this disadvantage, this method offers no particular advantage over the others except maybe its simplicity to implement.

5.5.4 Half uniform crossover

A variation of the previous method can be implemented to avoid the destruction of the schema and exploit its potential. First, the Hamming distance[4] **[5.5]** between two parents is calculated. This number is divided by two. The resulting number determines how many of the bits that do not match between the two parents will be swapped.

As an example, in **Figure 5.8** two chromosomes are depicted where the non matching bits are marked with ▼. It can be seen that the Hamming distance is 8; therefore, 4 bits (left marked with ▼ in **Figure 5.9**) are swapped to form the offspring. These 4 bits are randomly chosen among those non matching 8 bits.

[4] How different two individuals are can be measured by the *Hamming distance*, which can be defined as the total number of positions in the chromosomes (of equal length) where the genes differ from one individual to the other. When the representation is binary, the typical Hamming distance for the first generation (randomly generated individuals) is half the length of the chromosome. If the encoding is not binary, then the Hamming distance is almost the same as the length of the chromosome.

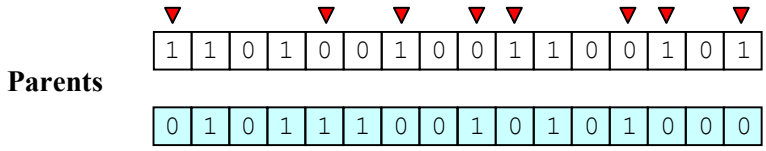


Figure 5.8 – Non matching bits

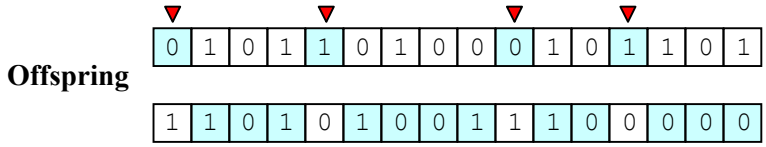


Figure 5.9 – Swapped bits

5.6 Mutation

Mutation is used to maintain genetic diversity from one generation to the next one. The classic example of a mutation operator involves a probability that an arbitrary bit in a genetic sequence will be changed from its original state. The purpose of mutation is to allow the algorithm to avoid local minima by preventing the chromosomes from becoming too similar to each other, thus slowing or even stopping evolution. This analysis also explains the fact that most GA systems avoid only taking the fittest of the population in generating the next in a deterministic way, but instead use selection methods with a weighting system that only bias the opportunities toward those individuals that are fitter.

5.7 Termination

The generational process is repeated until a termination condition has been reached. For example:

- a solution is found that satisfies minimum criteria,
- fixed number of generations reached,
- allocated budget (computation time, cost) reached,
- the highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results,
- manual inspection,
- combinations of the above.

5.8 Process parameters

Regardless of the problem, in every implementation of a genetic algorithm the operators regulating the population, selection of the individuals and variations have to be arranged. Following, some common parameters **[5.6]** or characteristics are discussed.

5.8.1 Population management

Population size is the term that describes how many chromosomes (individuals) are in a population in one generation. A small population allows faster progress but increases the risk of getting trapped in a local optimum because it can only maintain fewer alternatives, hence less diversity. In such a case, there are too few chromosomes and the algorithm has few possibilities to perform crossover. In consequence, only a small part of the search space is explored. On the other hand, if there are too many chromosomes, GA slows down. Research shows that after some limit (which depends mainly on encoding and the problem) it is not useful to use large populations because it does not evolve faster than moderate sized populations.

At first, the individuals are generated randomly and the average fitness of the population is rather low but, more important, the chromosomes from individual to individual are very different. As the population evolves, the individuals start becoming fitter and more similar to each other. Bearing this in mind, it makes no sense trying to keep a large population through out the whole process since it would do nothing but slower it. The algorithm is started with a fairly large population instead, which consistently decreases from generation to generation to a lower limit.

5.8.2 Scaling

At the beginning of the genetic optimisation process, that is in the first generations, the population is quite diverse and its individuals can present wide differences in fitness. This induces the algorithm to early convergence, increasing the risk of falling into a local solution, due to hyper-fit individuals. Conversely, towards the end of the process the population is fairly uniform and its individuals show similar values of fitness. This makes it difficult for the slightly better individuals to be picked up by the selection during the reproduction step, slowing or even stopping the evolution process.

A way to address this is by scaling the fitness. If the ratio between the mean fitness (f_{mean}) and maximum fitness (f_{max}) through out a population at a given generation is close to zero (frequently during the first few generations), it means high diversity and the fitness should be equalised to some degree. On the other hand, if the ratio is close to 1 (at the end of the process or when the algorithm is stuck in a plateau of the search space, where all the individuals tend to become alike after a few generations) the values of the fitness should be differentiated. **Figure 5.10** shows an example run of a genetic algo

rithm during 500 generations, used to optimise the modification factors of a suspension assembly. The ration between the mean and the maximum fitness is tracked along the generations. It can be seen how the population progresses from a highly diverse one to a situation in which most individuals are rather similar, after approximately 150 generations.

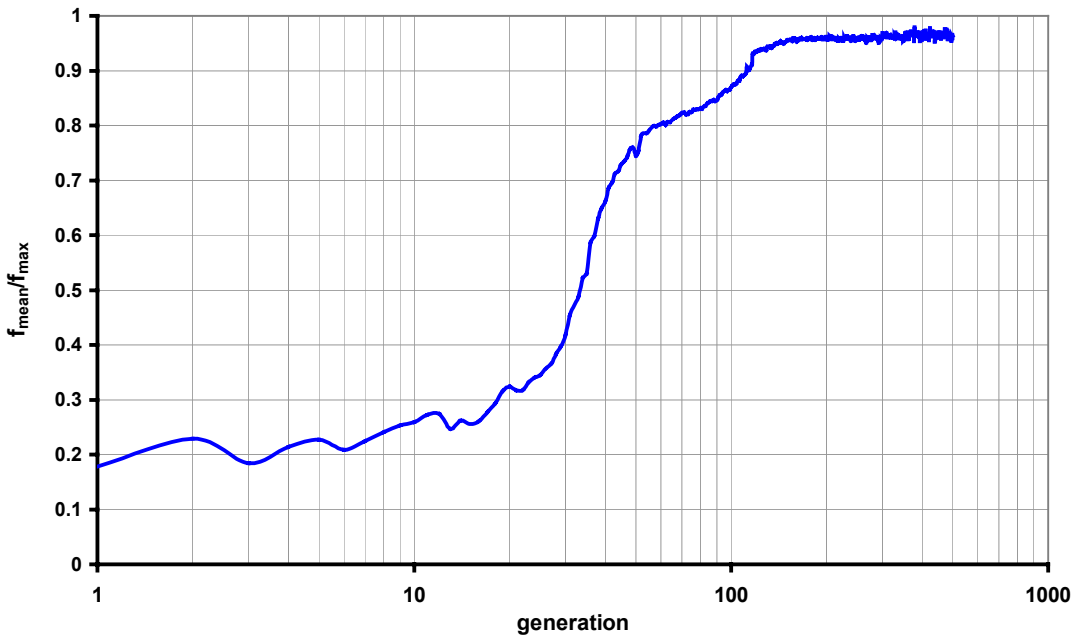


Figure 5.10 – Ratio between mean and maximum fitness in a genetic algorithm run

A method to differentiate the individuals is by the use of exponential scaling, according to the following equation:

$$\textit{scaled}\ \text{fitness}_i = \text{fitness}_i^{(k \cdot n)} \tag{5.4}$$

$$\text{being}\ n = \frac{\overline{\text{fitness}}}{\text{fitness}_{max}} \tag{5.5}$$

where

fitness_i = fitness if individual i

$\overline{\text{fitness}}$ = mean fitness

fitness_{max} = maximum fitness

The factor k is a problem-dependent number that determines the limit of what is called a homogeneous population. If $k = 2$ the scaled fitness takes the same value of the fitness, if the ratio between the mean and the maximum fitness is 0,5 ($= k^{-1}$). If an individual has a fitness value higher than k times

the mean value of the population, it will considered as a hyper-fit element and therefore be evened. Conversely, if the best individual of a population has a fitness value lower than k times the mean value of the population, this will be considered homogeneous and therefore its fitness scaled in order to increase the differences among its individuals.

Another way to prevent that hyper-fit elements take over the population on the beginning of the process, but at the same time avoiding stagnation of the algorithm at the end of it (where the gene pool is rather uniform), is by linearly normalising the fitness of the population, also called *ranking*. For example over a population of 5 subjects, the fittest will get 100, the second 80, and so forth. The last will get 20. Even if the differences between the subjects are very strong, or weak, the difference between probabilities of reproduction only depends on the ranking of the subjects **[5.7]**.

In **Tables 5.1** and **5.2** the effect of both methods in a 5 individuals population is shown ($k = 2$):

individual	fitness	probability	scaling		scaled probability	
			exponential	ranking	exponential	ranking
#1	1	0.91%	1.0000	5	7.46%	6.67%
#2	2	1.82%	1.3566	4	10.12%	13.33%
#3	3	2.73%	1.6216	3	12.10%	20%
#4	4	3.64%	1.8404	2	13.73%	26.67%
#5	100	90.91%	7.5858	1	56.59%	33.33%
max/min		*100*			*7.59*	*5*

Table 5.1 – Effect of scaling for a hyper-fit element in a population

individual	fitness	probability	scaling		scaled probability	
			exponential	ranking	exponential	ranking
#1	98	19.60%	8023.52	5	19.22%	6.67%
#2	99	19.80%	8184.84	4	19.61%	13.33%
#3	100	20.00%	8347.73	3	20.00%	20%
#4	101	20.20%	8512.20	2	20.39%	26.67%
#5	102	20.40%	8678.24	1	20.79%	33.33%
max/min		*1.04*			*1.08*	*5*

Table 5.2 – Effect of scaling for a homogeneous population

As it can be noticed, in a given population there can be a hyper-fit individual that could take most of the chances to survive to the next generation, not allowing potential areas of the search space to be further investigated by the algorithm. Homogeneous populations also stop the algorithm from progressing. As shown in the tables, both scaling methods deliver a smoother fitness profile of the popu

lation for the first case, while enhancing the differences when the individuals present similar fitness values, more so with the ranking method.

5.8.3 Selection pressure

When applying the crossover operator, it is not compulsory to erase all the parents for the next generation. In fact, it may occur that the reproduction, crossover and mutation operators produce no individual at all with improved fitness through out the population. Therefore, it is always advisable to keep some of the fittest parents from one generation to the next one. The extent of bias preferring the good candidates over weak ones is called selection pressure. If the new generation consists of 75% offspring and 25% parents, it is said that there is a selection pressure of 4, according to the following definition:

$$\text{selection pressure} = \frac{\text{total population}}{\text{surviving parents}} \tag{5.6}$$

If the selection pressure is too low, only a few or none new offspring are introduced in the population, which inherently slows the evolution process by allowing the parents with lesser fitness to survive from generation to generation. Essentially, it degrades evolutionary search into random walk.

On the other hand, if most of the parents were eliminated from one generation to the next one, it would cause faster progress but with two risks:

- useful schemata can be erased from the population, requiring new generations to recreate it,
- it increases the chances that the algorithm gets stuck in a local optimum by being too greedy.

5.8.4 Mutation rate

The classic example of a mutation operator involves a probability that an arbitrary gene in a chromosome will be changed from its original state. A method to do so involves generating a random variable for each bit in a sequence that tells whether or not a particular bit will be modified. Another method randomly chooses both an individual and a position in the chromosome, and mutates the corresponding gene. The number of iterations of this process over the population in relation with the size of the population in a given generation determines the mutation rate. Naturally, it makes sense to keep a low mutation rate during the first stages of the algorithm where the population is diverse, and increase it towards the end where the individuals are more similar to each other. However, using too high mutation rate speeds up the search but can prevent fine tuning on the optimum **[5.3]**. A good way to dynamically adapt the mutation rate to the process requirements is by keeping track of the fitness of the population in one generation. If the ratio between the average and the maximum fitness is too low (close to zero), it can be assumed two things:

- there are some individuals (those with the maximum fitness) with a fitness that prompts the population towards change and, hopefully, evolution and improvement,
- the population is diverse enough to cover potentially good areas of the search space.

If the ratio is close to 1 it means that the individuals with the best fitness are actually too similar with the rest of the population, so an increased mutation rate introduces the necessary changes to avoid the stagnation of the algorithm.

In addition, when the size of the population is being decreased from generation to generation, the mutation rate must be correspondingly reduced to avoid noise and improve the fine tuning of the solution in the last stages of the optimisation process.

These considerations lead the following schematic equation to regulate the mutation rate:

$$\text{mutation rate} = k \cdot P \cdot Ch \cdot \left(\frac{\overline{\text{fitness}}}{\text{fitness}_{max}} \right)^n \tag{5.7}$$

where

P = population size

Ch = chromosome length (number of optimisation parameters)

$\overline{\text{fitness}}$ = average fitness

fitness_{max} = maximum fitness

n = geometry factor, which regulates how aggressively the mutation rate must be increased when the population diversity falls

k = proportionality constant to scale the mutation rate

When the population is in the early generations the diversity is high enough to allow keeping the mutation rate low, but when the algorithm has already focused on some areas of the search space a typical value is 0.001 (1 mutated gene every 1000), which can be increased to 0.01 or even higher towards the end of the process.

5.8.5 Random immigrants

During the genetic optimisation process, individuals begin to converge early on towards solutions which may be local optimums. Several methods have been proposed to solve this by somehow increasing genetic diversity and preventing early convergence, either by increasing the probability of mutation when the solution quality drops (called *triggered hypermutation*), or by occasionally intro

ducing a small number (1% of the population) of entirely new, randomly generated elements into the population, also known as *random immigrants*.

5.9 Refining the solution

Given that the genetic algorithm is a stochastic process, the solution may not be the optimal even for a well designed and implemented process. It is therefore always convenient to run the algorithm several times in order to obtain alternatives to choose from. However, even after several runs the algorithm produces no further improvement. In such a case, it is possible to re-initiate the genetic algorithm, this time allowing the solution from a previous run to be introduced early in the process as an immigrant. Since the objective is to give the new run just a hint of where to look, it is important to keep a ratio of 0.01 or less on the number of immigrants to the total population, to avoid the new process to inevitably converge to the previously attained solution. The goal of this is actually to force the algorithm to try new variations of an already good solution, in case that such solution was found in a region of the search space where small changes gave no improvements, such as a plateau. Therefore, new, bigger changes over a good solution may lead to better results, although this is not guaranteed.

6 SENSITIVITY ANALYSIS

In order to reduce the size of the problem the number of parameters that must be handled during the optimisation process must be decreased. For this an important assumption is made: the gradient of one DOF (its influence on the performance of the suspension) is not affected by modifications over other DOF. What really matters is that the modification of some DOF does not alter the global behaviour or significantly vary the magnitude of influence of some other DOF, therefore downgrading the quality of the result or invalidating the method. From a mathematical standpoint it is senseful to be concerned about this phenomenon. The next two examples show cases where the assumption is false:

- when DOF *a* is weakened by 30%, the gradient of DOF *b* changes from positive to negative,
- before the DOF *a* is weakened by 10%, the DOF *b* decreased the compliance significantly; afterwards it is irrelevant.

Nevertheless, when analysing real problems there is a handful of DOFs that strongly dominate the performance of the assembly. The behaviour of these DOFs is, in turn, fairly independent from the unimportant DOFs as well as from each other, that is, they are not mutually affected, at least not substantially. Further more, and what is more important: those DOFs that were important during the sensitivity analysis will still be so after all the unimportant DOFs were multiplied by the default modification factor during the first step of the optimisation process (called minimisation).

As a starting point only one component of the assembly (**Figure 6.1**) will be optimised: a left upper control arm containing 3 hardpoints (**Figure 6.2**).

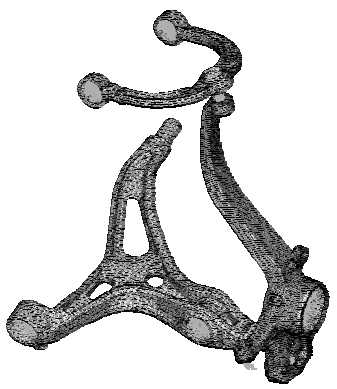


Figure 6.1 – Suspension assembly

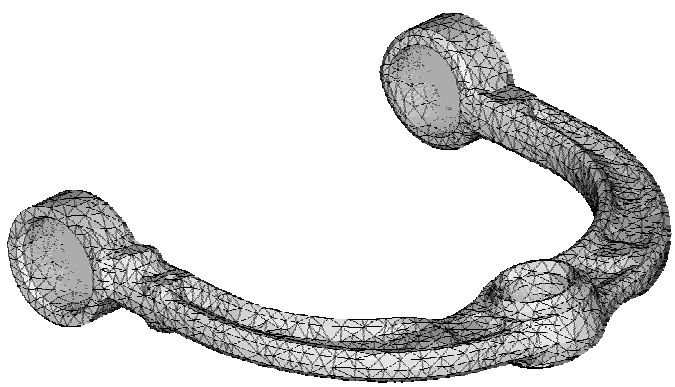


Figure 6.2 – Upper control arm

Initially, an interactive simulation in ADAMS is performed with the original .mnf file. Next, 18 more simulations are run in batch mode for the 6 DOF of each of the 3 hardpoints, each one of those perturbed its stiffness by a *default modification factor*, for example, 0.7. Once the requests are read and the compliance for each of the 19 simulations is calculated and stored, the output of this is a reference value and a table with 18×2 elements (as shown in **Table 6.1.a**) where the first column is the DOF index and the second column the camber compliance change when that DOF is perturbed.

As shown in **Table 6.1.b**, the DOFs are rearranged by decreasing order of camber compliance change, so the upper left cell specifies the DOFs that most increases the camber compliance value; on the other hand, the lower left cell specifies the DOFs that most decreases the camber compliance value. Since the reference value for this measurement is negative (due to the sign convention chose in ADAMS/cars), the table shows that DOF 1 can be weakened and the compliance of the suspension is actually decreased, while weakening DOF 14 means increasing the compliance.

DOF	Δcamber
1	6.88E-05
2	-1.32E-04
3	-5.73E-06
4	5.73E-06
5	-0.00E+00
6	5.73E-06
7	4.01E-05
8	-0.00013751
9	1.15E-05
10	-0.00E+00
11	5.73E-06
12	1.72E-05
13	5.73E-06
14	-0.000573
15	-5.73E-06
16	5.73E-06
17	-0.00E+00
18	5.73E-06

a

sort

DOF	Δcamber
1	6.88E-05
7	4.01E-05
12	1.72E-05
9	1.15E-05
4	5.73E-06
6	5.73E-06
11	5.73E-06
13	5.73E-06
16	5.73E-06
18	5.73E-06
5	0.00E+00
10	0.00E+00
17	0.00E+00
3	-5.73E-06
15	-5.73E-06
2	-1.32E-04
8	-0.00013751
14	-0.000573

b

Table 6.1 – Camber compliance change (a), sorted (b)

By arranging the values of **Table 6.1.b** as in **Figure 6.3**, it becomes clear which DOF in the component can be weakened without affecting the measurement (those in the middle of the chart), which DOF make the suspension assembly more elastic (those to the right, since they rise the magnitude of the compliance) and which DOF stiffen of the suspension assembly (those to the left, that decrease the magnitude of the compliance).

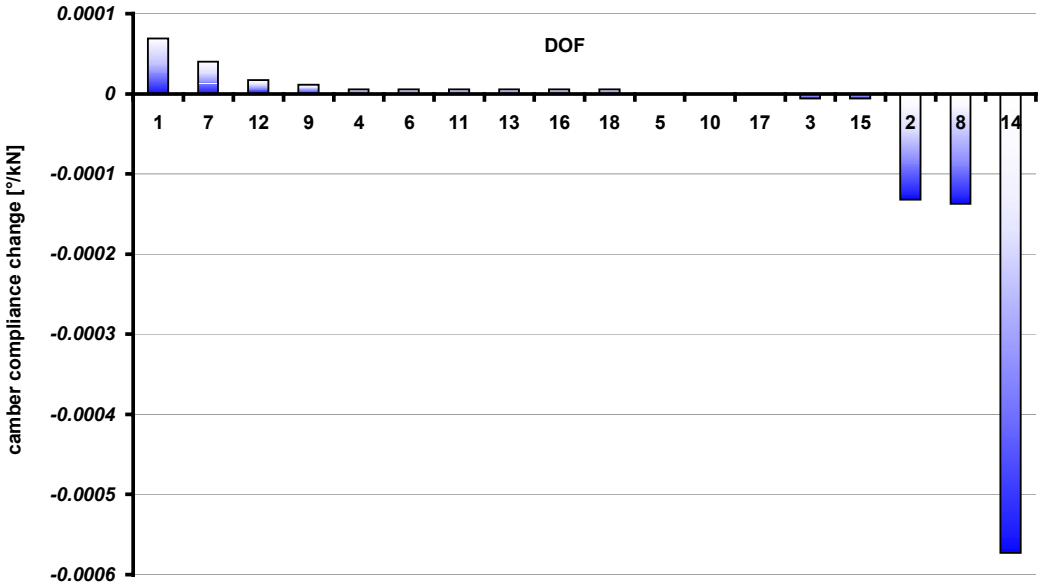


Figure 6.3 – Camber compliance change when the upper control arm is weakened, sorted

When the optimiser has to deal with more than one component, the process can include an intermediate step after the interactive simulation to get the reference value and before the sensitivity analysis. This step allows narrowing down the unknowns and focusing on the components with greater influence over the camber compliance. In this case, the complete stiffness matrix of each of the components must be weakened by the default modification factor. Then a simulation must be carried out for each of the components and the camber compliance change read. Once again the output is a matrix with $n \times 2$ elements, where n is the number of components: the first column stores the component ID and the second column stores the camber compliance change when that component is weakened. The lines are rearranged by decreasing camber compliance change so the upper left cell specifies the component with greater influence on the camber compliance. On the other hand, the lower left cell specifies the component with less influence on the camber compliance. Therefore, the next step focuses only on the significant components, working on the DOF level within the stiffness matrix.

In other words, what the optimiser does is, firstly, to pinpoint which components have more influence on the camber compliance, and from those, with the help of the sensitivity analysis described before, which DOF. However, this method does have a disadvantage: when modifying at once the whole stiffness matrix of a flexible body, the DOF within a component can present a behaviour that counteract each other, masking the real effect of some DOF in particular. That is, there might be cases where some DOF in the component produce an increment of the measurement while others decrease it, the overall result being that the component seems to have no influence on the measurement and therefore being dismissed from the optimisation process.

A second way to analyse an assembly with multiple flexible bodies is by analysing each DOF in the assembly, regardless of the component they belong to. It not only addresses the problem explained before, but it also allows the optimisation of the assembly as a whole and taking into account the interaction between components. The disadvantage is the need of running a sensitivity analysis that includes all the DOF in the assembly, which is more time consuming.

In order to highlight what this means and the differences between both methods, a suspension assembly with three components is considered (**Figure 6.4**), which consists of:

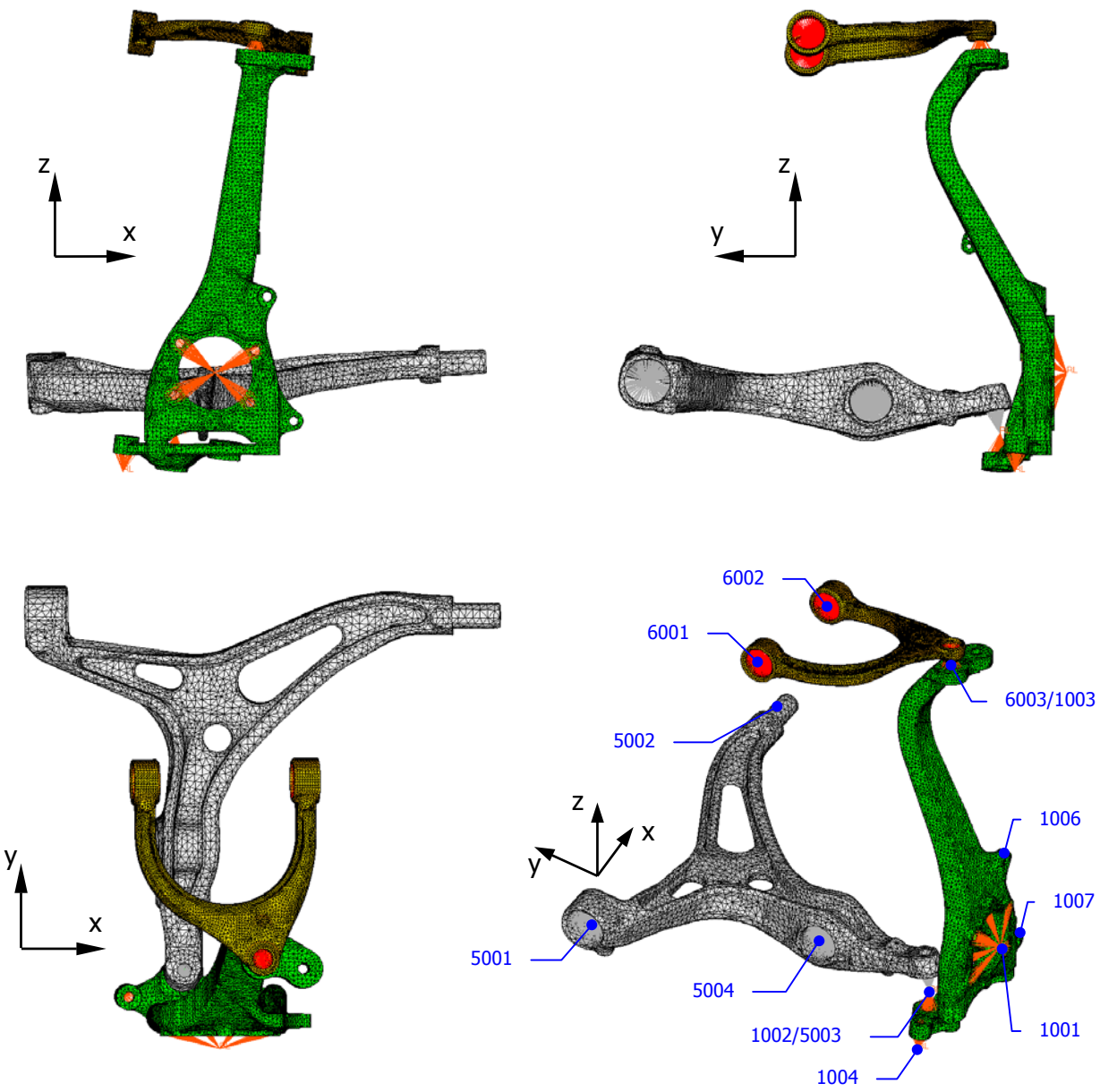


Figure 6.4 – Suspension assembly with a total of 78 DOF

- component 1 – lower control arm, 4 hardpoints
  - 5001: inner front,
  - 5002: inner rear,
  - 5003: outer,
  - 5004: arm to strut.
- component 2 – upper control arm, 3 hardpoints:
  - 6001: inner front,
  - 6002: inner rear,
  - 6003: outer.
- component 3 – wheelcarrier, with 6 hardpoints:
  - 1001: wheelcarrier to wheel,
  - 1002: lower,
  - 1003: wheelcarrier to upper control arm,
  - 1004: wheelcarrier to steering link,
  - 1006: wheelcarrier to brake calliper, upper,
  - 1007: wheelcarrier to brake calliper, lower.

These three components have a total of 13 hardpoints, adding 78 DOF. The first method presented above involves running 3 simulations for the first stage of the sensitivity analysis, one for each component. The results in **Figure 6.5** show that the upper control arm has barely any influence in the performance of the suspension, accounting only for 0.67% of the camber compliance[5], compared with the sum of the three components, while the lower control arm contributes with 6.31%. Therefore, the sensitivity analysis focuses only on the wheelcarrier; this means adding 36 simulations to the second stage of the process, making a total of 39 simulations.

With the second approach the sensitivity analysis takes 78 simulations, one for each DOF in the system. The relevant results of such analysis are shown in **Figure 6.6**. This way it is quite evident that although the wheelcarrier has the biggest influence on the camber compliance behaviour, there exists a DOF or two that belong to the other two components that also play a roll.

[5] The camber compliance's original value is –0.193946 °/kN, which means that every negative value in the sensitivity analysis actually increases the magnitude of the compliance, while a positive value decreases it.

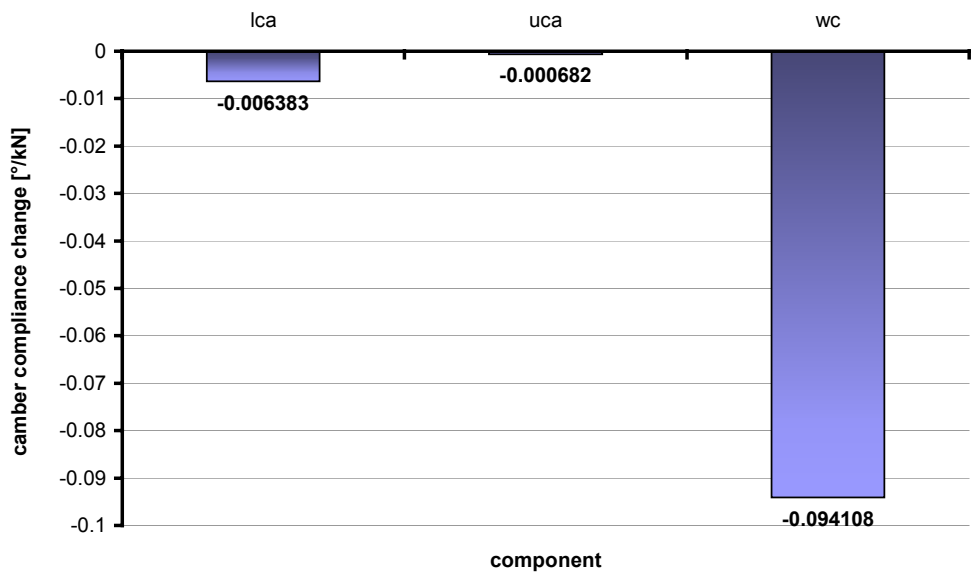


Figure 6.5 – Camber compliance change, by component

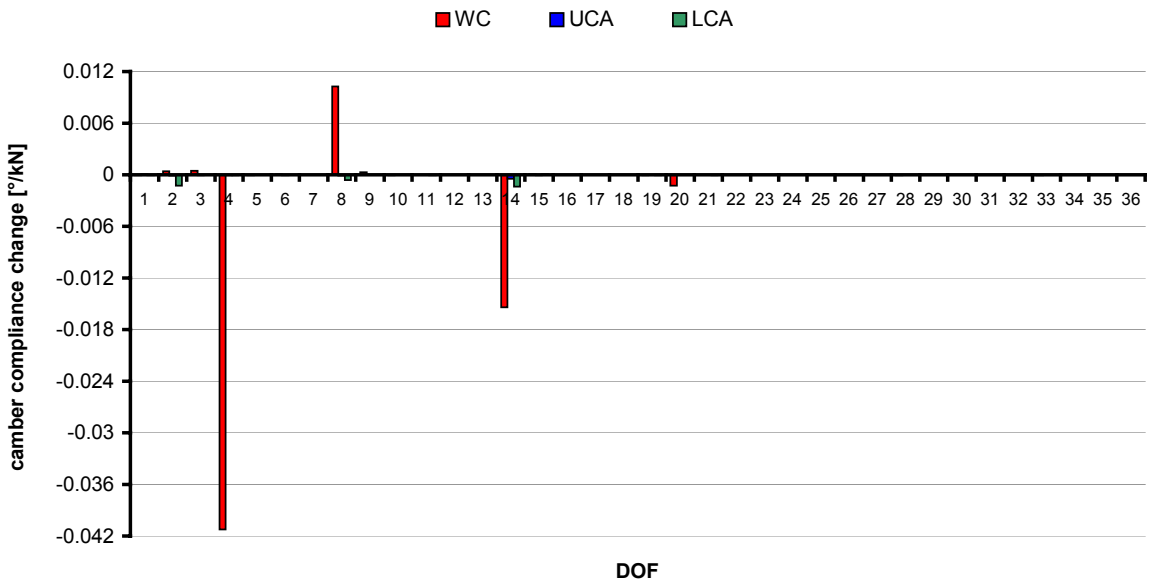


Figure 6.6 – Camber compliance change, by DOF

The negative sign on the sensitivity of some DOF shows that the magnitude of the camber compliance is inversely proportional to the stiffness of the components, so a decrement in the optimisation factor (which weakens that DOF) leads to an increment in the magnitude of the camber compliance. Therefore, when the DOF is optimised, if the camber compliance is higher than the target, the modification factor must be increased in order to stiffen the DOF and lower the camber compliance. On the other

hand, if the camber compliance is lower than the target the modification factor must be decreased, in order to weaken the DOF and raise the camber compliance.

The results of the camber compliance change induced by every DOF are sorted in ascendant order for **Figure 6.7**, where only the 15 most significant DOF are charted: the 10 DOF to the left of the division are those that, when weakened, make the suspension more compliant, and the 5 DOF to the right of the division that, when weakened, decrease the compliance of the assembly. There are two DOF that are clearly predominant over the rest: DOF 4 and DOF 14 of the component 3 (wheelcarrier). However, the chart shows as well that both control arms also manage to have some influence on the measurements. The wheel carrier contributes with three DOF to this list, while the lower and the upper control arms with four and three DOF respectively.

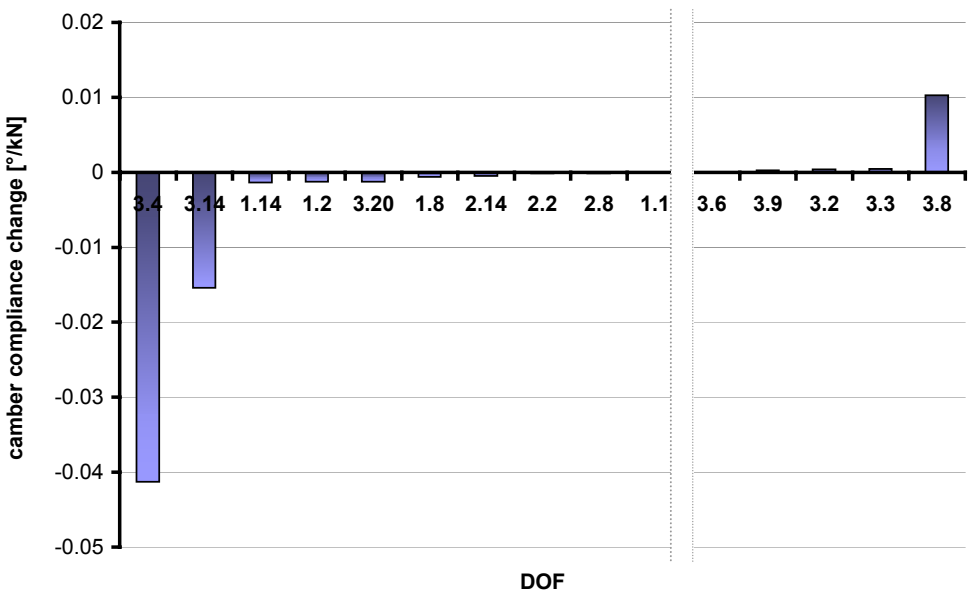


Figure 6.7 – Camber compliance change (sorted), by DOF

As a result of the sensitivity analysis, the DOF can be sorted out in two groups:

- those DOF that have virtually no influence in the behaviour of the model, regardless of how low the modification factor is,
- those that have great influence in the behaviour of the model.

The first group denotes those DOF that can be weakened without loosing any significant performance. In principle, with a lack of any better criteria these DOF can be multiplied by a factor such as 0.5. The second group denotes the DOF that have a major impact on the measured variable; therefore, careful assessment has to be carried over these DOF.

So far the process has been analysed with only a single constraint. However, in most cases during optimisation of a suspension assembly it is more common to have several constraints: camber and toe compliance, for example. In these cases, after running the sensitivity analysis six possibilities appear:

- the DOF increases both compliances (↑↑) → the DOF is labelled as important.
- the DOF is irrelevant for both compliances (--) → the DOF is labelled as unimportant.
- the DOF decreases both compliances (↓↓) → the DOF is labelled as important..
- the DOF is irrelevant for one of the compliances and increases the other one (-↑) → the DOF is labelled as important..
- the DOF is irrelevant for one of the compliances and decreases the other one (-↓) → the DOF is labelled as important.
- the DOF increases one of the compliances and decreases the other one (↑↓) → the DOF is labelled as important.

The above is summarised in **Table 6.2**:

effect	label
↑↑	important DOF
--	unimportant DOF
↓↓	important DOF
-↑	important DOF
-↓	important DOF
↑↓	important DOF

Table 6.2 – Labelling of the DOF according to their influence on the suspension performance

Thus, whenever a DOF has an effect on one or more measurements, whether positive or negative, it must be handled carefully and therefore it will be labelled as *important DOF*. Then, two important aspects are discussed in the next sections: why are the DOF on the right of side of the division in **Figure 6.6** considered to be important, and when can it be said that a DOF is actually important?

6.1 Important DOF: the sign

It seems obvious that the DOF at the left end of **Figure 6.7** should be considered important and not be automatically weakened, because even when that would imply a reduction in weight, it would also

mean that the suspension becomes more compliant. On the other hand, those DOF at the right end of **Figure 6.7** seem to be advantageous both for weight saving and for the stiffness of the suspension. They seem to be the first candidates to weaken instead of grouping them with the important DOF and include them in an optimisation process. For example, weakening DOF 3.8 might mean that DOF 2.14, 1.8, 3.20, 1.2 and 1.14 can be weakened. However, there are two reasons to proceed this way:

- since the objective function is influenced by both the weight (target) and the performance (constraints), the goal of the optimisation process can be better achieved by providing the optimisation algorithm with means both to reduce and to increase the different compliances, allowing it to tune the suspension to reach the highest level of fitness.
- in some situations, a minimum value of compliance is required for the suspension due to the driving characteristics intended for the vehicle. For example, in passenger cars a certain degree of understeer is preferable, so a given amount of toe compliance must be allowed to dynamically change the alignment of the wheels under cornering and induce such behaviour.

6.2 Limit DOF

When is a DOF relevant for a measurement? Or, conversely, when can it be said that a DOF is irrelevant for a measurement? In order to answer this, the same sensitivity analysis of **Figure 6.7** is shown once again in **Figure 6.8**, with some modifications.

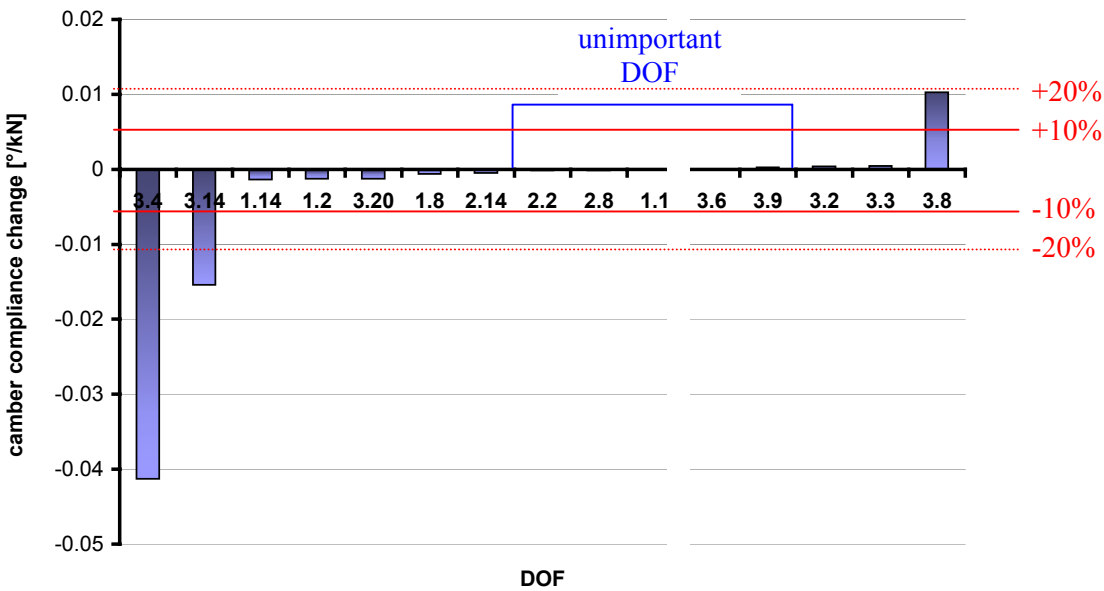


Figure 6.8 – Camber compliance change (sorted), by DOF

As it can be seen, the DOF number 4 of component 3 is the one that induces the biggest increment in camber compliance when it is weakened. This is not a coincidence since this DOF represents the rotation of the wheel in the *x* direction, as it illustrated in **Figure 6.9**. Clearly, there is a group of DOF that have no influence on the measurement, namely those from DOF 1,1 (component 1, lower control arm, DOF 1, translation in *x*) to DOF 3,6 (component 3, wheelcarrier, DOF 6, rotation in *z*), comprised by the blue segment in the **Figure 6.8**. To the left of this group are those DOF that increase the compliance when weakened, while to the right are those DOF that decrease the measurement.

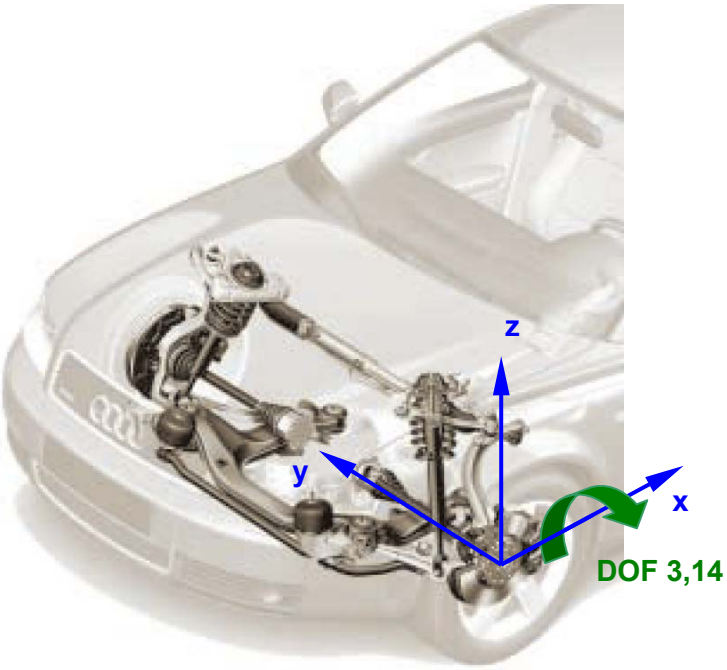


Figure 6.9 – Relationship of the DOF 3,14 with the camber angle

To set a limit that allows the DOF to be sorted out in *important* and *unimportant*, the DOF that has the strongest influence (in absolute value) is taken as a reference, in this case DOF 3,4. Next, any DOF with an influence on the camber compliance smaller than a given percentage of that reference value may be considered unimportant. This limit can be 10%, 20% or any other value up to 100%. Obviously, when the limit is lowered, more DOFs will be included among the important ones, since smaller changes in the camber compliances are enough to pass the lower limit.

For example, as shown in **Figure 6.8**, when the limit is set to 20% the only DOFs that are labelled as important are the 3,4 and the 3,14; when the limit is lowered to 10%, DOF 3,8 is also included. If the camber compliance is the only constraint imposed to the optimisation problem, the optimiser has to deal only with three parameters: the three modification factors applied to the pairs column/row that represent these DOFs in the stiffness matrix of the wheelcarrier.

The following discussion assumes that there is only one constraint for the optimisation process. The aim is to introduce the reader with a basic optimisation method. This allows a better understanding of the effect of the modification factors on the measurements. The presented algorithm has some parameters to modify (the modification factors of the important DOFs) and an upper limit for the camber compliance which can not be exceeded.

After the sensitivity analysis is carried out, it is determined that, with a limit of 10%, there are three important DOFs. Hence, the optimisation process begins multiplying every row/column corresponding to the unimportant DOFs by the default modification factor. The camber compliance is then measured and the result is expected to be very close to the reference value, since only the DOF labelled as unimportant are modified. Next, every important DOF, starting with the less relevant, is multiplied one after another by the default modification factor and the camber compliance is checked against the target. The two possibilities are:

- if the target is not exceeded after all DOFs were modified, the algorithm assumes that all the components involved in the optimisation can be weakened without failing the target.
- if the target is exceeded when applying the default modification factor to an important DOF, the algorithm understands that this DOF is the one that must be tuned in order to reach the target. This is called the *limit DOF*.

The aim of this method is that more DOFs can be weakened and therefore, in principle, more weight saved[6]. The unimportant DOFs are modified at once by the default modification factor to save computing time, taking advantage of the information collected by the sensitivity analysis. The algorithm assumes that the *limit DOF* lies within the important DOFs. The codification of this process is fully explained in **Figure 6.10**. For the development of the optimisation process, a process status window containing DOF, modification factor and camber compliance value can be issued and updated at every iteration to allow the user to detect when the process diverges and interrupt it. The limit that separates unimportant from important DOFs is set by the user at the beginning of the process.

Convergence depends, among other things, on the condition that the camber compliance after modifying the unimportant DOFs is lower than the target camber compliance. If it was higher, it means that too many DOFs where multiplied by the default modification factor in the first step of the optimisation process. In such a case the criterion to separate important from unimportant DOFs must be adjusted, that is, the percentage used as limit must be lowered to allow more DOFs to be included in the optimisation, and avoid multiplying them at once at the beginning of the process. Therefore, a check must be performed before starting the optimisation.

[6] There is no assumption regarding the importance of every DOF to the weight of the component. Each DOF is assumed to have the same influence on the final weight saving of the real component.

Once the limit DOF is found, a method is implemented to find the exact step to add to the default modification factor. The new modification factor is then applied to the limit DOF in order to get the camber compliance as close as possible to the target value.

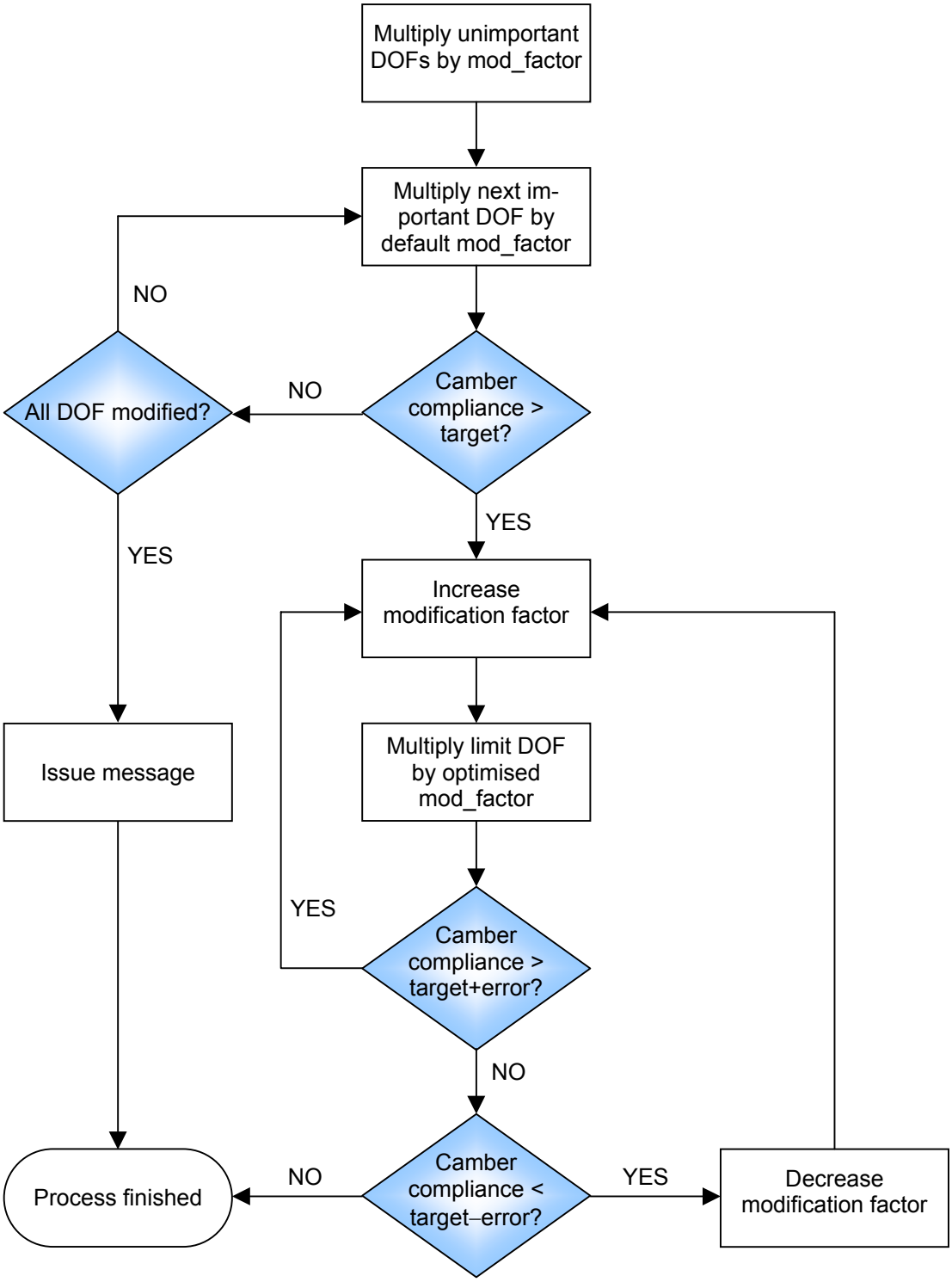


Figure 6.10 – Optimisation process by the bisection method

The criterion to choose the step size for the modification factor is given by the bisection method. A simple bisection procedure **[6.1]** for iteratively converging on a solution which is known to lie inside some interval [a, b] proceeds by evaluating the function (which in this case it consists of a simulation in ADAMS) at the midpoint of the original interval $x = \frac{a+b}{2}$, and testing to see in which of the sub-intervals $\left[a, \frac{a+b}{2}\right]$ or $\left[\frac{a+b}{2}, b\right]$ the solution lays. The procedure is then repeated with the new interval as often as needed to locate the solution with the desired accuracy. The advantage of this method over more sophisticated ones is that it is robust. The disadvantage is the convergence speed.

In order to determine a priori the number of iterations needed, n, let a_n and b_n be the endpoints at the nth iteration (with $a_1 = a$ and $b_1 = b$) and r_n be the nth approximate solution. The iterations required to obtain an error $|r_n - r|$ smaller than ε is found by noting that:

$$b_n - a_n = \frac{b-a}{2^{n-1}} \tag{6.1}$$

If r_n is defined by the middle point of a and b in the nth iteration, that is:

$$r_n \equiv \frac{1}{2}(a_n + b_n) \tag{6.2}$$

In order for the error to be smaller than ε,

$$|r_n - r| \leq \frac{1}{2}(b_n - a_n) = 2^{-n}(b-a) < \varepsilon \tag{6.3}$$

Taking the natural logarithm of both sides gives:

$$-n \cdot \ln 2 < \ln \varepsilon - \ln(b-a) \tag{6.4}$$

so

$$n > \frac{\ln(b-a) - \ln \varepsilon}{\ln 2} \tag{6.5}$$

The first part of the procedure in which the unimportant DOFs are modified requires only one simulation to check the resultant camber compliance. Every important DOF takes the same time to be modified and checked. In **Figure 6.11** it can be seen that the target camber compliance was exceeded only when modifying the last DOF in iteration 7. Therefore, the limit DOF is that with the highest effect on the camber compliance. Hence, three simulations are needed to go through every important DOF. The final part of the process is to optimise the limit DOF.

If an error $\varepsilon < 1e-5$ is allowed, the process to tune the limit DOF by the bisection method takes 9 simulations, from iteration 8 to 16. The progress of both modification factor and camber compliance

during the optimisation process can be seen in **Figure 6.12**. It shows the progress of both camber compliance in absolute value and the modification factor at each iteration.

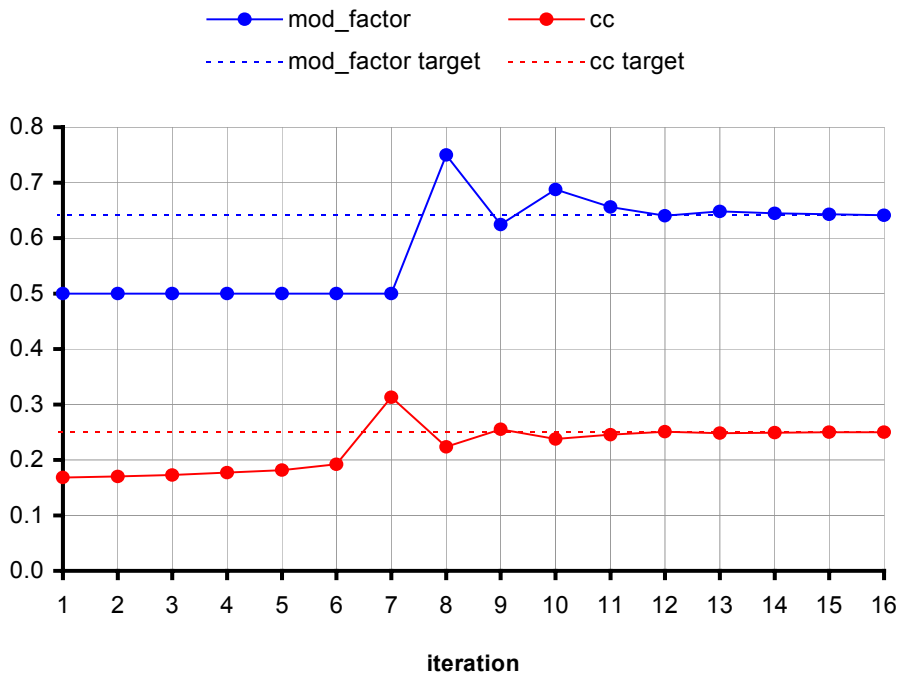


Figure 6.11 – Optimised modification factor and camber compliance [°/kN] by the bisection method

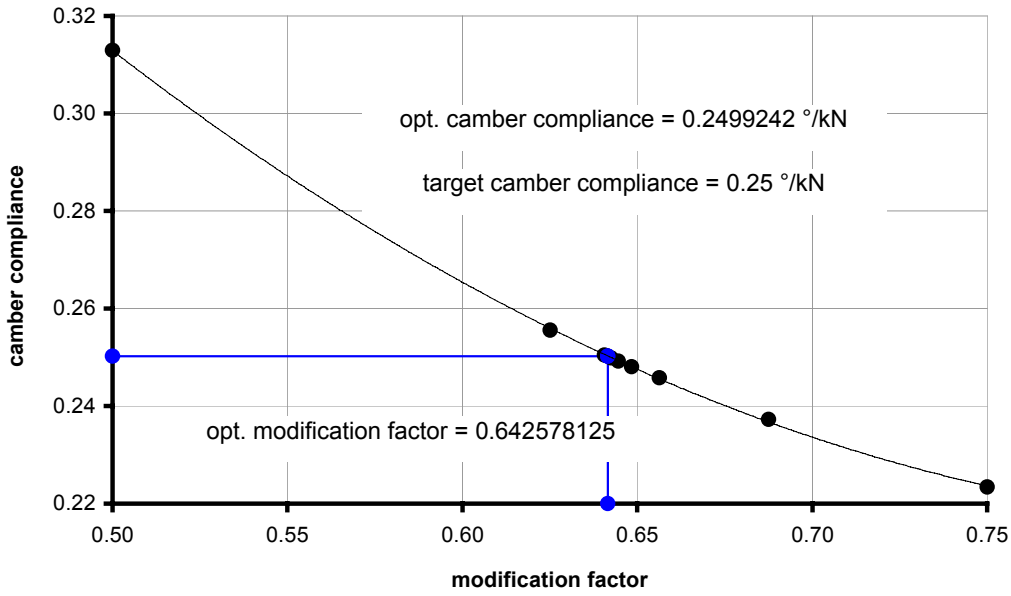


Figure 6.12 – Evolution of the camber compliance change versus the modification factor

In iteration 1 the unimportant DOFs are multiplied by the default modification factor: the camber compliance is below the target. Iterations 2 to 7 show the important DOFs being multiplied by the default modification factor: the camber compliance does not reach the target until the last iteration, in which the last (and most relevant) DOF is modified. From iteration 8 to 16 the modification factor is optimised by the bisection method.

A method with a higher rate of convergence such as Newton-Raphson would only have an effect in the last part of the process (the right side of the **Figure 6.11**). Assuming that such a method needs only 3 iterations instead of 10 (reasonable expected improvement when comparing both methods), it would mean a reduction of about 40% in the computation time.

7 KNOWLEDGE DATABASE

Arguably, the main disadvantages of the genetic algorithms compared with other methods are their slow convergence and the high number of function consultations in order to assess and assign a fitness value for every individual in the population at each generation.

In the particular case of a suspension assembly in which the constraints of the optimisation process are the compliances of the wheel and a genetic algorithm is used, every individual in the population at each generation must be assessed by measuring the compliance with the corresponding simulation in ADAMS/Cars which, at the present time, takes about 20 seconds to be completed[7]. If a population of just 100 individuals is used, a typical run of a genetic algorithm takes about 300 generations to deliver a satisfactory solution. This means an estimated of 166 hours and 40 minutes, or almost 7 days. That is, if only one simulation per individual is needed, which most certainly is not the case. Clearly, a way to implement the algorithm in order to speed up the process is needed.

7.1 Assumption of independence of the optimisation parameters

"Epistasis" is the term used to describe a situation in which a gene or group of genes influences the expression of another gene. It can happen that the presence of one gene might suppress the effect of another; alternatively, one might have a gene whose effect will not be evident unless a group of other, prerequisite genes are present as well **[7.1]**. For example, in a combination lock none of the numbers in the sequence counts until all the others have been taken care of, in order.

In mathematical language, if two factors applied to two DOF define a search space, the directional derivative of the measured compliance (as a function of those two factors) in the direction of one DOF is not dependent on the other DOF. For example, if the camber compliance is referred to as $CC_{(f_1,f_2)}$, the previous assumption can be written using the partial derivatives as:

$$\left.\frac{\partial CC_{(f_1,f_2)}}{\partial f_1}\right|_{f_1=f_{1_0}} = k \quad \forall f_2 / f_{2_{\min}} \le f_2 \le f_{2_{\max}} \tag{7.1}$$

or

[7] To deliver this performance the computer used for this research is basically a 3.2 GHz, Pentium 4 processor, with 2 GB of RAM.

$$\frac{\partial\left(\frac{\partial CC_{(f_1,f_2)}}{\partial f_1}\right)}{\partial f_2}=0 \quad \forall f_1 / f_{1_{\min}} \le f_1 \le f_{1_{\max}} \quad \wedge \quad \forall f_2 / f_{2_{\min}} \le f_2 \le f_{2_{\max}} \tag{7.2}$$

where

$f_{i_{\min}} = 0.5$ (user defined)

$f_{i_{\max}} = 1$

The minimum and maximum values for f_1 and f_2 define the domain of the problem. Likewise,

$$\left.\frac{\partial CC_{(f_1,f_2)}}{\partial f_2}\right|_{f_2=f_{2_0}} = k \quad \forall f_1 / f_{1_{\min}} \le f_1 \le f_{1_{\max}} \tag{7.3}$$

or

$$\frac{\partial\left(CC_{f_2}(f_1,f_2)\right)}{\partial f_1}=0 \quad \forall f_1 / f_{1_{\min}} \le f_1 \le f_{1_{\max}} \quad \wedge \quad \forall f_2 / f_{2_{\min}} \le f_2 \le f_{2_{\max}} \tag{7.4}$$

In case that there are *n* modification factors to optimise, the rule can be expressed by:

$$\frac{\partial\left(\frac{\partial CC_{(f_1,f_2,\ldots,f_n)}}{\partial f_i}\right)}{\partial f_j}=0 \quad \forall f_{1\ldots n} / f_{1\ldots n_{\min}} \le f_{1\ldots n} \le f_{1\ldots n_{\max}} \quad \wedge \quad \forall i \ne j \tag{7.5}$$

In practical terms this assumption means that the directional derivative calculated with respect to f_1 does not change after modifying f_2 and vice versa. Although this is not exactly the case, it does get close since the degree of epistasis is actually very low. Therefore, with the help of the knowledge database it is possible to predict, by a simple algebraic addition, the variation in camber compliance (or toe compliance, or any other compliance used as constraint) when applying simultaneously several changes to a set of modification factors.

If the parameters of the optimisation (the DOFs) are independent or have very slight effect on each other, they can be treated as if the others did not exist, that is, no epistasis present in the genotype (genes) that affects the phenotype (behaviour) of the individual. Based on this concept, running simulations in ADAMS with the components based on the stiffness matrices in which one factor is modified at a time, a table can be built for each compliance in which each of the DOF (the columns) is perturbed by a modification factor[8] (the rows), and the variation in the measurement is determined by a simulation run and stored. For each constraint, a similar table is built and all the tables are stored in a matrix called the *knowledge database*.

[8] As discussed previously, the modification factors applied to the stiffness matrices are allowed to take values from a lower limit set by the user (for example 0.5) up to 1.

Later on the genetic algorithm refers to this table every time that it has to assess an individual and assigns a fitness value to it, instead of a computationally expensive consultation of the constraint value by running the actual simulation. Looking up in the table takes about 1/2000 of a second, so compared with an ADAMS simulation it is about 40.000 times faster, which means that the optimisation process mentioned before takes now about 15 seconds.

7.2 Interpolation of the modification factors

Figure 7.1 to **7.3** show the impact in the suspension compliances, of different modification factors (ki) applied to the 11 important DOFs of a suspension assembly. These charts are based on measurements from actual simulations in ADAMS performed by modifying one DOF at a time, through the range of values allowed for the modification factors, in 16 steps.

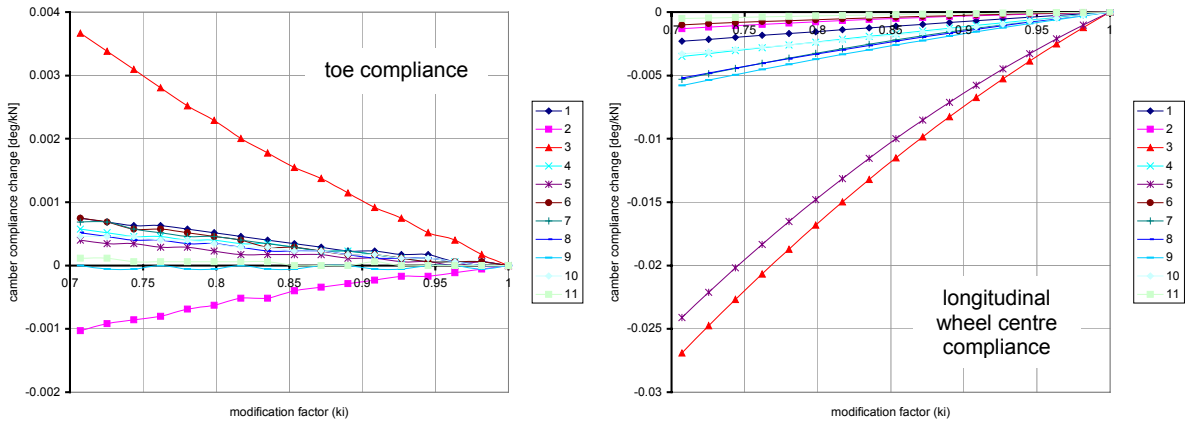


Figure 7.1 – Load case: acceleration force

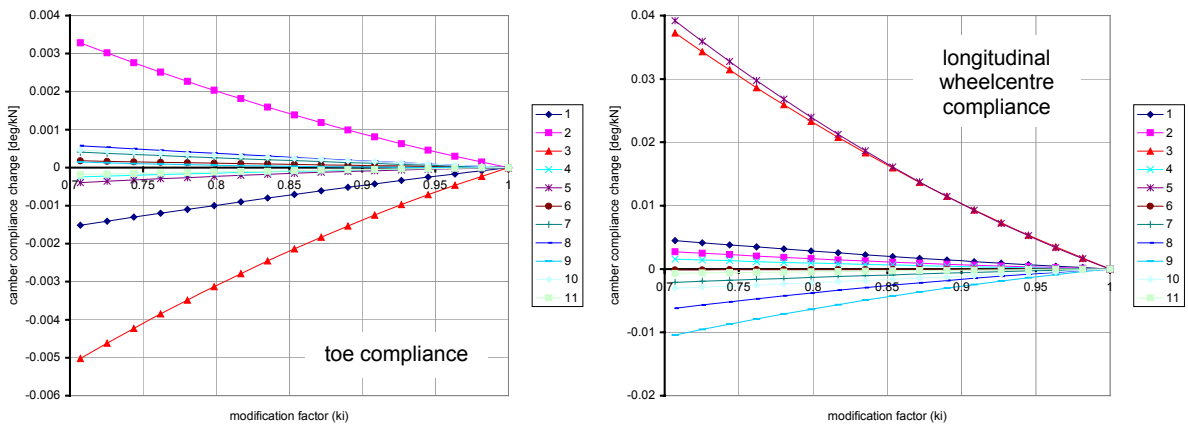


Figure 7.2 – Load case: braking force

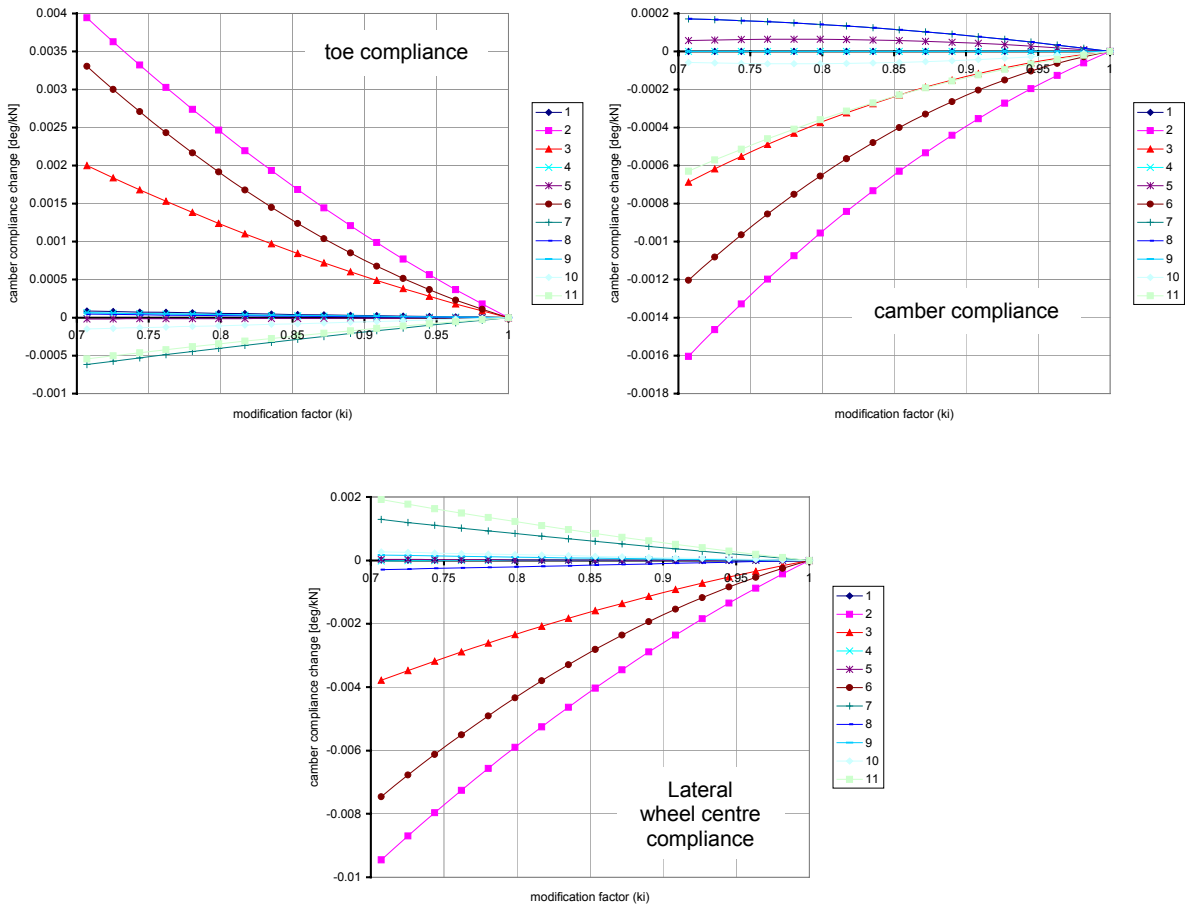


Figure 7.3 – Load case: lateral force

As expected, all the charts present zero variation in the measurements when a factor 1 is applied to any DOF, and the closer the optimisation factor is to the lower limit, the more important is the variation of the compliances. From **Figure 7.1** it can be seen that DOF 3 is the most important for both measurements; the difference lays in that, while the toe compliance reference value is positive, the camber compliance reference value is negative, so both compliances are increased in magnitude when the DOF is weakened. During a braking force the behaviour is exactly the opposite. In any case, it can be confirmed that the importance of the DOF does not change due to variations of the modification factor since, regardless its value, the DOF always has a high effect on the measurement.

It can also be noticed that the curves apparently follow a parabolic shape. This fact leads to the idea that these plots can actually be approximated by a second order polynomial. In this way, for every curve, instead of 17 simulations only 3 points should be sampled and the rest can be approximated by a function that approximates the sample points, corresponding to a reduction in time needed for the process of building the knowledge database.

The next charts (**Figures 7.4** to **7.6**) show the result of such approach to build the knowledge database. As it can be seen most cases the curves are identical. An exception is the first chart, which corresponds to the change in toe compliance under acceleration. The reason is that the toe angle and the toe compliance are rather sensitive parameters with very small variations. ADAMS is able to measure such variations in the simulation at the cost of capturing also mathematical noise, mainly due to numerical round off errors.

Focusing for example in DOF 3, it can be seen in **Figure 7.7** its influence on the longitudinal wheel centre compliance under acceleration, both for the actual measurement from a simulation in ADAMS and for the approximated values given by a second order polynomial based on the points (x_1,y_1), (x_2,y_2) and (x_3,y_3).

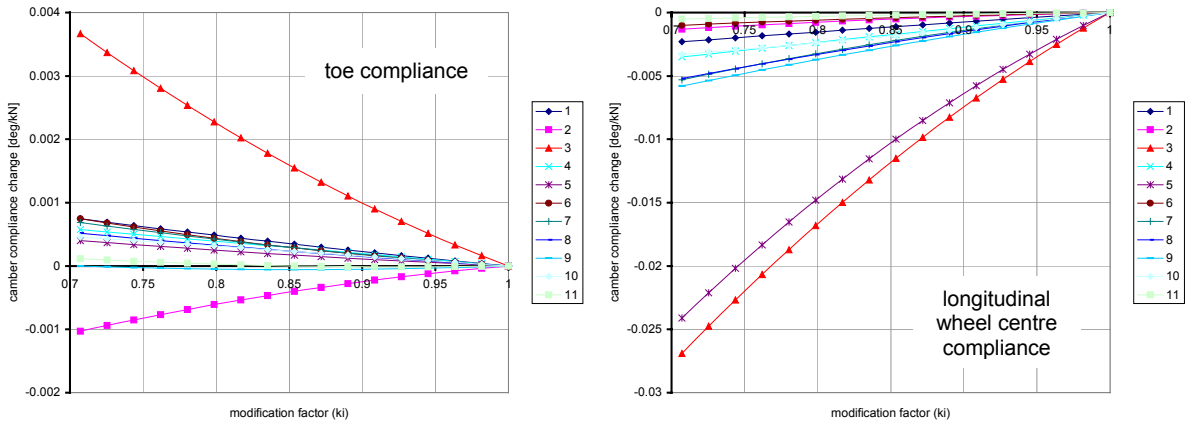


Figure 7.4 – Load case: acceleration force

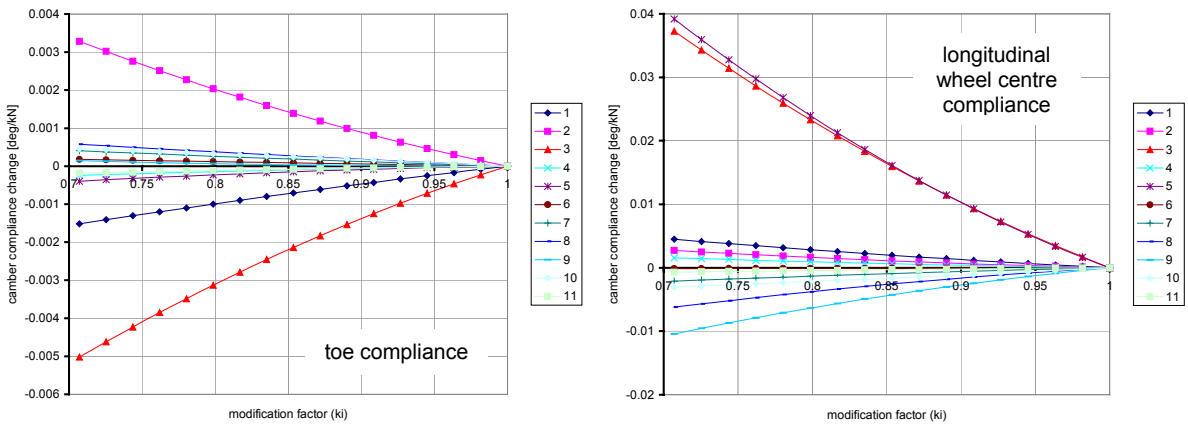


Figure 7.5 – Load case: braking force

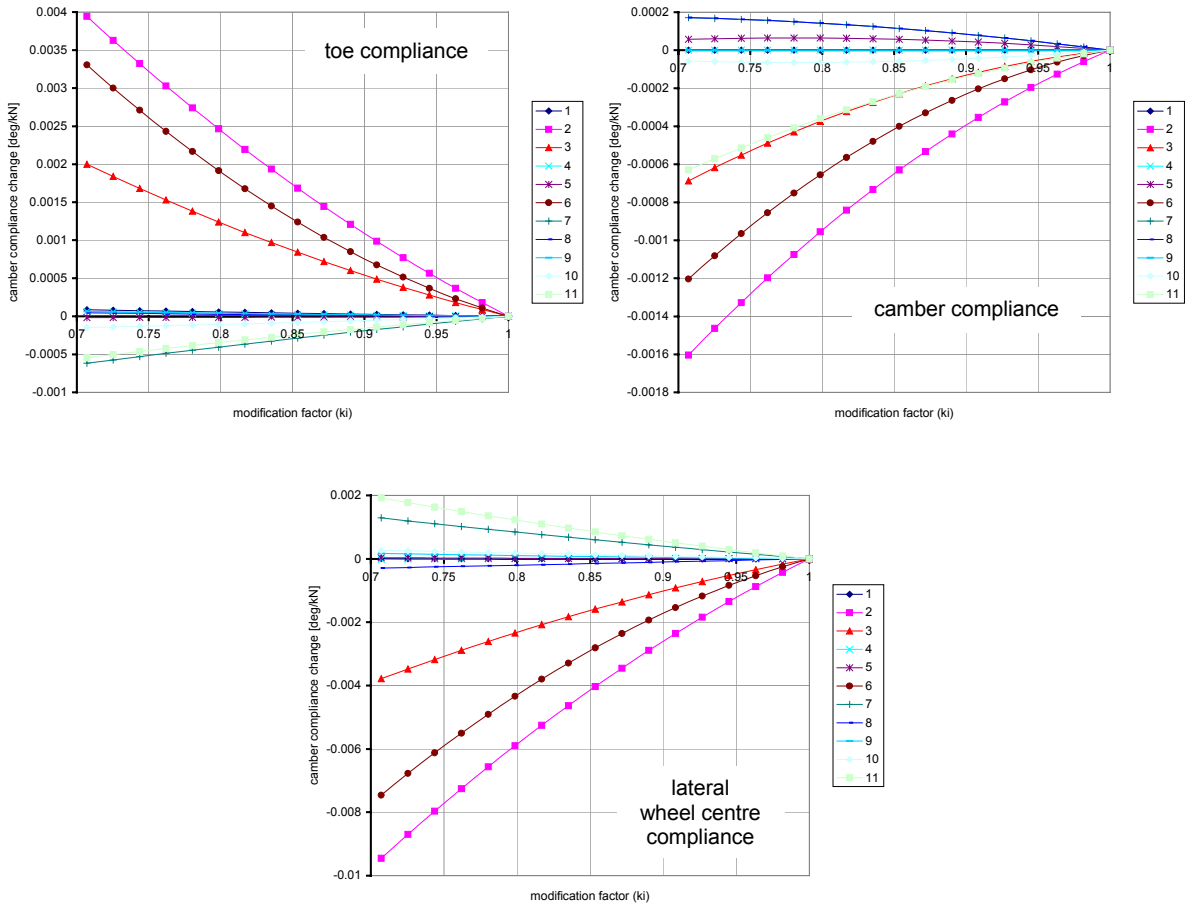


Figure 7.6 – Load case: lateral force

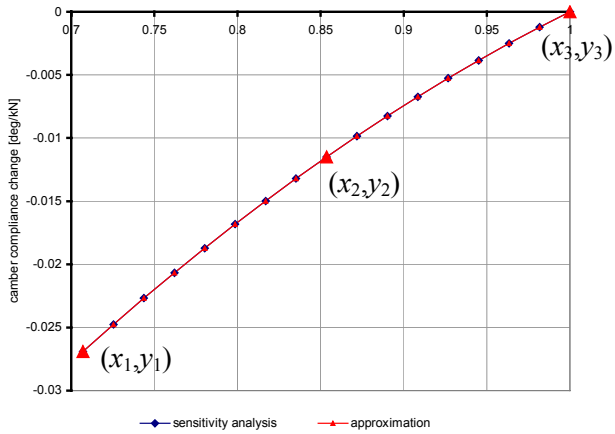


Figure 7.7 – Measured camber compliance and approximated curve

Figure 7.8 shows the relative error made by the approximating curve based on the points (x_1,y_1), (x_2,y_2) and (x_3,y_3), which is insignificant.

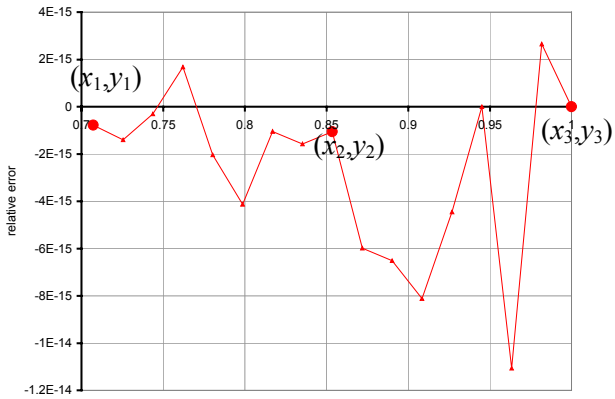


Figure 7.8 – Error by replacing the measurements with the approximated values

The right chart of **Figure 7.7** shows that the relative error of the approximated curve in two of the three points used to build the second order polynomial is still not 0. This is merely due to numerical error and, looking at the scale of the vertical axis (10e-15), it is insignificant.

The equation used to build the approximating polynomial f is $f = a \cdot x^2 + b \cdot x + c$, where:

$$a = \frac{\dfrac{y_1 - y_3}{x_1 - x_3} - \dfrac{y_2 - y_3}{x_2 - x_3}}{x_1 - x_2} \tag{7.6}$$

$$b = \frac{y_2 - y_3}{x_2 - x_3} - a \cdot (x_2 + x_3) \tag{7.7}$$

$$c = y_3 - a \cdot x_3^2 - b \cdot x_3 \tag{7.8}$$

and (x_1,y_1), (x_2,y_2) and (x_3,y_3) are the three approximation points, being

$x_1 = \dfrac{\sqrt{2}}{2}$ ← lower limit set by the user

$x_3 = 1$ ← maximum possible value for the modification factors

$x_2 = \dfrac{x_1 + x_3}{2}$ ← middle point

and y_1, y_2 and y_3 the measurement (in this case, the longitudinal wheel centre compliance under acceleration) at x_1, x_2 and x_3, respectively.

A further and important advantage of building an approximating function based on a few sampling points is the possibility of increasing the resolution of the approximated curve, for instance by dou

bling the amount of approximated points (**Figure 7.9**) in order to allow the genetic algorithm a finer tuning during the optimisation process.

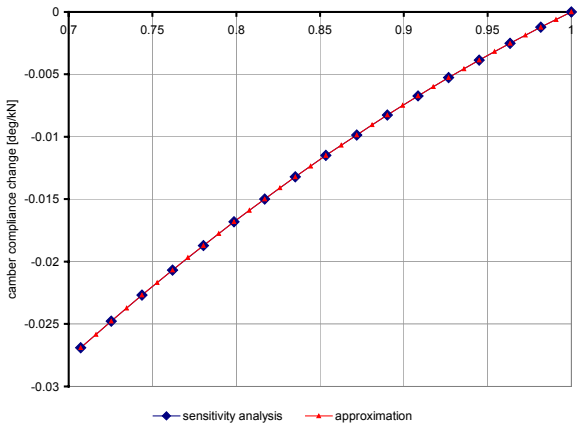


Figure 7.9 – Approximation using more points

8 TAKING MEASUREMENTS WITH ADAMS/CARS SIMULATIONS

In order to run any optimisation process over a model that represents a suspension assembly, the initial performance must be measurable as well as any change in this performance. The first measurements of the compliances serve as reference values against which future measurements, obtained from the modified model, are compared in order to assess the proposed solution.

When the optimisation process by means of a genetic optimiser is running, the performance of each individual of the population must constantly be monitored. Part of the performance is given by the ability of the individual to fulfil the constraints imposed to the optimisation problem, or at least deliver a performance as close as possible to the target compliances. Regardless of implementation improvements that speed up the process of assessing an individual during the optimisation (as it is the building of a knowledge database), the compliances must be measured for the original suspension assembly, as well as those resulting from perturbing its DOF. For this purpose, a multi-body static and dynamic analysis program is used, called MSC.ADAMS/Cars, from now on referred to simply as ADAMS. Each of the measurements taken from the simulations in ADAMS has a particular meaning, and it is important to fully understand the differences between them and what they add to the suspension, both from the performance standpoint and as constraints.

For the following, the camber compliance is the measurement considered. As mentioned in Section **3.1**, when a simulation is performed in ADAMS/Cars the results are posted in a request file (.req). ADAMS measures the angle of the wheel before and after the force is applied, and the following equation is used:

$$CC = \frac{\Delta\gamma}{\Delta F} = \frac{\gamma_f - \gamma_i}{F_f - F_i} \tag{8.1}$$

where

CC = camber compliance

γ_i = camber angle at time i (initial)

γ_f = camber angle at time f (final)

F_i = force at time i (initial)

F_f = force at time f (final)

The units of the camber compliance are °/kN. Usually, F_i and F_f are in the order of 1 kN ± 50 N, so that (8.1) becomes:

$$CC = \frac{\gamma_f - \gamma_i}{100\ N} \tag{8.2}$$

Therefore, γ_i and γ_f are read from the request file at the end of the simulation. It is important to notice that the compliances of a suspension assembly are fairly constant up to values of approximately 10 kN of force applied, be it at the contact patch (depending on the suspension layout) or at the wheel centre. This is due to the linear elastic behaviour of the components responsible for most of the compliance, the bushings, until they start reaching their limit of deformation. This means that the components under optimisation, the control arms and other metal parts, work well below their elastic limits and, therefore, also linearly. Consequently there is no need of measuring and calculating the compliances at other force levels.

Three different static simulations are performed in ADAMS, and the respective measurements are:

- lateral force:
  - toe angle,
  - camber angle,
  - lateral wheel centre displacement.
- acceleration force:
  - toe angle,
  - longitudinal wheel centre displacement.
- braking force:
  - toe angle,
  - longitudinal wheel centre displacement.

Although the importance of each measurement was already discussed in Section **1.4**, a brief reminder of each one of them is provided, arranged by simulation.

8.1 Lateral force

When a vehicle is cornering the wheels scrub and develop a slip angle (**Figure 8.1**) that allows the tires to build up grip. The grip, expressed as a friction coefficient, can be translated as a lateral acceleration exerted on the vehicle when it turns. Multiplied by the vertical load on the wheel, gives the lateral force that the tire develops.

In order to simulate this situation in ADAMS and to study its effects on the suspension compliances, a lateral force of 950 N and 1050 N is applied at the contact patch (**Figure 8.2**). Different measurements (toe and camber angle, and lateral wheel centre displacement) are taken at both force levels and the compliances calculated, according to (8.2).

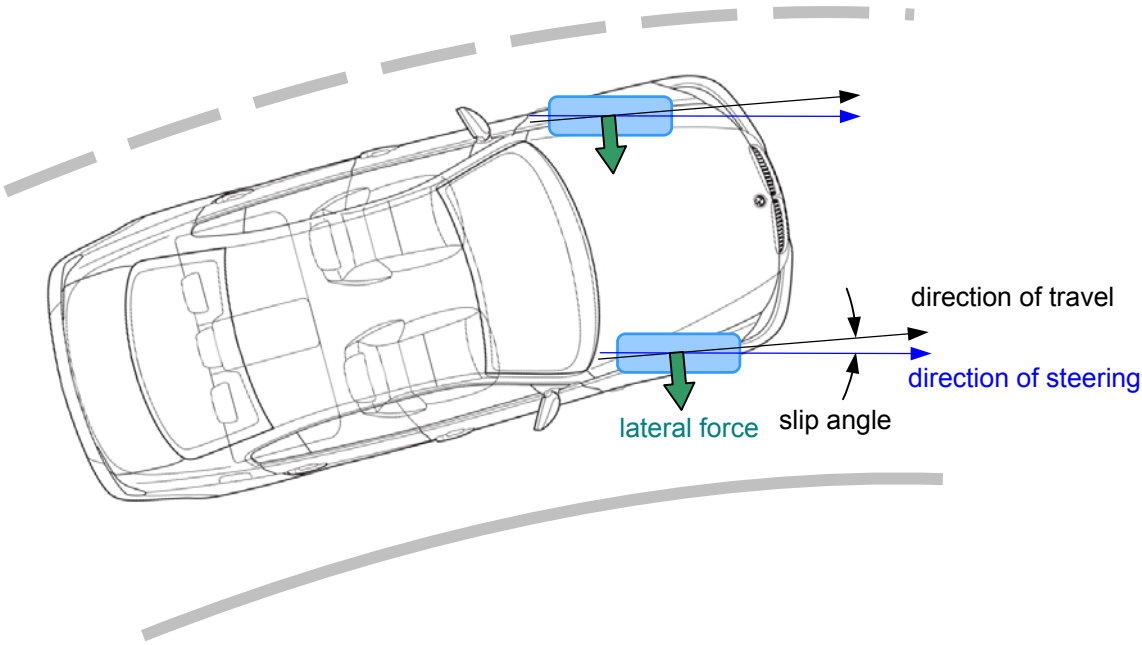


Figure 8.1 – Slip angle

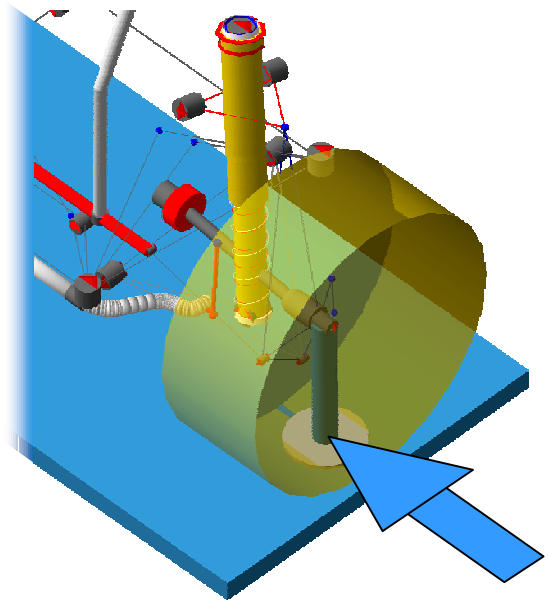


Figure 8.2 – Lateral force applied by ADAMS

During cornering, the lateral forces created at the tire contact patch induce changes in the wheel settings for two reasons: compliances of the suspension assembly and changes in the suspension geometry due to the travel of the wheels, being compressed (outer wheel) or extended (inner wheel) as a result of load transfer produced by the cornering forces. When the lateral load is applied by ADAMS during the simulation, the wheel is locked in its design position, so no wheel travel is allowed in order to measure only the compliances.

8.2 Acceleration force

Every time a vehicle accelerates the power of the engine is transmitted along the powertrain and delivered to the wheels, which through the contact patch exert the impulsion force against the road. In the case of a rear-wheel drive vehicle, the path is: pistons, crankshaft, clutch, gearbox, propeller shaft, differential, drive shafts, rear wheels. In such a case there is, of course, no force applied to the front wheels. In a front-wheel drive vehicle, the path is: pistons, crankshaft, clutch, gearbox, differential, drive shafts, front wheels. Therefore, there is a torque coming out of the vehicle and applied to the wheel, which, in turn, is divided by the radius of the wheel, converted into impulsion force.

In order to correctly represent and simulate this situation in ADAMS the acceleration force is not applied at the contact patch but at the wheel centre, as shown in **Figure 8.3**, since the front suspension assembly experiences no torque when the vehicle is accelerating, and the wheel rotates with respect to the wheel carrier with no interference from it in its movement.

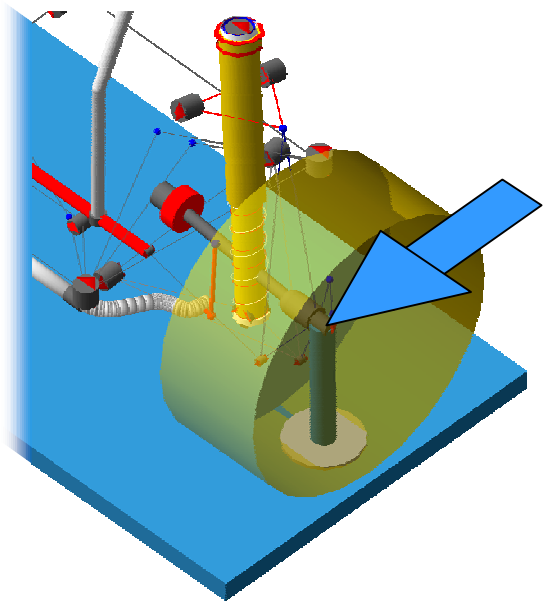


Figure 8.3 – Acceleration force applied by ADAMS

When the acceleration force is applied the longitudinal force induces changes in the wheel settings due to the compliances of the suspension assembly. Changes in the suspension geometry due to wheel travel are avoided by fixing the wheel centre in the vertical direction of displacement. The procedure is analogous to the lateral load case previously discussed.

8.3 Braking force

Most modern passenger vehicles braking systems rely on the so called disc brakes like the one shown in **Figure 8.4**, so this type is considered for the purpose of the following discussion.

When a vehicle brakes the brake callipers attached to the wheelcarrier press the brake pads against the discs, which are attached to the wheels. Large amounts of heat are generated during this process at the brake pads, which are dissipated by the disc since most of it is exposed to the open air. This heat represents the kinetic energy stored in the moving vehicle. The speed of the energy conversion controls the rate of retardation of a vehicle, i.e., its rate of deceleration. Obviously, such process induces not only heat but also great mechanical loads into the suspension components.

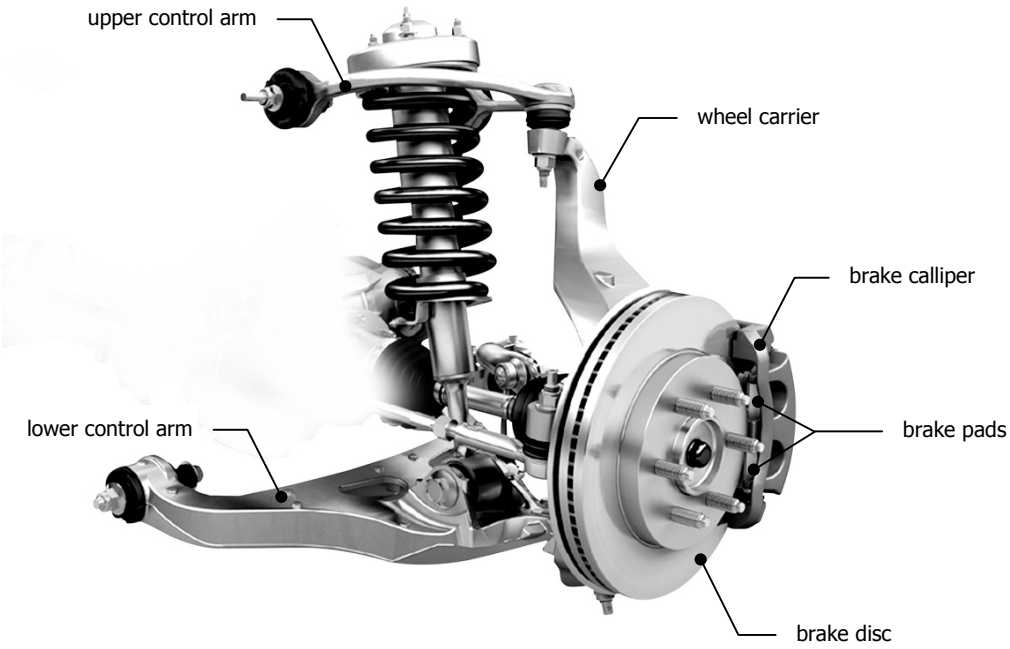


Figure 8.4 – Braking system

The frictional force applied at the brake pads times the distance of the application point on the disc induces a torque on the wheel centre, which divided by the wheel radius gives the braking force applied at the contact patch, as shown in **Figure 8.5**. The result of the combination of this system of

forces is a tendency of the wheel carrier (the mounting piece of the brake calliper) in the direction of rotation of the wheel. In addition, the wheel carrier must withstand bending forces, compression forces, or both (depending on the location of the mounting points of the calliper to the wheel carrier). This forces, together with the braking forces, are then transmitted to the control arms and finally to the vehicle body, braking it.

In order to correctly represent and simulate this situation in ADAMS the braking force is applied at the contact patch, since the front suspension assembly experiences a torque when the vehicle is braking. Changes in the suspension geometry due to wheel travel are also here avoided by fixing the wheel centre in the vertical direction of displacement. The procedure is analogous to the acceleration load case, as discussed previously.

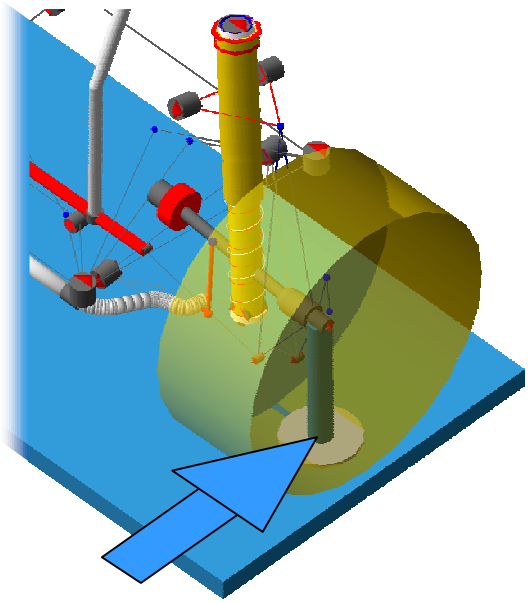


Figure 8.5 – Braking force applied by ADAMS

9 THE STATIC ANALYSIS

The static analysis is the last step of the optimisation process, according to the flux diagram shown in **Figure 3.1**. The aim of this stage is to issue the set of targets for the components of the suspension whose stiffness matrix has been optimised. This set of targets is then passed to the CAD department to proceed with their redesign.

In order to run the static analysis, the stiffness matrices are used by a software package, in this thesis it is MSC.Nastran 2004. The analysis consists in applying different border conditions to the components, that is sets of loads (forces or moments) and constraints (translational or rotational) to the hardpoints, and assesses their flexibility by measuring the deflection of those hardpoints in the desired directions. In order to make a clean analysis, one force is applied at a time, always 1 kN, and the deflection of the hardpoint to which the force is applied is measured in the same direction of the force. **Figure 9.1** shows an example where a simple beam is constrained from translation and rotation at the left end, while at the right end a vertical force is applied. The displacement of the right end in the vertical direction is measured.

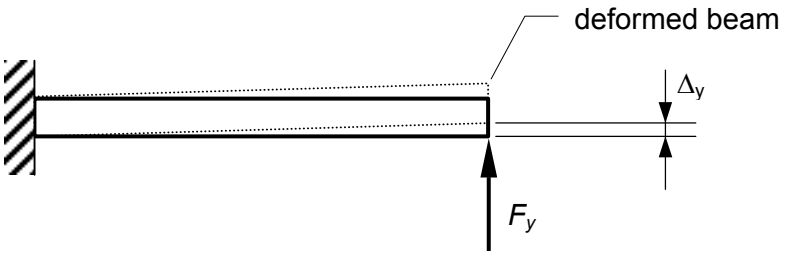


Figure 9.1 – Deformed beam under load

In the case of the load cases selected to apply to the suspension components, they must be set so to represent as well as possible the real solicitations exerted to the components during their service life. On one hand, a set of forces that does not occur during the real operating conditions of the suspension would do nothing but increase the number of constraints to the optimisation process, leaving out solutions that could actually mean an improvement in the final design. On the other hand, all possible load cases that do happen during the real use of the components should be foreseen by the user of the optimisation tool and included in the list of load cases to apply.

As an example of the first, it makes no sense to measure the torsional stiffness of a steering link (also referred to as tierod) such as the one shown in **Figure 9.2**, given that the component is attached at one end by a balljoint, which in principle does not transmit rotational loads in any direction.

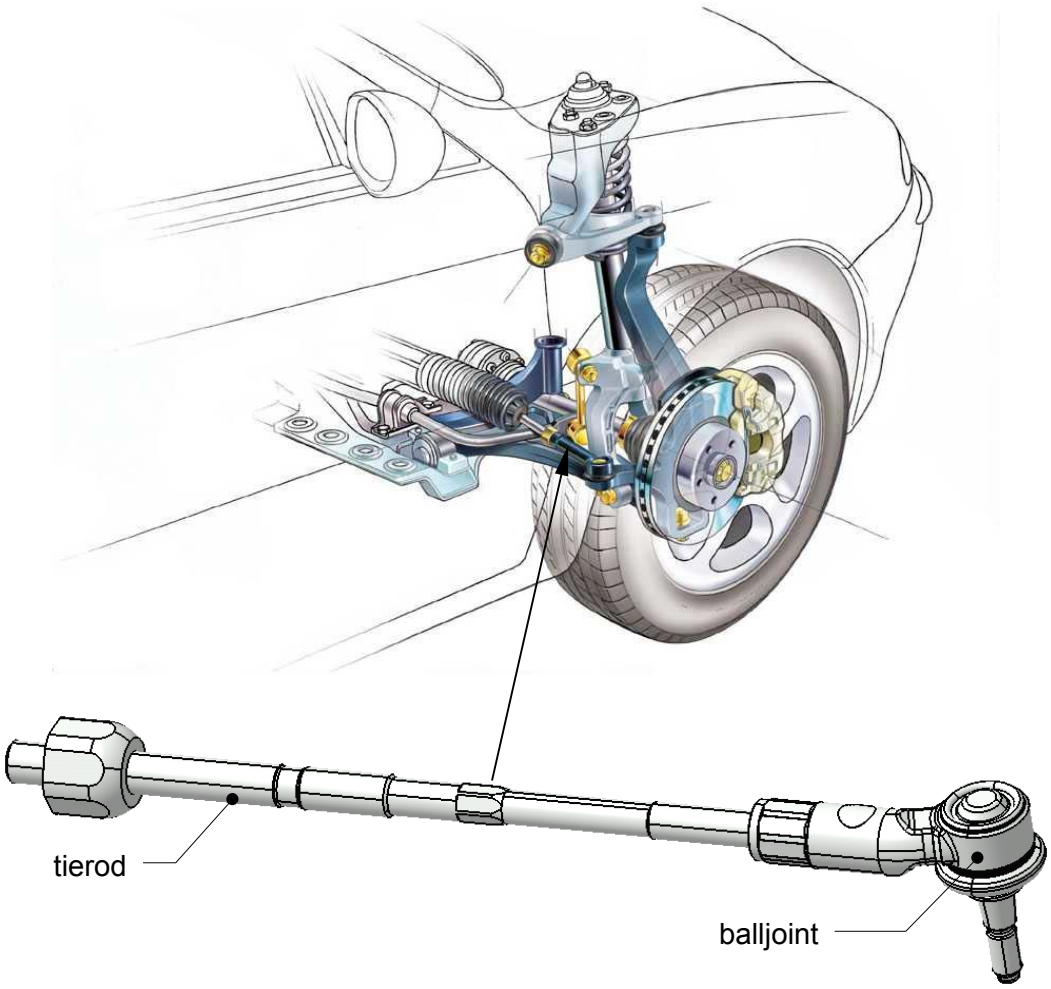


Figure 9.2 – Tierod in a suspension assembly

As an example of the second, it is imperative that a compression load on the tierod is included in the set of load cases to apply to this component, since this is its actual function in the suspension assembly, transmitting the steering force from the steering rack to the wheelcarrier.

9.1 Lower control arm

Continuing with the three components assembly presented in **Figure 6.4**, the set of load cases for the lower control arm is developed and presented. As depicted in **Figure 9.3**, the lower control arm is statically analysed under the following conditions: the two inner hardpoints connecting the arm to the

chassis are constrained from displacing in *y* (lateral) and *z* (vertical) directions. The front inner hardpoint is also constrained in the *x* (longitudinal) direction. The hardpoint connecting the arm with the strut is constrained in *z* (vertical) direction, since this is the one that supports the vertical loads to the suspension spring-damper unit.

For this component three load cases are used, where a force of 1000 N is applied and the displacement measured, successively on the three directions of movement at the outer hardpoint. The information is summarised in **Table 9.1**.

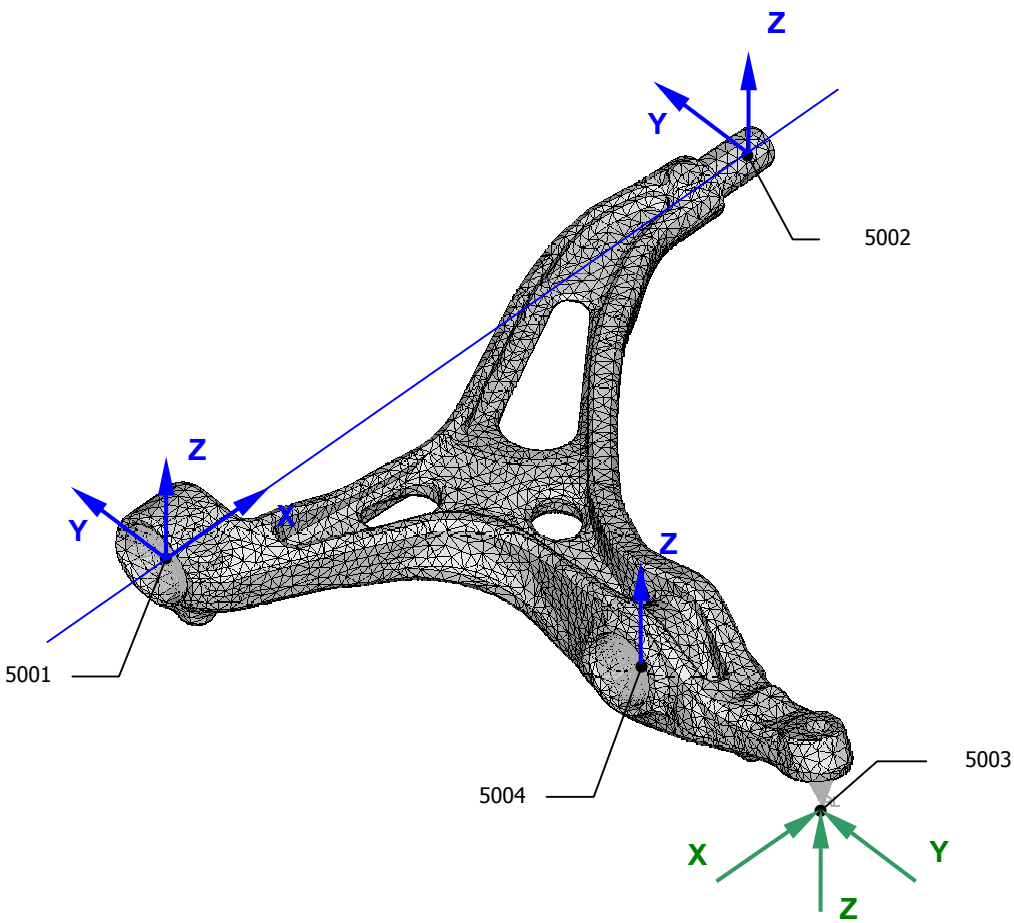


Figure 9.3 – Load cases for static analysis of lower control arm

	force	constraints			mea.
	5003	**5001**	**5002**	**5004**	**5003**
lc_1	1	123	23	3	1
lc_2	2	123	23	3	2
lc_3	3	123	23	3	3

Table 9.1 – Summary of load cases for static analysis of lower control arm

9.2 Upper control arm

In **Figure 9.4** it can be seen how the upper control arm is statically analysed: the two hardpoints connecting with the control arms are constrained from displacing in *y* (lateral) and *z* (vertical) directions. The front inner hardpoint is also constrained in the *x* (longitudinal) direction to avoid the component to move as a free body when the force is applied.

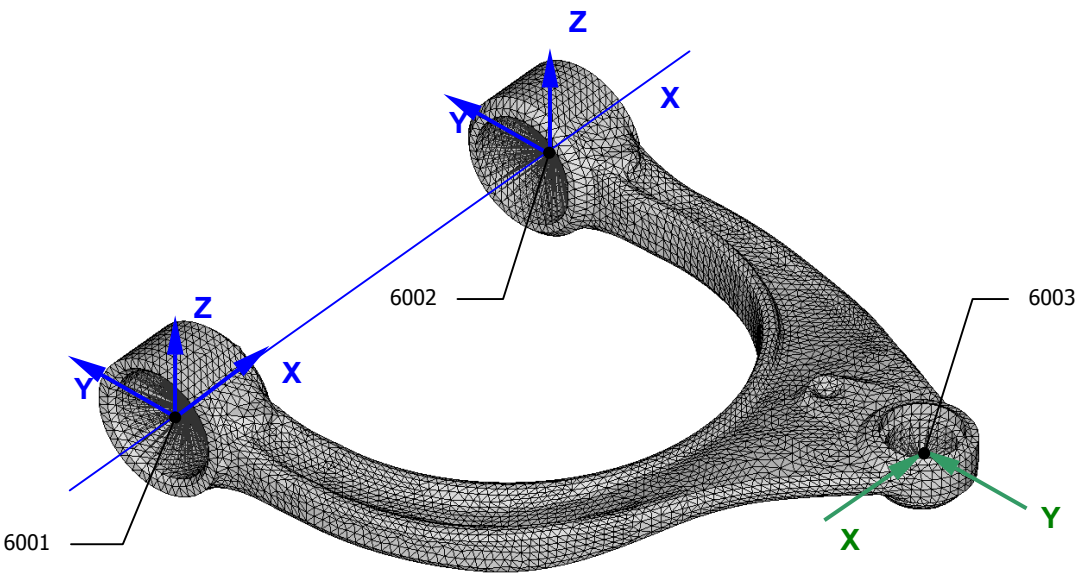


Figure 9.4 – Load cases for static analysis of upper control arm

In each of the two load cases a force of 1000 N is applied on the *x* (longitudinal) and *y* (lateral) directions of movement at the outer hardpoint, and the displacement in the respective direction measured. The information is summarised in **Table 9.2**.

	force	**constraints**		**mea.**
	6003	**6001**	**6002**	**6003**
lc_1	1	123456	23456	1
lc_2	2	123456	23456	2

Table 9.2 – Load cases for static analysis of upper control arm

For the lower control arm no force is applied in the vertical direction, since this component within the assembly only controls the position of the upper hardpoint of the wheelcarrier through the arc of movement of the wheel. The suspension forces are dealt with only by the lower control arm.

9.3 Wheelcarrier

The wheelcarrier is statically analysed under the following situation (**Figure 9.5**): the two inner hardpoints, connecting to the lower and upper control arms, are constrained from displacing in *x* (longitudinal) and *y* (lateral) directions. The lower hardpoint is also constrained in the *z* (vertical) direction, since this one transmits the suspension forces to the wheel. The hardpoint connecting the knuckle with the tierod is constrained in *y* (lateral) direction. Once again, no force or vertical constraint is applied at the upper hardpoint (1003) since only the lower one copes with suspension loads.

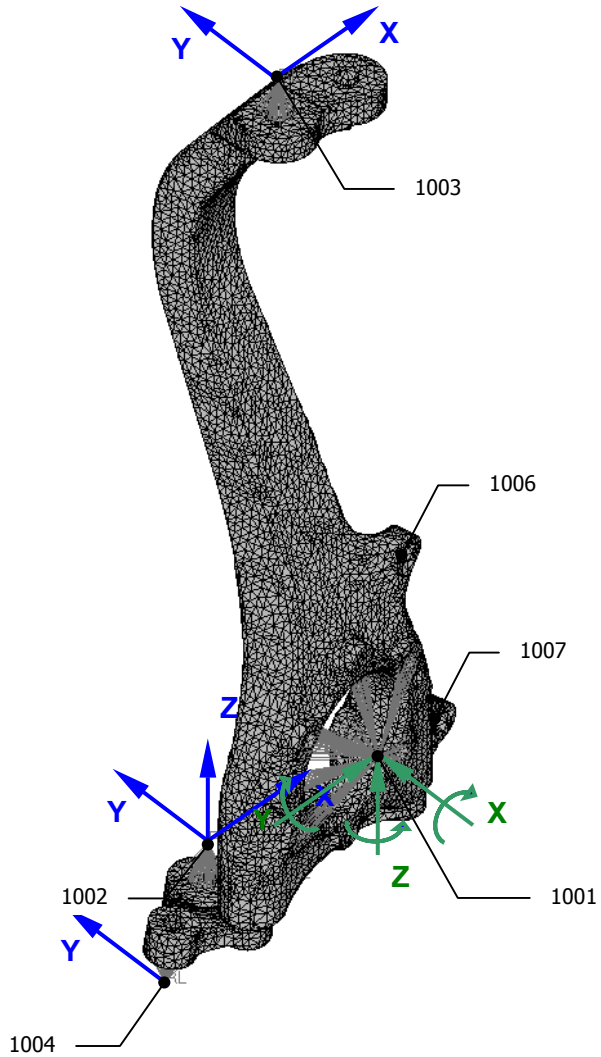


Figure 9.5 – Load cases for static analysis of the wheelcarrier

In each of the six load cases, a force/moment of 1000 N/Nm is applied on the six directions of movement at the wheel centre, and the displacement (rotation) measured. The information of the load cases applied to the knuckle is summarised in **Table 9.3**.

	force	constraints			mea.
	1001	**1002**	**1003**	**1004**	**1001**
lc_1	1	123	12	2	1
lc_2	2	123	12	2	2
lc_3	3	123	12	2	3
lc_4	4	123	12	2	4
lc_5	5	123	12	2	5
lc_6	6	123	12	2	6

Table 9.3 – Load cases for static analysis of the knuckle

9.4 Relationship between measurements and load cases

Due to the complex interaction of the parts in the suspension assembly, the relationship between the load cases and each of the measurements is not straightforward. However, some of them are clearly related, as it is discussed in the next paragraphs.

9.4.1 Lower control arm

Basically, the load cases exerted on the lower control arm aim to measure the elasticity of the outer hardpoint when forces in the three translational DOFs are applied. By applying the force in the *x* direction (**Figure 9.6.a**) pushes the wheel rearwards, consequently it is related with the longitudinal wheel centre compliance. When the force is applied in the *y* direction (**Figure 9.6.b**) the wheel is pushed inwards, relating the load case with the lateral wheel centre compliance. Since the force is applied with an offset below the wheel centre, the load case is related with the camber compliance.

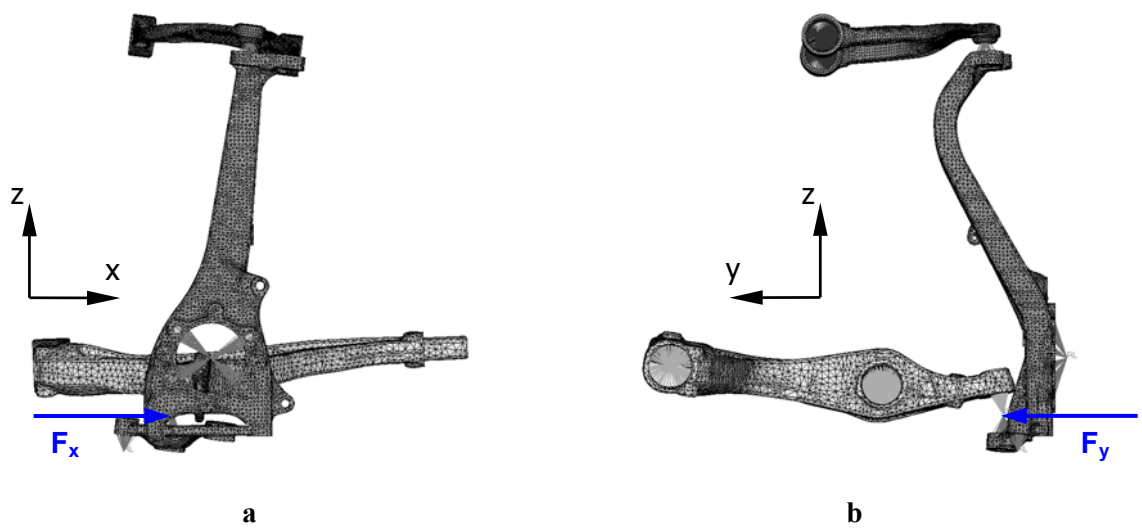
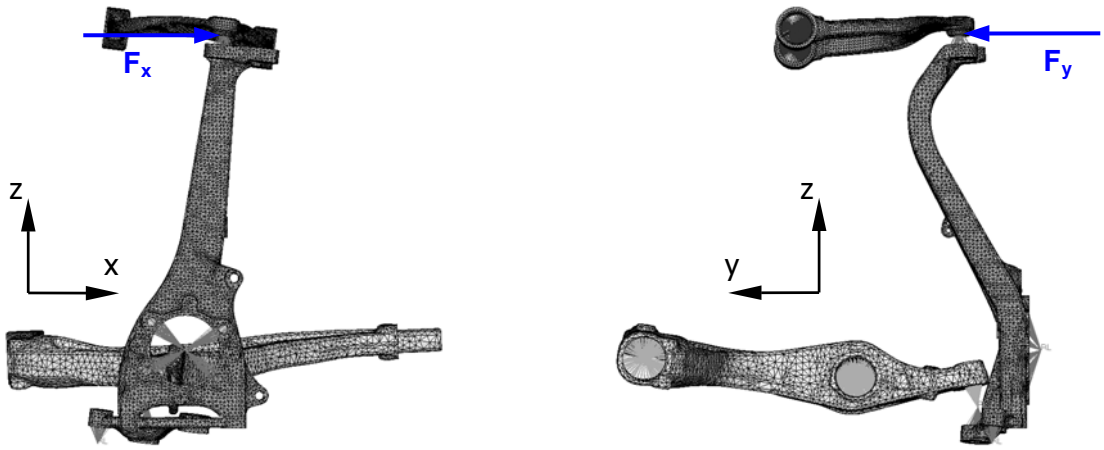


Figure 9.6 – Forces applied on the lower control arm

9.4.2 Upper control arm

The load cases applied to the upper control arm aim to allow the measurement of the elasticity of the outer hardpoint when forces in the two horizontal translational DOF are applied. By applying the force in the *x* direction (**Figure 9.7.a**) is related with the longitudinal wheel centre compliance and in a lesser degree with toe compliance, since there is an important offset of the force with the wheel centre. When the force is applied in the *y* direction (**Figure 9.7.b**) with a small offset with respect to the wheel centre, the load case is somewhat related with the lateral wheel centre compliance and the camber compliance.



a **b**

Figure 9.7 – Forces applied on the upper control arm

9.4.3 Wheelcarrier

The load cases applied to the wheelcarrier intend to measure the elasticity of the wheel centre when forces and moments in all directions are applied. By applying the force in the *x* direction (**Figure 9.8.a**) is related with the longitudinal wheel centre compliance, and in a lesser degree with toe compliance, since there is a small offset of the force with the kingpin axis (the rotation axis of the wheelcarrier, defined by the inner hardpoints, **Figure 9.8.b**). This situation arises when the vehicle accelerates or brakes. When the force is applied in the *y* direction (**Figure 9.8.b**), the load case is related with the lateral wheel centre compliance and the camber compliance. The load case when the force is applied in the vertical direction (**Figure 9.8.b**) is related with the camber compliance.

The situation of applying a torque in *x* direction (**Figure 9.8.b**) is associated with the camber compliance and the lateral wheel centre compliance. Application a torque in the *y* direction (**Figure 9.8.a**) is related with the longitudinal wheel centre compliance. Finally, applying a torque in the *z* direction (**Figure 9.8.c**) is connected with the toe compliance and longitudinal wheel centre compliance.

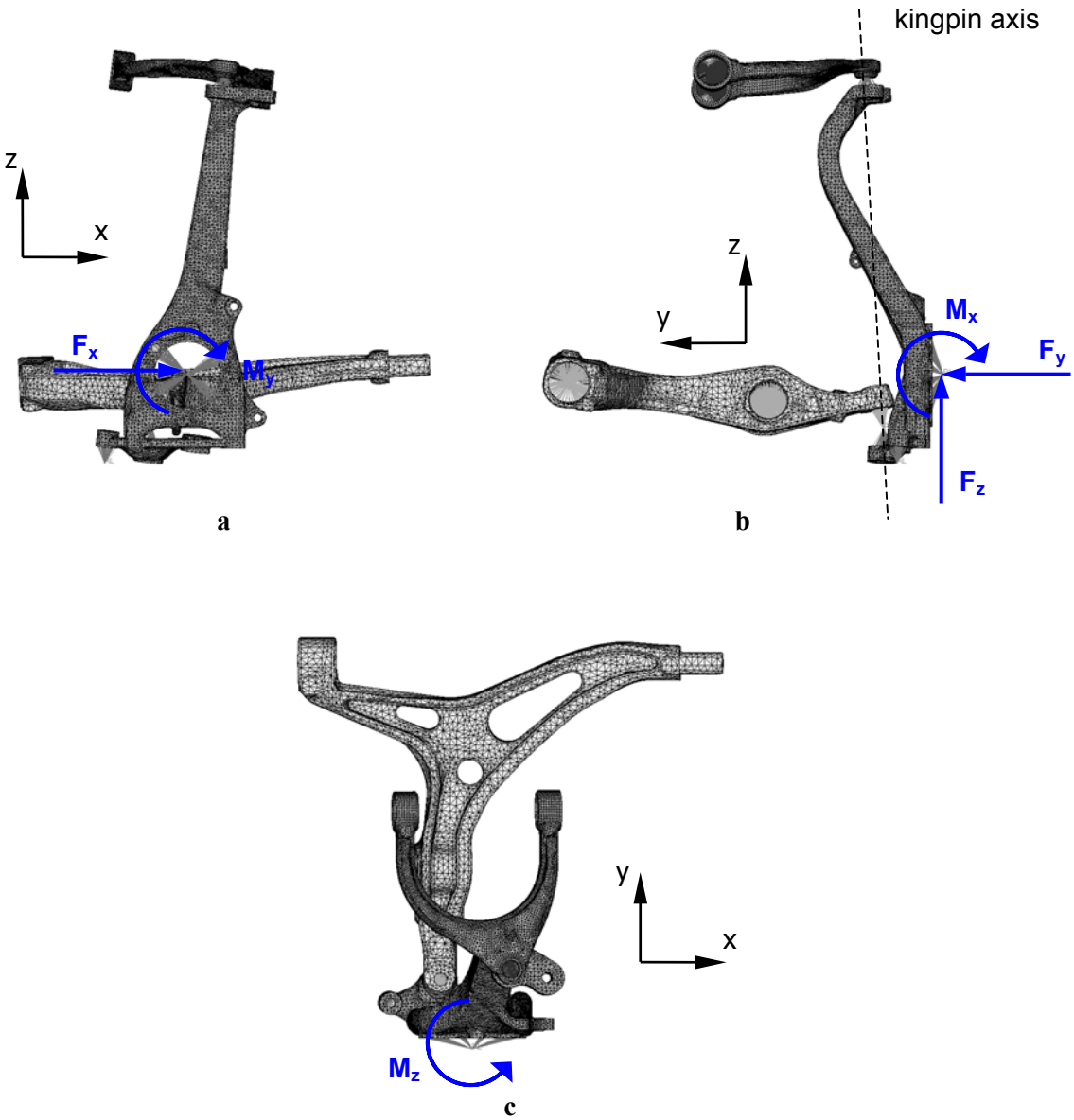


Figure 9.8 – Forces applied on the wheelcarrier

10 PROCESS STEPS

The optimisation process is managed by MATLAB with a series of six executable files (called m-files, with extension .m). These files are named phase_1.m to phase_6.m, and they make use of other m-files created ad hoc as well as functions, both built in and custom made. A description of each of the phases used to carry on the optimisation task is presented in this chapter.

The optimisation process developed is the result of taking the most suitable alternatives described in previous chapters and adding intermediate steps or procedures in order to speed up the process. But before MATLAB takes control and the user interacts only with it, reference simulations in ADAMS must be performed to produce the files that later on will be used for the rest of the process.

Once those first simulations are performed MATLAB runs the process that takes each of the component FEM files and condenses them to their hardpoints. Each of these hardpoints has 6 degrees of freedom, so each condensed component will have as many DOF as 6 times its hardpoints, and so many columns/rows will have the stiffness matrix corresponding to that component. These actions are performed by the file phase_1.m, which starts by showing the user a series of dialog boxes that allow the software to gather information about the process itself and the model to optimise, some necessary parameters, and to determine the file management by specifying the working directories of the different programs involved, namely MATLAB, ADAMS and Nastran.

From these condensed components, the stiffness matrices are systematically perturbed and the measurements taken, in order to determine which DOF are important for the behaviour of the suspension and which are not. This step, performed by the file phase_2.m, becomes necessary since a typical suspension (comprising control arms, wheel carrier and even subframe) may have up to several dozens of hardpoints and, as it has been seen in the introduction to the optimisation methods available, the number of DOF or parameters to enter the optimiser would be too high to be efficiently handled, if possible at all. A sensitivity analysis becomes then necessary in order to downsize the problem. The gain is also seen in terms of computation time, since the sensitivity analysis takes much less time to eliminate DOF than for the optimiser to handle them, as seen in **Figure 10.1**. The curves are the result of a model that takes into account the basic process to run the condensation of the components and the simulations for the reference measurements (independent of the number of DOF), the time to run the simulations for the sensitivity analysis, and the access time of the optimisation algorithm to assess possible solutions, which increases geometrically with the number of design parameters.

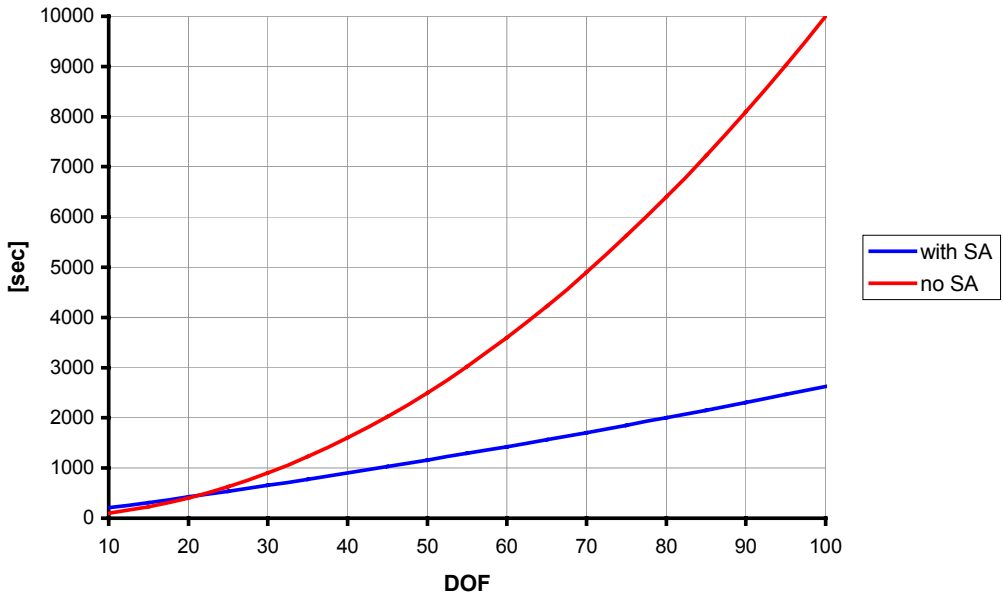


Figure 10.1 – CPU time for the optimisation process, with and without Sensitivity Analysis

For a suspension assembly with up to approximately 20 or 25 DOFs, the sensitivity analysis requires more computation time than that used by the optimiser to handle the corresponding amount of parameters during the optimisation stage. As mentioned, if an assembly includes several flexible components the amount of DOFs can rise well above 50, making the sensitivity analysis more than advantageous. Typically, about 80% of the DOFs prove to be of no relevance for the performance of a suspension[9].

The DOFs that make no contribution to the changes in performance of the suspension go to a *minimisation process*, performed by the file phase_3.m. It consists in weakening the unimportant DOFs as much as possible, that is multiplying their corresponding columns/rows in the stiffness matrices by the smallest modification factor allowed by the user.

The next step is to investigate the effect of each of the remaining (important) DOFs on the behaviour of the measurements through out the entire range of possible values for the modification factor. This range of values goes from a lower limit, set by the user, up to 1; as discussed, higher values would imply to make the component stiffer and thus heavier, which is contrary to the goal of the process. In the file phase_4.m the knowledge database is built, where each of the important DOFs is multiplied by a modification factor going from the lowest limit to 1 in steps, and the changes on the measure

[9] The exact amount of DOFs that makes it worth it to implement a sensitivity analysis depends on a number of factors such as the computer characteristics, model under optimisation, number of constraints and simulations, optimisation method chosen, etc. Although the presented data is for concept introduction purposes only, they do correspond to the experience of the author with the models used during the realisation of this thesis.

ments are saved in a table. This table is looked up during the optimisation stage carried out through the execution of the file phase_5.m by means of a genetic algorithm, in order to predict the changes in the measurements caused by a set of variations on the DOFs. The improvement is measured by a fitness function as a combination of two factors:

- average modification factor, interpreted as the weight of the optimised component,
- error, or closeness to the target values for the measurements.

After a given number of evaluations, the best found set of modification factors is applied to the stiffness matrices and the optimised components are used in a simulation to determine the actual change on the measurements, in opposition to the expected change predicted by the genetic algorithm based on the knowledge database. Finally, the results are analysed and, if good enough, the static analysis is run and the new stiffness targets issued. This final stage is performed by the file phase_6.m to produce the targets for each of the components under optimisation.

10.1 Process set-up

The user runs a first simulation in ADAMS in interactive mode, to allow the following simulations to be performed.

Step 1 – The user copies all the .mnf files attached to the flexible bodies in the assembly from ADAMS/Cars database to Phases\flex_bodies directory. This step is necessary since the optimisation process requires generating and replacing the existing .mnf file at every iteration; by doing so, the original .mnf files are preserved in the database.

Step 2 – The user opens the model in ADAMS/Cars and changes the .mnf files attached to each flexible body from those in the ADAMS/Cars database directory (**Figure 10.2**) to those in <matlab_dir>\flex_bodies directory (**Figure 10.3**).

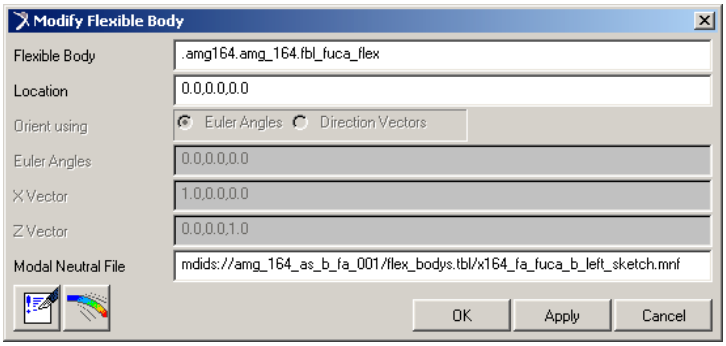


Figure 10.2 – Example of original location of the .mnf files associated with the flexible bodies

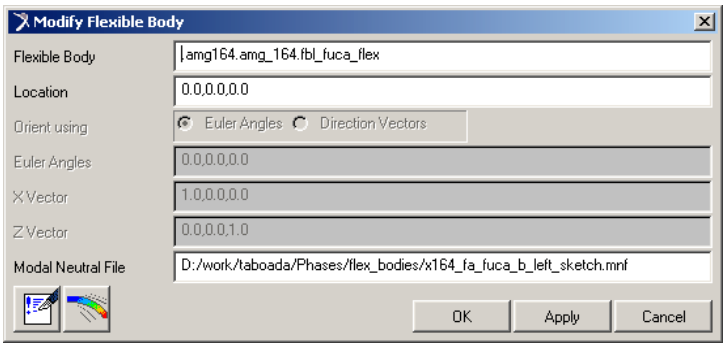


Figure 10.3 – Example of new location for the .mnf files associated with the flexible bodies

The **Inertia modelling** of every component has to be switched to **partial** or **full coupling**, the same for all components in the assembly, and the settings in file phase_1.m should be changed accordingly. When the .mtx file is produced, a total of nine invariants are included if full coupling is selected. The difference between partial and full coupling is that in the former the invariants 5 and 9 are skipped; that means that the gyroscopic effects of the components are ignored, which for static simulations are not needed. This speeds both the creation of the .mtx file and the simulations.

Step 3 – The user runs simulations in interactive mode in ADAMS/Cars (**Figure 10.4**).

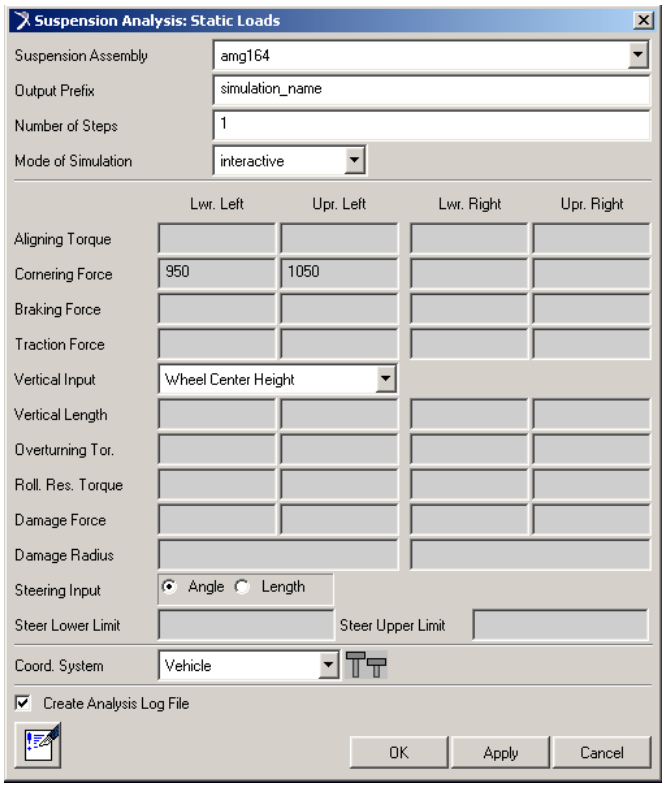


Figure 10.4 – Entering the parameters for the simulation in ADAMS

The reference measurements are read and the .adm, .acf and .mtx files are generated, which are needed to perform the simulations in batch mode later on during the optimisation process.

Step 4 – The user exits ADAMS.

10.2 Reference data collection

The user executes the m-file phase_1.m from MATLAB which, through dialog boxes, collects information about the assembly and its components (name and condensation nodes), the simulations to run and the measurements to take. Next MATLAB generates all the necessary files for Nastran to perform SOL103 1st and 2nd run, and the condensed .bdf. Finally, MATLAB calls Nastran and generates the stiffness matrices in .pch files, and the target .mnf and .mtx files.

Step 6 – With dialog boxes (**Figures 10.5** and **10.6**) in the file phase_1_loop.m, the components to include in the optimisation process and their condensation nodes are specified, and their .bdf files and associated .mnf and .mtx files are selected. Each one of these .bdf files will be a *component*.

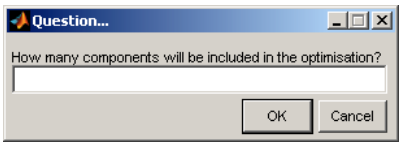
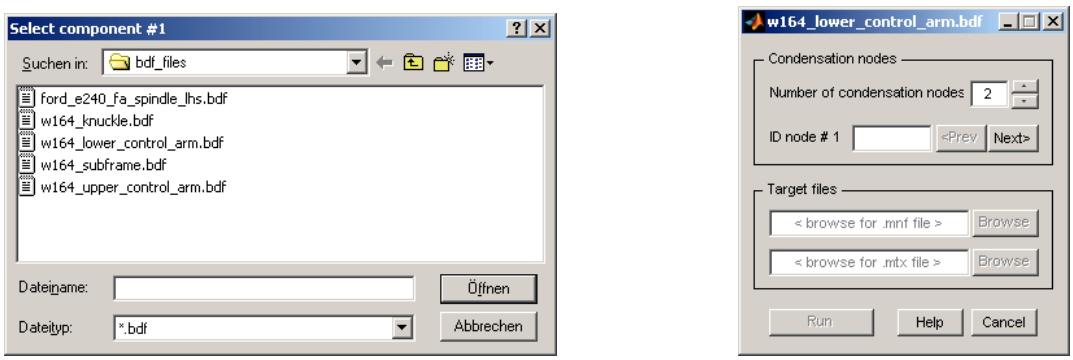


Figure 10.5 – Dialog box to input the number of components to be optimised



a **b**

Figure 10.6 – Dialog boxes to select the components (**a**), their condensation nodes and the associated .mnf and .mtx files (**b**)

The file phase_1_loop.m consists of several steps:

Step 1_1 – Launches the dialog box shown in **Figure 10.6.b**.

Step 1_2 – The file condensed_*component*.bdf is made, including the file *component*.pch and:

- the grid for the condensation points, which is searched by MATLAB in the original .bdf file,
- the outline data:
  - material properties,
  - beam properties,
  - connectivity.

The outline data is simply a series of elements of negligible stiffness properties (of about 10e-6 times the Young's modulus of the mesh) interconnecting the hardpoints. This is needed because otherwise there is nothing relating one point with each other from ADAMS/Cars point of view.

Step 1_3 – The file bdf2pch_template.dat is used to generate the file bdf2pch.dat.

Step 1_4 – Based on the file pch2mnf_template.dat, the file pch2mnf_*component*.dat is created including the files *component*.pch and condensed_*component*.bdf.

Step 1_5 –Nastran is run in batch mode calling the file bdf2pch.dat to generate the file bdf2pch.pch.

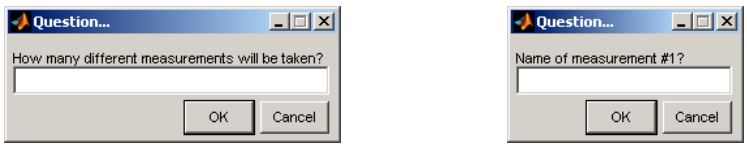
Step 1_6 – The file bdf2pch.pch is renamed to *component*.pch.

Step 1_7 – Nastran scratch files are deleted as well as the file bdf2pch.dat, which is no longer needed since the stiffness matrix was already generated based on the .bdf file, and the optimisation method is based on the modifications over the reduced stiffness matrix, not the entire model therein contained.

Step 1_8 – The m-file generate_mnf_and_mtx.m is used to generate the target .mnf and its associated .mtx files.

Step 1_9 – The files *component*.pch are renamed as *component*_ref.pch. This way the original .pch files containing the original stiffness matrices are kept for reference.

Step 7 – MATLAB collects information about the measurements to take for the optimisation process, which includes number and name of the simulations, number, name and location of the measurements in the request file produced by ADAMS/Cars and if they are angles or lengths (**Figures 10.7** to **10.10**).



a **b**

Figure 10.7 – Dialog boxes to specify the number (**a**) and denomination (**b**) of the measurements

a **b**

Figure 10.8 – Dialog box to specify the position of each step of the measurement in the request file (**a**) and to specify the type of measurement (**b**)

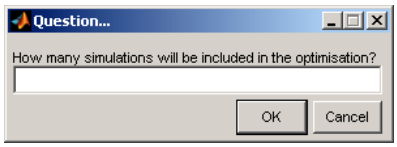
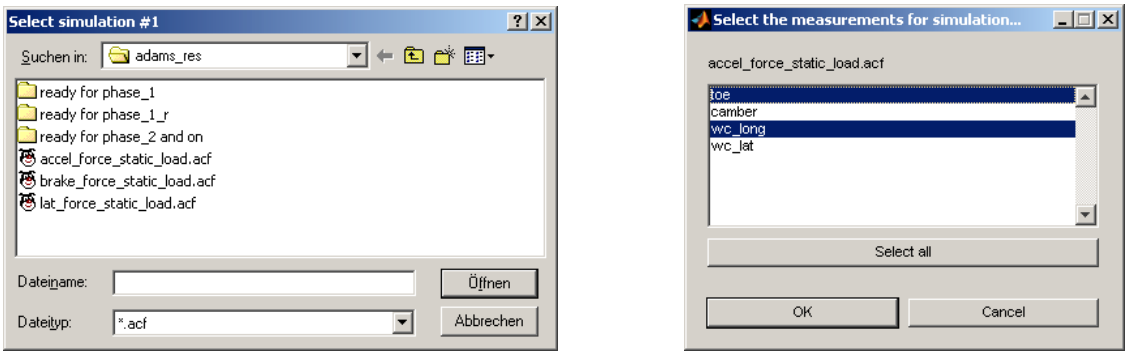


Figure 10.9 – Dialog box to specify the number of simulations to perform for the constraints of the optimisation



a **b**

Figure 10.10 – Dialog boxes to select the simulations (**a**) and the measurements to take (**b**)

Step 8 – MATLAB deletes the scratch files from the interactive simulation from ADAMS results directory, except those with extension .adm and .acf, necessary to run the simulations in batch mode later on.

Step 9 – MATLAB collects information (**Figure 10.11**) to be used during the optimisation process. It asks for the default modification factor to be used for the sensitivity analysis and as lower limit when building the knowledge database. It also asks for the number of steps to be used when building the knowledge database. Lastly, the percentage limit to sort out the important DOF is requested.

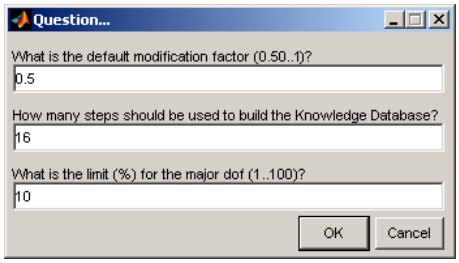


Figure 10.11 – Dialog box to declare the default modification factor for the sensitivity analysis, the steps for the knowledge database, and the tolerance to sort out the DOF

Step 10 – MATLAB runs the reference simulation in ADAMS, reading and storing the reference measurements in the table mea_ref.

10.3 Sensitivity analysis

With the execution of the m-file phase_2.m from MATLAB, the sensitivity of the measurements to each DOF is determined. The procedure is to systematically perturb each DOF by multiplying its corresponding row and column by the default modification factor. This factor is entered by the user in phase 1. As a starting point, a value of 0.5 us suggested.

Step 11 – The reference stiffness matrices from the files *component*_ref.pch are read and stored in the array k_matrix_ref.

Step 12 – A sensitivity analysis is run for each DOF of each component, storing the result in the arrays mea_sea_raw. Every time a simulation is run, copies of the reference .mnf and .mtx files are saved before the analysis and retrieved afterwards. In this way the reference ones are always kept.

10.4 Minimisation

With the execution of the m-file phase_3.m, MATLAB takes the data collected from the sensitivity analysis and determines which DOF can be weakened in the stiffness matrices without affecting the measurements. The tolerance to sort out the DOF is set by the user in phase 1, step 9. The degree of influence of this tolerance in the results is somewhat unknown, and must therefore be used with caution, trying different values until a satisfactory result is achieved. The default value is 10%.

Step 13 – The results of the sensitivity analysis are arranged and sorted out in the vectors minor_dof and major_dof, according to the tolerance set by the user.

Step 14 – The stiffness matrices are minimised and stored in the array k_matrix_min.

Step 15 – Copies of the original .mnf and .mtx files are saved. The files build_modified_pch.m and generate_mnf_and_mtx.m are executed to build the necessary .pch file containing the modified stiffness matrix and generate the corresponding .mnf and .mtx files necessary for the simulation.

Step 16 – The simulations are run in ADAMS/Cars in batch mode, and the resulting measurements are read and stored in the matrix mea_min.

Step 17 – The original .mnf and .mtx files are restored, overwriting the modified files.

10.5 Knowledge database

MATLAB builds a table with how much each measurement changes when each DOF is perturbed, using the m-file phase_4.m. The difference with the sensitivity analysis performed in phase 2 is that the perturbation is not one value but several, from the default modification factor (used as lower limit) to 1, in steps determined by the user in the MATLAB code. The sensitivity is studied only at some points and the curve is then approximated by a polynomial.

Step 18 – The sensitivity analysis is run using a range of values for the modification factor, from the minimum to 1, and stores the results in the array mea_sea_min_study_aux.

Step 19 – The multiple sensitivity analysis is completed by approximating the 3 points in the curve with a second order polynomial. The results are stored in the array mea_sea_min_study.

10.6 Genetic algorithm optimiser

Using the m-file phase_5.m, MATLAB executes a Genetic Algorithm (GA) optimiser to search for the best combination of modification factors that delivers the lowest weight (target) whilst maintaining the measurements (constraints).

Step 20 – All the necessary parameters for the genetic algorithm are set.

Step 21 –The first population for the GA is randomly generated and the fitness is rated. There is the option to include a very fit individual from a previous generation.

Step 22 – The three operators of the GA are applied:

- reproduction,
- crossover,
- mutation.

The genetic information of the best individual generated at any generation is stored in the variable gene_value_best. At every generation MATLAB informs the user about the elapsed number of generations.

Step 23 – MATLAB modifies the reference stiffness matrices according to the findings of the optimisation, and the resulting matrices are stored in the array k_matrix_opt.

Step 24 – Copies of the original .mnf and .mtx files are saved. The files build_modified_pch.m and generate_mnf_and_mtx.m are executed to build the necessary .pch file containing the modified stiffness matrix and to generate the corresponding .mnf and .mtx files necessary for the simulation.

Step 25 – MATLAB executes simulate_and_read.m to simulate and read the resulting measurements. The new values are stored in the matrix mea_opt.

Step 26 – MATLAB restores the original .mnf and .mtx files.

Step 27 – MATLAB calculates the estimated targets for the best individual, according to the knowledge database.

10.7 Static analysis

In this phase of the optimisation process, MATLAB uses the m-file phase_6.m to produce the new targets for the components in order to pass them to the topology optimisation stage. For this reason, it runs a static analysis (Nastran SOL101) on the best individual achieved by the genetic algorithm in the previous phase. The load cases for each component are entered by the user.

Step 28 – Information about the static analysis is collected. A dialog box (**Figure 10.12**) is launched to select the results from the topology optimisation, according to criteria set by the user to decide which results are worth to use. It also asks (**Figure 10.13**) if the same load cases from a previous run (if there was any) should be used.

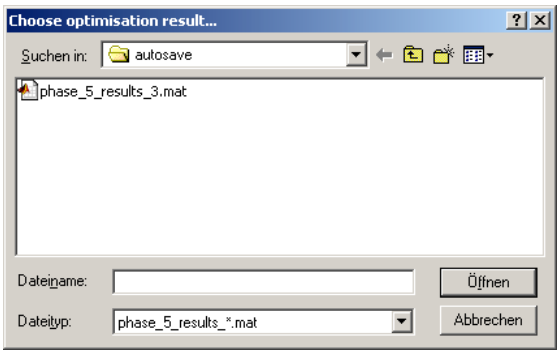


Figure 10.12 – Dialog box to select the results from the GA optimisation

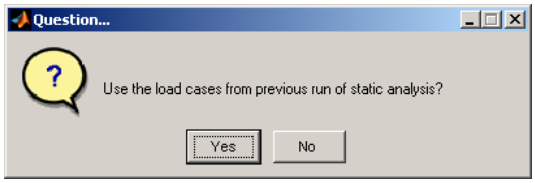


Figure 10.13 – Dialog box to allow the use of previous runs of the static analysis

If the user chooses No in the previous dialog box (because there is no previous run or because there are new load cases to be used) a dialog box (**Figure 10.14**) is launched to set the number of load cases applied and the node (hardpoint) where the measurements are taken in each component. A second dialog box (**Figure 10.15**) is used to select the load cases.

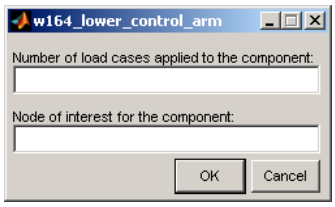


Figure 10.14 – Dialog box to enter the number of static load cases to apply to the components and the hardpoints in which the measurements are taken

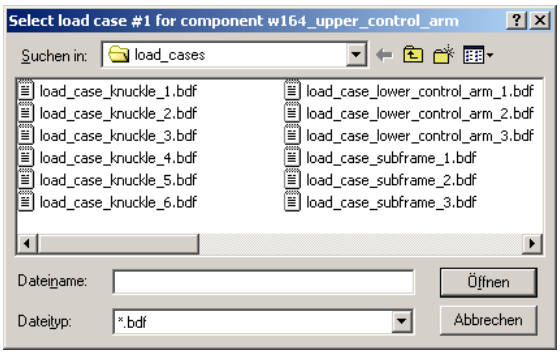


Figure 10.15 – Dialog box to select the static load cases to apply to the component

Step 29 – The load cases are applied to the reference components by calling Nastran in batch mode. The results are stored in the array sol101_set_ref.

Step 30 – The stiffness matrices for the optimised components are built.

Step 31 – The load cases are applied to the optimised components by calling Nastran in batch mode. The results are stored in the array sol101_set_opt.

Step 32 – The results are rearranged in the matrices sol101_set_ref and sol101_set_opt to improve the readability.

11 OPTIMISATION CASE

Once the optimisation process has been programmed an optimisation case is run in order to draw attention to the capabilities and limitations of the developed method. As it was determined, the process downsizes the problem by running a sensitivity analysis, and then finds the combination of modification factors that best satisfies the goal and constraints of the problem.

In the present chapter a suspension assembly is presented and used as a test mule to run the optimisation process. The results of each optimisation stage are reported and discussed, with particular emphasis on those implemented in phase_5.m and phase_6.m, the genetic algorithm and the static analysis, respectively.

11.1 Process set-up

The first step in the optimisation process is performed by initialising ADAMS. The assembly containing the suspension to optimise (**Figure 11.1**) is loaded, and the three basic simulations described in Chapter 8 are performed: lateral force, acceleration force and braking force. Once the three simulations are finished without errors, ADAMS is closed.

The output of this phase are the two files necessary to run ADAMS simulations in batch mode later on during the optimisation process. Another important result of this phase are the .mtx files generated for the simulation based on the .mnf files. Of these .mtx files, those corresponding to the components under optimisation will not be actually used (they will be replaced by the ones generated during the optimisation process based on the modified stiffness matrices of the components) but they are useful since they provide the name of the generated .mtx files, which ADAMS defines with its own criteria.

This stage also offers the user the opportunity to check that everything in the model is correct and that the simulations actually run without any inconvenient.

11.2 Reference data collection

For this stage and all the stages that follow MATLAB is used to control the process. First, MATLAB is initialised and the file phase_1.m is edited. At the beginning of this file all the directories regarding the process are specified: location of installation folders for MSC.Nastran and ADAMS, working directories for MATLAB and ADAMS (where files are created and replaced during the process, as

needed) and some parameters, such as coupling mode of the components in the assembly. Next, the file is executed. The first step is to introduce the information about the components that will be included in the optimisation, the simulations to run and the measurements to take. In this case, the lower and upper control arm are the components included, with 4 and 3 hardpoints respectively. Finally, the simulations are run and the reference measurements taken.

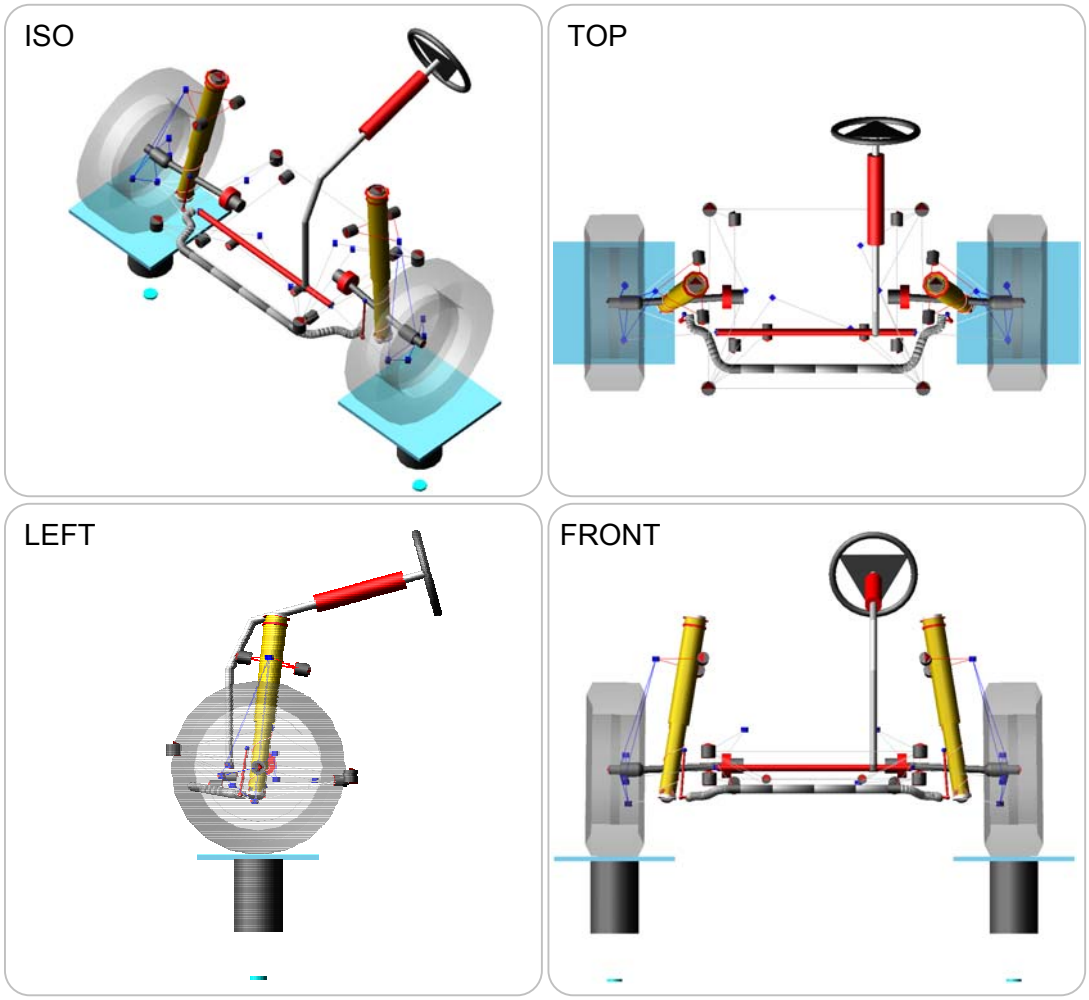


Figure 11.1 – Suspension assembly in ADAMS/Cars

The output of this stage is the reference measurements, as shown in **Table 11.1**, including their name for every cell. These values do not correspond with any particular requirement from a manufacturer. They simply show the actual performance of the suspension assembly at the stage of development and will be taken as targets for the optimisation task. The values are given with an accuracy of 4 digits only, because it makes no sense to worry about more exactitude, although in the computer they are handled with higher precision,. The reason is that in the real situation such small differences are neither measurable nor important for the actual performance of the suspension.

simulation	measurement [°/kN]		
acceleration force	**0.2955** toe compliance	**-2.4818** longitudinal wheel centre compliance	
braking force	**-0.1979** toe compliance	**2.2768** longitudinal wheel centre compliance	
lateral force	**0.1583** toe compliance	**-0.1941** camber compliance	**-0.2231** lateral wheel centre compliance

Table 11.1 – Reference measurements

Other output at this stage is the original stiffness matrix of each component, while the user enters the lowest modification factor applicable to a DOF (0.5), the amount of steps in which to divide the search space for each modification factor (16), and the limit (10%) for the lowest influence of a DOF in comparison with the most important, in order to be also considered important.

11.3 Sensitivity analysis

In this stage the user plays no role and MATLAB reads the stiffness matrix of each component and determines the influence of each of the 42 DOFs in each of the 7 measurements. The stiffness matrices are presented in **Figure 11.2**, where the values are replaced by a colour scale for ease of interpretation, since the actual figures have no particular importance here.

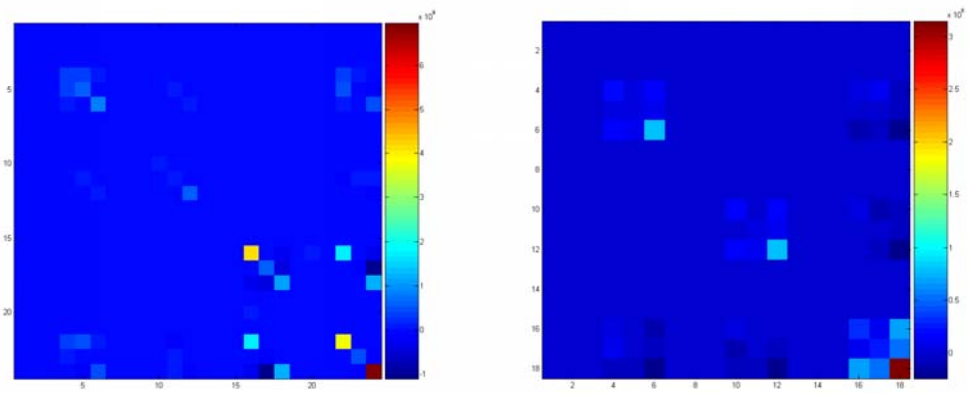


Figure 11.2 – Representation of the stiffness matrices of lower (left) and upper (right) control arms

The vertical and horizontal axes indicate the DOFs of each component, and refer to the cells indexes in the stiffness matrices. The first 6 DOFs (1 to 6) correspond to the first hardpoint of the component, first the translations and then the rotations in the three directions (*x*, *y* and *z*). The second 6 DOF (7 to 12) correspond to the second hardpoint, and so on. For example, in the left chart of the **Figure 11.2**, DOF 9 corresponds to the second hardpoint of the lower control arm (named 5002 as in **Figure 9.3**), and it corresponds to a translation in the *z* direction.

What is important to note is that the charts are symmetric, as it is expected from the four properties described in Section **2.2**. Specially in the upper control arm it can be noticed that the values corresponding to the rotational DOF (which seem like squares of 3×3 on the chart) present higher values than the rest of the matrix, in particular for DOF 16, 17 and 18, which correspond to the rotational DOF of the hardpoint 6003 (**Figure 9.4**), that is, the one in the middle of the upper control arm. Obviously, for the three hardpoints it can be intuitively noticed that they are harder to twist around a vertical axis. This is reflected in the higher values of DOF 6, 12 and 18. Similarly, DOF 24 in the lower control arm stiffness matrix presents the highest value, and it corresponds to the rotational DOF around the vertical axis of the hardpoint 5004 in the middle of the component, where the suspension strut meets the arm.

Secondly, the top of the scale for the lower control arm (10^9) is one order of magnitude higher than for the lower control arm (10^8), which is due to the lower solicitations of this component with respect to the first. A reason is that the upper control arm is made of aluminium (Young's modulus: 70.000 N/mm^2) while the lower control arm is made of steel (Young's modulus 170.000 N/mm^2).

The results of the sensitivity analysis are given in **Table 11.2**. For convenience of space in the columns of the table, the names in the seven measurements have been replaced in the first row by their abbreviations, as follows:

- Δtc = toe compliance change,
- Δcc = camber compliance change,
- Δlong_wc = longitudinal wheel centre compliance change,
- Δlat_wc = lateral wheel centre compliance change.

In the two columns under the label "DOF", the first column indicates the component, where:

- 1 = lower control arm,
- 2 = upper control arm.

The second column indicates the DOFs of every component. The first 6 DOFs (1 to 6) correspond to the first hardpoint of the component, the second 6 DOFs (7 to 12) to the second hardpoint, and so on.

DOF	acceleration force		braking force		lateral force		
	Δtc	Δlong_wc	Δtc	Δlong_wc	Δtc	Δcc	Δlat_wc
1 1	0.000687549	-0.0024	-0.001300614	0.00444	0.000068755	0	0
1 2	-0.000802141	-0.0007	0.00257831	0.00218	0.00311689	-0.001260507	-0.00748
1 3	0.000171887	-0.0007	0.000045837	-0.00044	-0.000057296	0.000057296	0.00009
1 4	0.000057296	0.0001	0.00000573	-0.00001	-0.00000573	0	0
1 5	0.000057296	-0.0002	0.000011459	-0.00006	-0.000028648	0.000057296	0.00004
1 6	0	0	0.000057296	-0.0001	-0.000057296	0.000057296	0.00010
1 7	-0.000114592	0.0004	0.000143239	-0.00032	-0.000034377	0.000057296	0.00002
1 8	0.003151268	-0.022	-0.004308643	0.03081	0.001799087	-0.000572958	-0.00343
1 9	0.000343775	-0.0022	-0.000126051	0.00101	0	0.000057296	0.00005
1 10	0	0	0	0	-0.00000573	0.000057296	0
1 11	0	0.0003	0	0.00004	0.000011459	0	-0.00003
1 12	0	-0.0001	-0.000011459	0.00011	0	0	-0.00002
1 13	0.000458366	-0.0241	-0.000446907	0.03922	-0.000017189	0	0.00002
1 14	0.000974028	-0.0012	0	0.00017	0.003649741	-0.001317803	-0.00823
1 15	0.000458366	-0.0034	0.000280749	-0.00148	-0.000395341	0.000114592	0.00081
1 16	0	0.0001	0	1E-05	0	0.000057296	0
1 17	0	0	0.00000573	0	-0.00000573	0	-0.00001
1 18	0	0	0	0	0	0	-0.00001
1 19	0	0	0.00000573	0.00002	0	0	0
1 20	0.000057296	0.0001	0.00000573	0.00002	0	0	0
1 21	0	0.0003	0	0.00007	0.00000573	0.000057296	-0.00003
1 22	0	0.0001	0	0.00003	0	0.000057296	-0.00001
1 23	0.000057296	-0.0003	-0.000011459	0.00029	0	0	0
1 24	0	0.0001	0.00000573	0.00004	0	0	-0.00001
2 1	0.00040107	-0.003	0.000429718	-0.00337	-0.000022918	0.000057296	-0.00013
2 2	-0.000114592	-0.0016	-0.000171887	-0.00235	-0.000045837	-0.000057296	0.00037
2 3	0.000057296	0	0.000034377	-0.00007	-0.000017189	0.000057296	0.00002
2 4	0	0.0001	0	0.00002	0	0.000057296	0
2 5	0	0	0	0	-0.00000573	0	0
2 6	0	0.0002	-0.000011459	0.00016	-0.00000573	0.000057296	-0.00001
2 7	0	-0.0031	0.000085944	-0.00595	0.000080214	0.000057296	-0.00006
2 8	0.000458366	-0.0023	0.000360963	-0.00206	-0.000160428	-0.000114592	0.00034
2 9	0	0.0001	-0.00000573	0.00002	0	0	-0.00002
2 10	0	0.0001	0.00000573	-0.00001	0	0.000057296	-0.00001
2 11	0	0	0.00000573	0	-0.00000573	0	-0.00001
2 12	0.000057296	0.0002	0	0.00021	0	0.000057296	-0.00001
2 13	0.000114592	-0.0015	0.000103132	-0.00169	-0.00000573	0	0
2 14	0.000171887	-0.0005	0.00000573	-0.00046	-0.000389611	-0.00040107	0.00127
2 15	0	0	0.000022918	-0.00009	-0.000011459	0	0.00002
2 16	0	0	0.00000573	0	-0.00000573	0	-0.00001
2 17	0	0	0	0	0	0	0
2 18	0	0	0	0	0	0	0

Table 11.2 – Results of the sensitivity analysis

Figures 11.3 and **11.4** show the results from the sensitivity analysis for the two measurements taken from the first simulation, acceleration force, and the 10% limit imposed to the measurements. As can be seen from **Figure 11.3**, weakening the component DOFs worsens the performance since the toe compliance is in most cases increased, reducing the control of the suspension over the wheel setting.

On the other hand, although the longitudinal wheel centre compliance is for most DOFs reduced, due to the sign convention it means that the magnitude of the compliance is increased, since the reference value given in **Table 11.1** is negative for this measurement (-2.4818 °/kN).

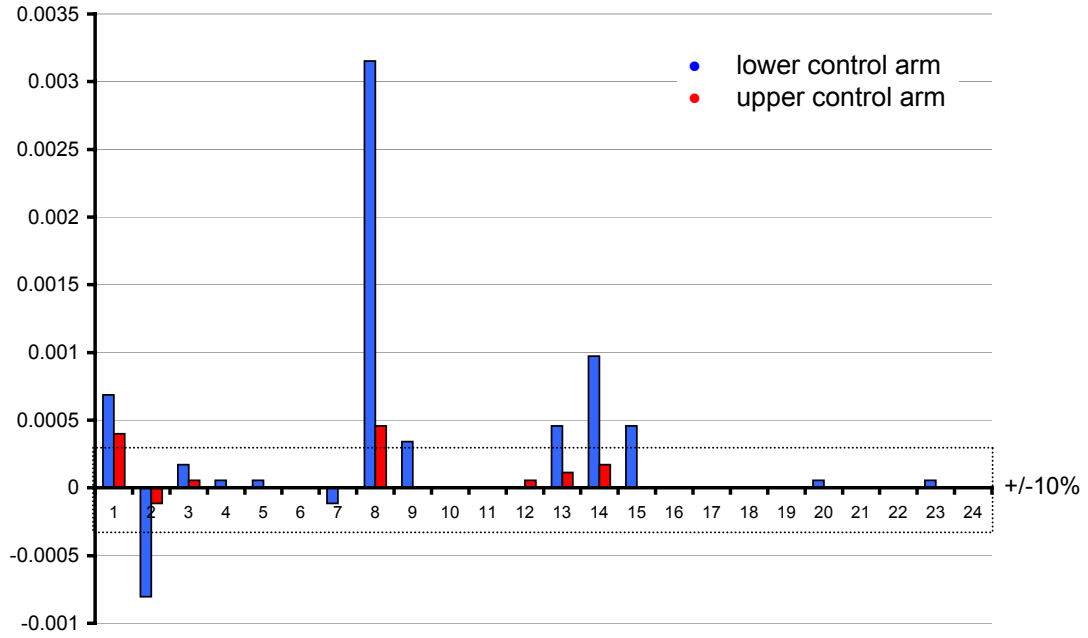


Figure 11.3 – Toe compliance change vs. DOF, under acceleration force

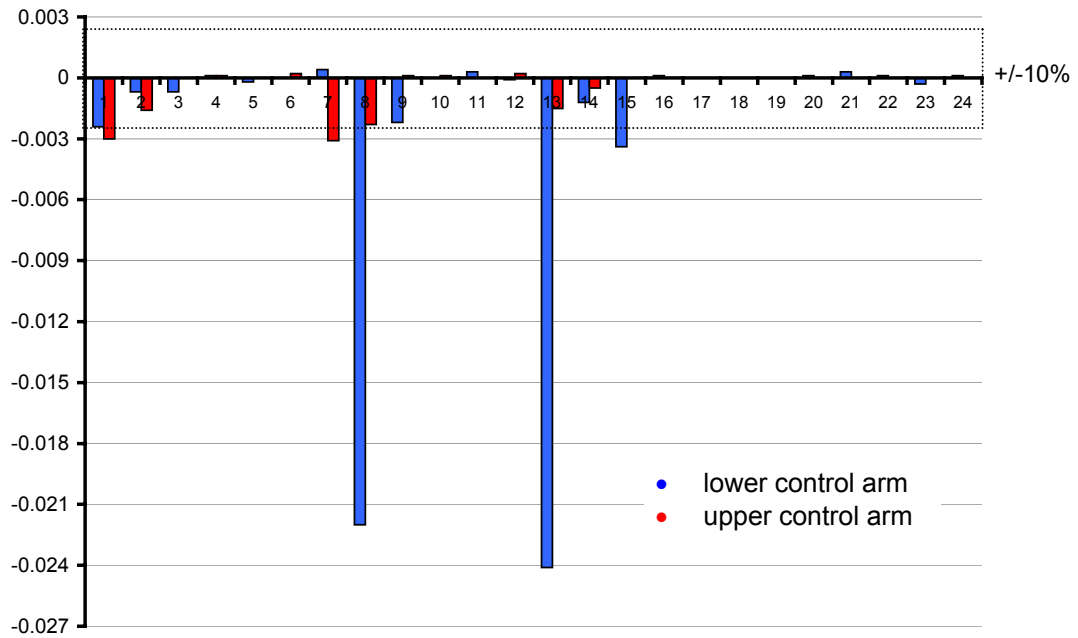


Figure 11.4 – Longitudinal wheel centre compliance change vs. DOF, under acceleration force

From the previous two figures it can be seen that, when applying an acceleration force, out of a total of 42 DOFs only the following 9 are important:

- lower control arm: 1, 2, 8, 9, 13, 14 and 15.
- upper control arm: 1 and 8.

The results of the sensitivity analysis after applying a braking force are presented in **Figures 11.5** and **11.6**. As can be seen the tendency is not so well defined in the case of the toe compliance, because two DOFs of the lower control arm, 1 and 8, increase the compliance (in absolute value), while DOF 2 decreases it significantly. The upper control arm presents, in general, the opposite behaviour: DOF 1 and 8 decrease the toe compliance, while DOF 2 increases it (always in absolute value).

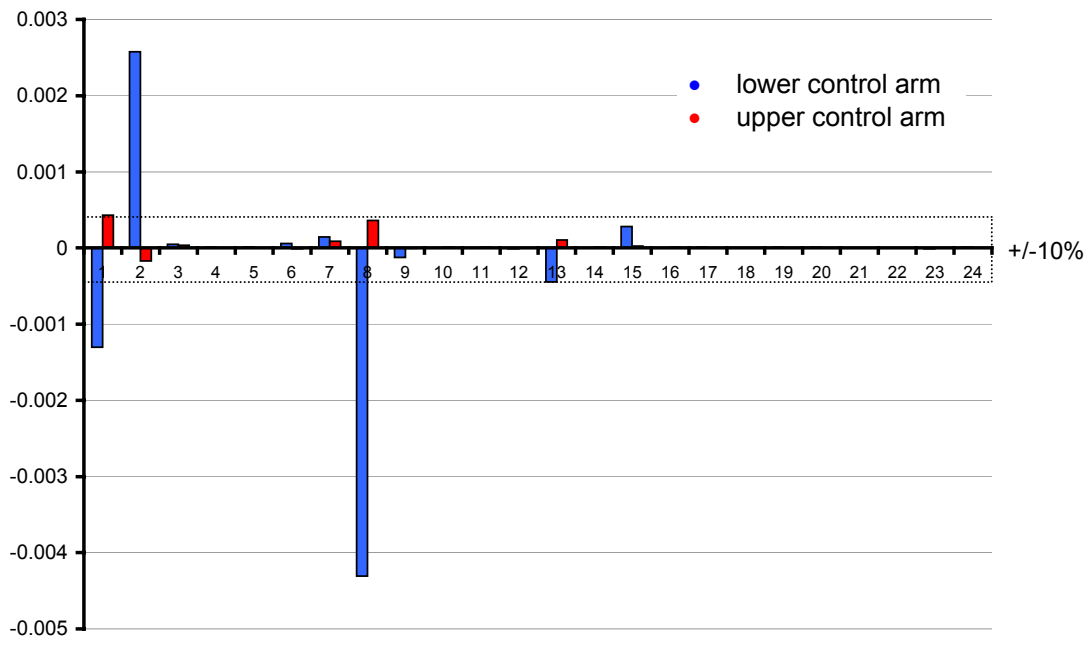


Figure 11.5 – Toe compliance change vs. DOF, under braking force

DOF 1 is for both control arms the displacement of the front hardpoint that connects them to the vehicle's body, in the *x* direction; DOF 8 represents the displacement of the rear hardpoint that connects the arms to the vehicles body in *y* direction. It seems that weakening these DOFs in the upper control arm is a benefit for wheel position control, but in the lower control arm it is the opposite. DOF 13 presents the same behaviour in a smaller amount.

Conversely, DOF 2 is the displacement in *y* direction of the front hardpoint that connects the arms to the vehicle's body; its behaviour is inverse to what it was previously discussed.

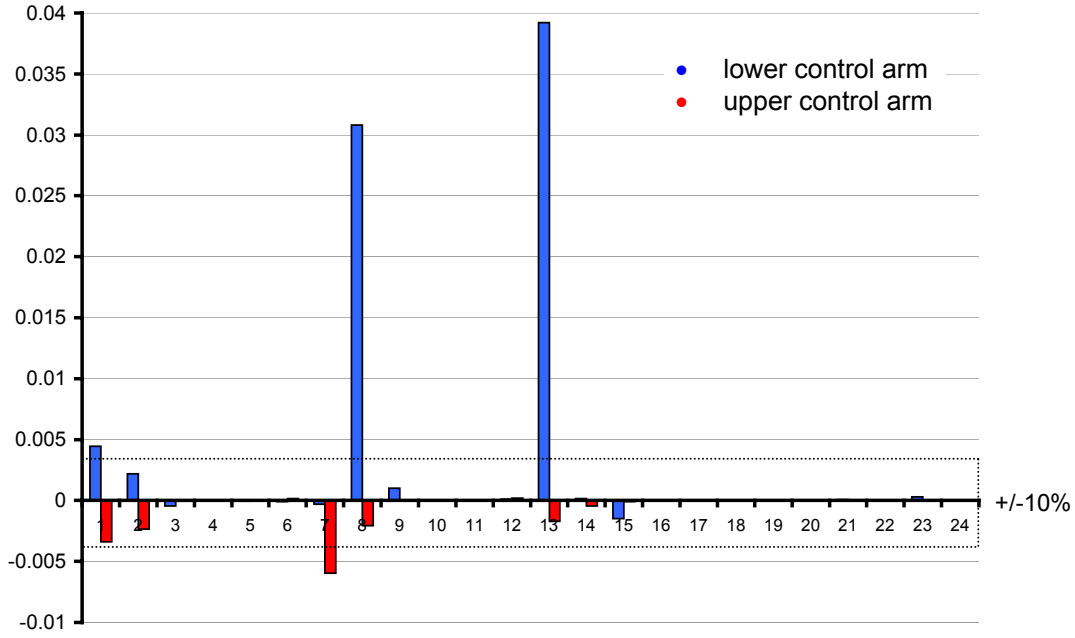


Figure 11.6 – Longitudinal wheel centre compliance change vs. DOF, under braking force

The response of both components is also opposed when it comes to longitudinal wheel centre compliance. Weakening similar DOFs in upper and lower control arm leads to opposite changes in the compliance, although the lower control arm shows much more influence in the final result, like in all the previous measurements.

From the previous two figures it can be seen that, when applying a braking force, out of a total of 42 DOFs only the following 5 are important:

- lower control arm: 1, 2, 8 and 13.
- upper control arm: 7.

Comparing the results from the acceleration and the braking forces the behaviour seems to be in conflict with each other: when a DOF is weakened, the longitudinal wheel centre compliance increases during braking and decreases during acceleration. The same can be said about the toe compliance. However, this is due to the sign convention of the measurements, as seen in **Table 11.1**: for example, to increase the toe compliance under acceleration force the variation must be positive, but under braking the variation must be negative. Hence, it can be said that the same DOF (for example, 2, 8 and 13 from the lower control arm) increase or decrease the two measured compliances in absolute value in the same way for both cases, acceleration and braking.

The next three charts (**Figures 11.7 to 11.9**) show the results of the sensitivity analysis when simulating a lateral force applied to the suspension. The first figure shows the toe compliance in which the components seem again to behave in opposite ways, except for DOF 15 from lower control arm, which

seems to have out-of-the-tendency behaviour. This DOF corresponds to the displacement in the *z* direction of the outer hardpoint of the arm, which connects it to the lower hardpoint of the wheelcarrier.

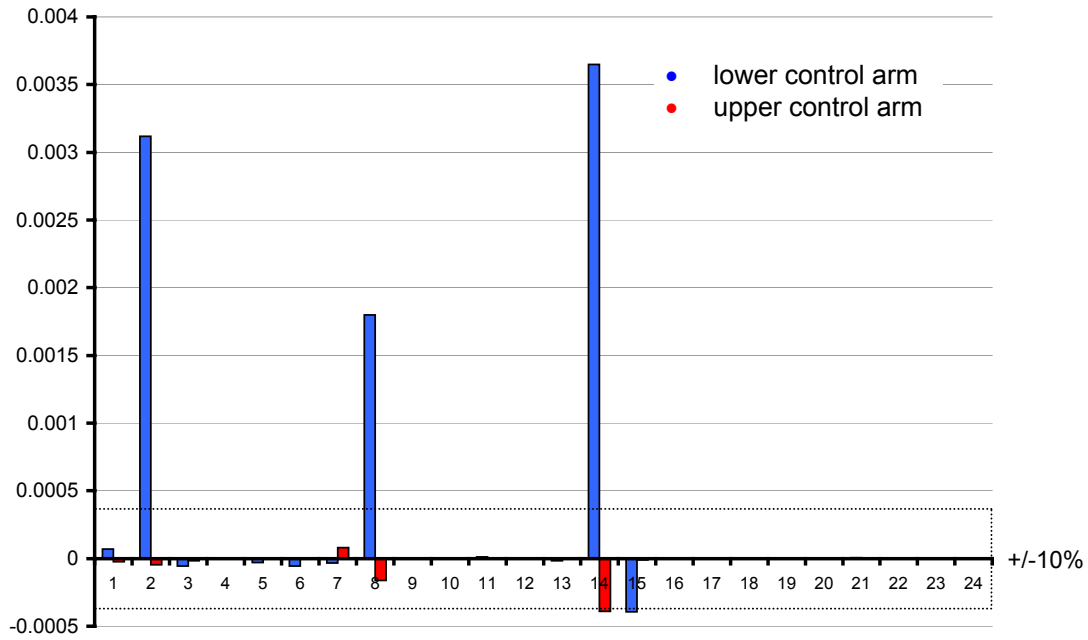


Figure 11.7 – Toe compliance change vs. DOF, under lateral force

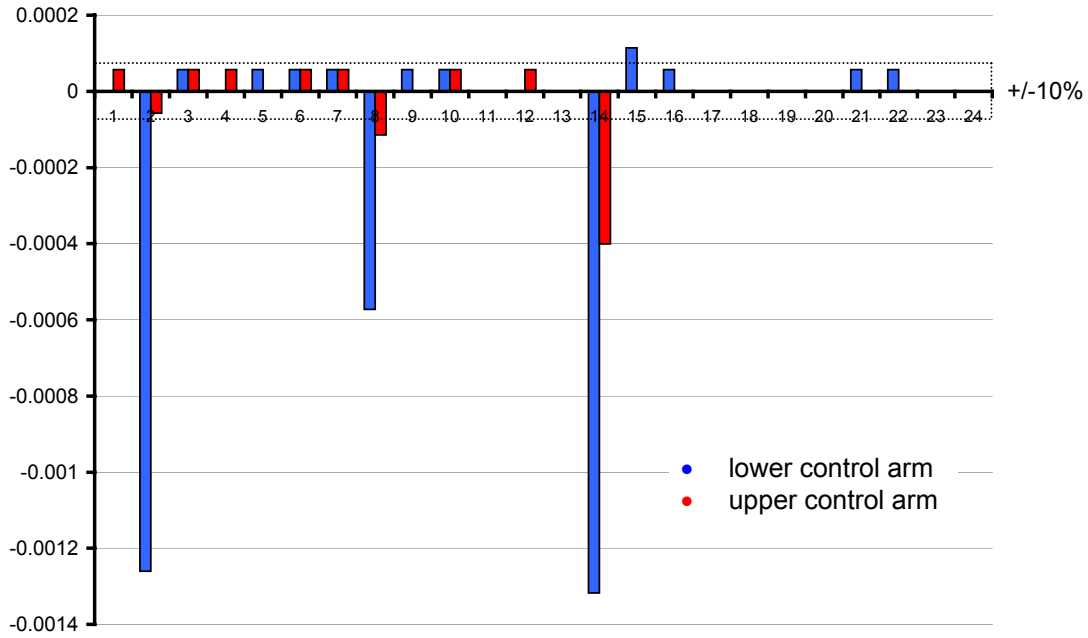


Figure 11.8 – Camber compliance change vs. DOF, under lateral force

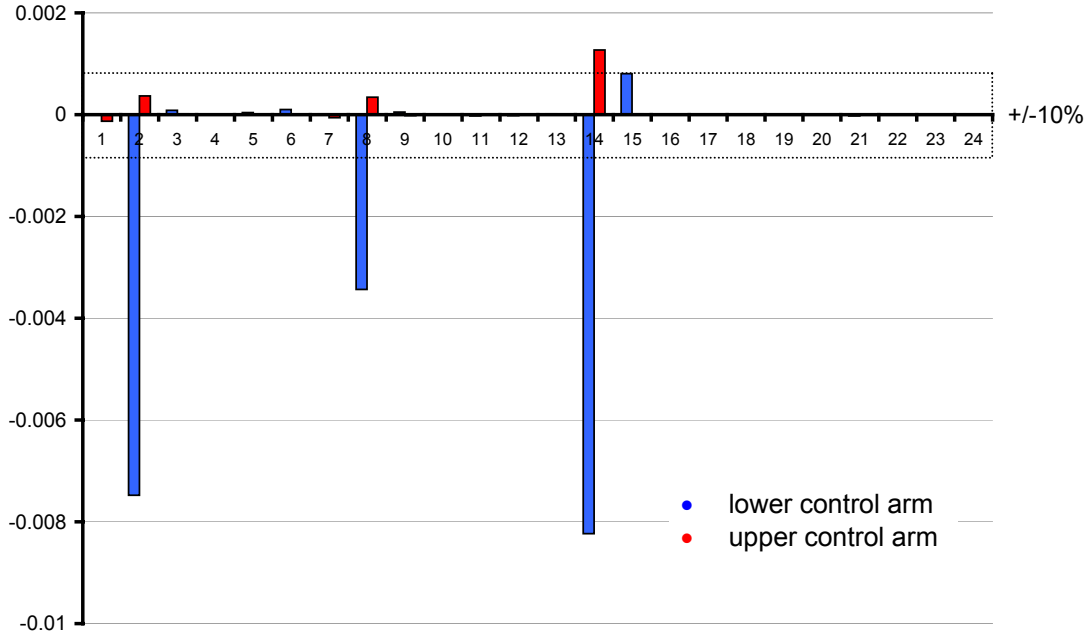


Figure 11.9 – Lateral wheel centre compliance change vs. DOF, under lateral force

It becomes evident that both camber and lateral wheelcentre compliance are dominated by a few DOFs which correspond to the lateral displacement of the hardpoints, since they change the camber setting along with their movement. From the previous three figures it can be seen that, when applying a lateral force, out of a total of 42 DOFs only the following 6 are important:

- lower control arm: 2, 8, 14 and 15.
- upper control arm: 8 and 14.

11.4 Minimisation

With the results from the sensitivity analysis, those DOFs that make a contribution to the change of the compliances smaller than 10% of the most significant DOFs (in absolute value) are weakened by the default modification factor, the same value used for the sensitivity analysis. Combining the analyses in the previous section, the components add up 11 important DOFs, as following:

- lower control arm (7 DOFs): 1, 2, 8, 9, 13, 14 and 15.
- upper control arm (4 DOFs): 1, 7, 8 and 14.

After minimising the stiffness matrices, new simulations in ADAMS are performed in order to check that the measurements did not change significantly, as predicted. The maximum variation in absolute value is less than 0.3% (**Table 11.3)**, and it corresponds to the lateral wheel centre compliance.

reference	0.2955	-2.4818	
[°/kN]	-0.1979	2.2768	
	0.1583	-0.1941	-0.2231
minimised	0.2955	-2.4854	
[°/kN]	-0.1978	2.2724	
	0.1582	-0.1941	-0.2224
variation	0.00%	0.15%	
	-0.04%	-0.19%	
	-0.11%	0.00%	-0.28%

Table 11.3 – Variation of the measurements after the minimisation process

Figure 11.10 shows the variation in the amount of DOF considered important against the limit percentage to sort them out. This agrees with the results **Figure 11.11**, where the limit to sort out the important from the unimportant DOF is moved in the range 0 (where no change on the compliances is required to be considered an important DOF) to 0.4.

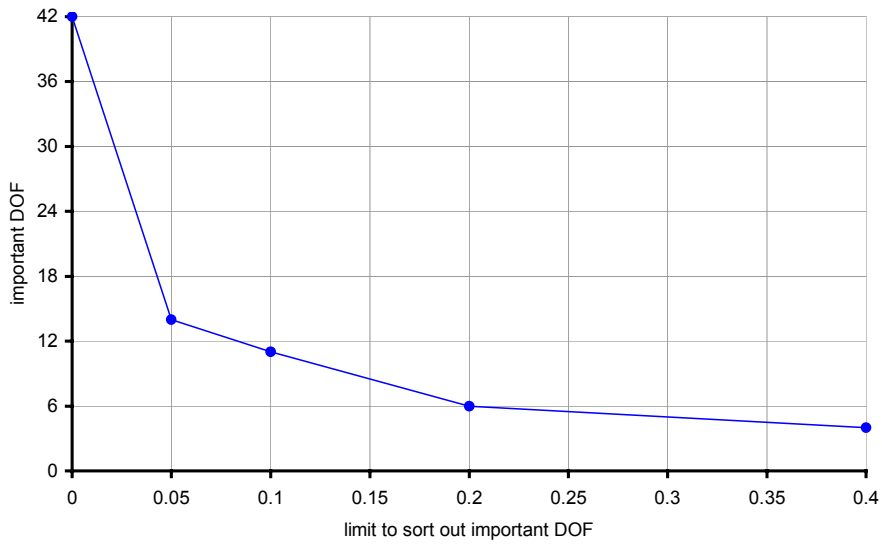


Figure 11.10 – Important DOF vs. limit for important DOF

It becomes evident that the higher this limit is, the more DOFs are considered unimportant for the optimisation since they require a bigger influence over the compliances to qualify for being considered important. As a result, more and more DOFs are multiplied by the default modification factor during the minimisation and their effect can be detected during the simulations as they increase the variations with respect to the reference values.

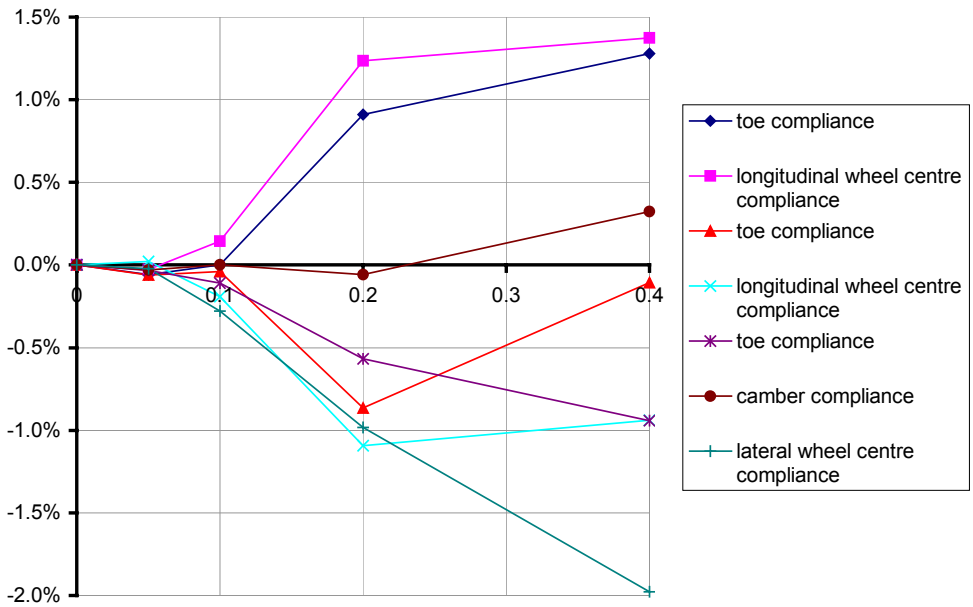


Figure 11.11 – Compliance changes due to minimisation vs. limit for important DOF

11.5 Knowledge database

MATLAB builds the database measuring the effect of every important DOF on the compliances, by applying a modification factor varying from the lowest value given by the user, to 1, in 16 steps (user entered). Hence, a total of 17 modification factors are applied and the variations of the compliances stored. However, due to the smooth behaviour of the DOFs this procedure is made faster by taking the readings only at the end values and in the middle of the interval; the intermediate values are interpolated by a second order polynomial. The results of this process are shown in **Figures 11.13** to **11.14**.

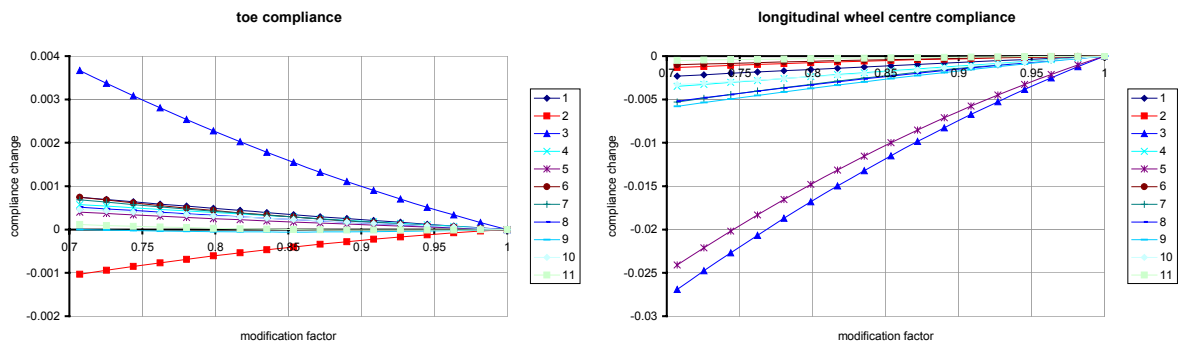


Figure 11.12 – Compliance changes vs. Modification factor, under acceleration force

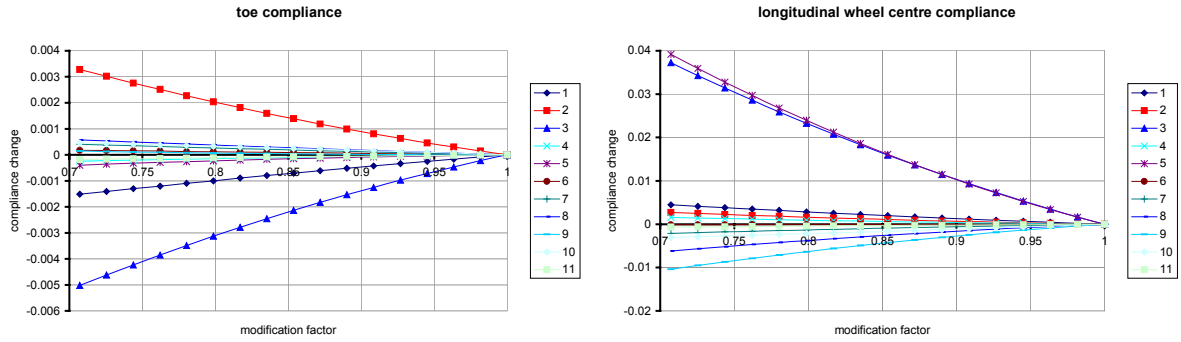


Figure 11.13 – Compliance changes vs. Modification factor, under braking force

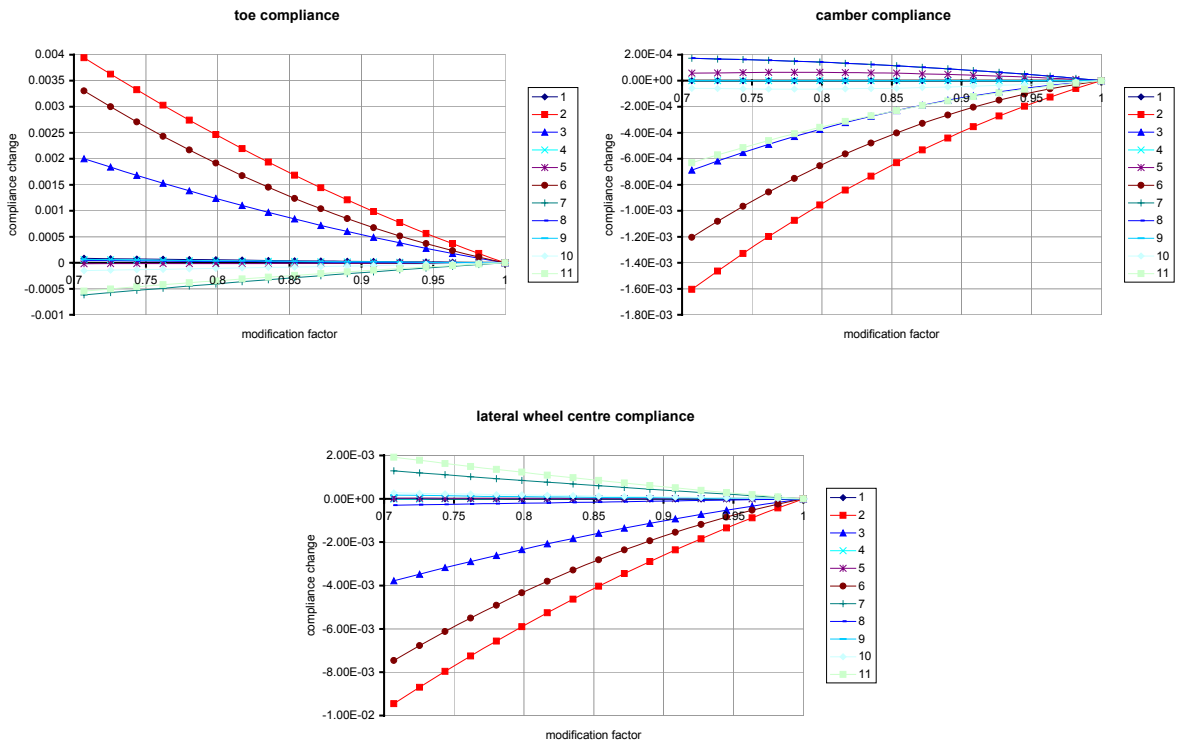


Figure 11.14 – Compliance changes vs. Modification factor, under lateral force

The important DOFs have been labelled with ID numbers from 1 to 11, the first 7 corresponding to the lower control arm and the remaining 4 to the upper control arm. The colours from chart to chart were kept, so it can be appreciated that a few DOFs, even among the important ones, are certainly dominant, such as the number 2, 3, 6 and 11. **Figure 11.15** shows both arms plus the wheelcarrier, and these DOFs represent:

- ID 2 = DOF 1,2: lower control arm, inner front hardpoint, displacement in *y* direction.
- ID 3 = DOF 1,8: lower control arm, inner rear hardpoint, displacement in *y* direction.

- ID 4 = DOF 1,14: lower control arm, outer hardpoint, displacement in *y* direction.
- ID 11 = DOF 2,14: upper control arm, outer hardpoint, displacement in *y* direction.

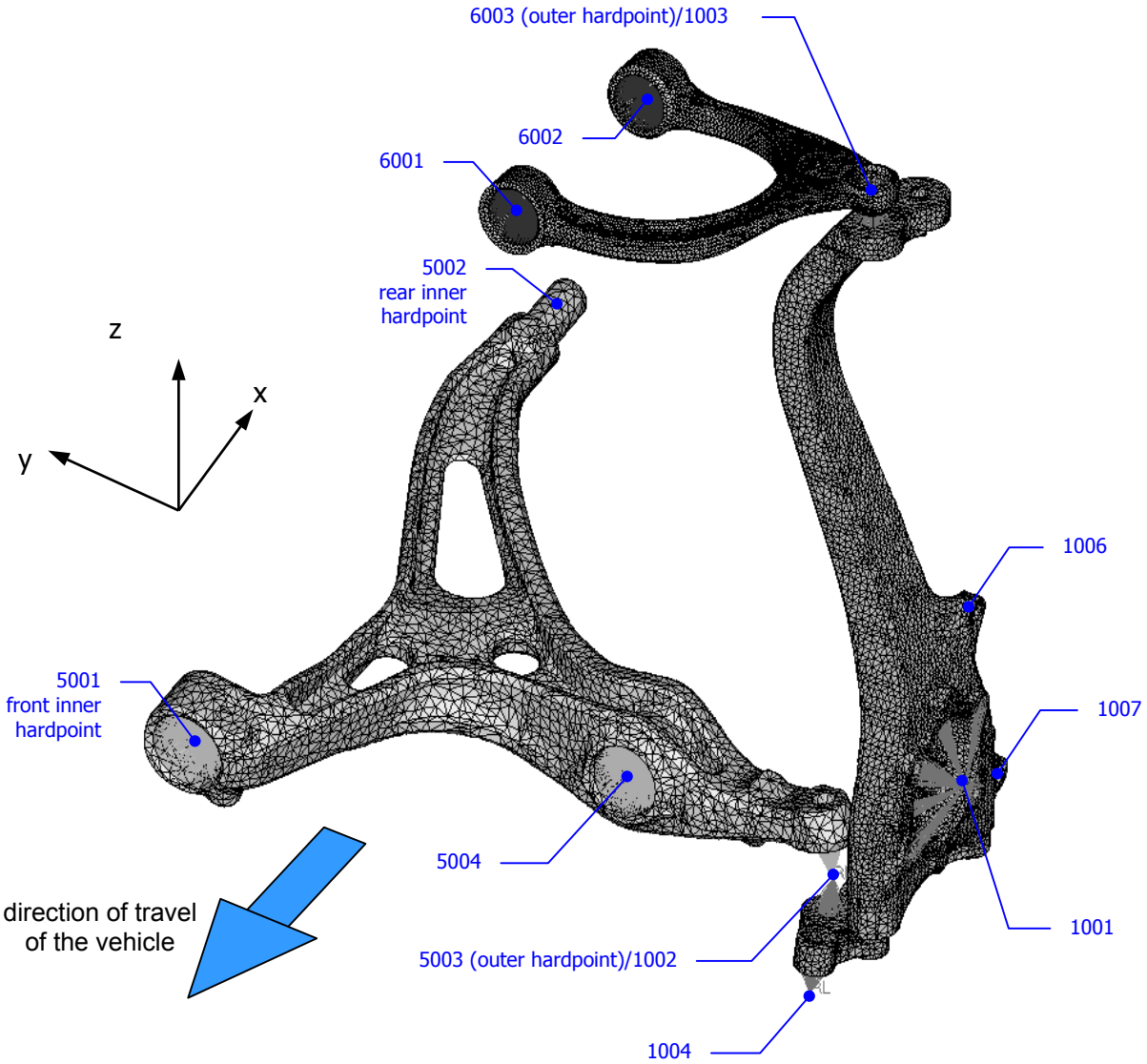


Figure 11.15 – Suspension assembly with DOF indicated

11.6 Optimisation

The solution for the optimisation problem consists of a vector containing the values for the 11 modification factors that maximises the fitness function. This function is inversely proportional to the average of the modification factors (the weight of the components) and the error with respect to the reference compliances. In order to find the vector solution a genetic algorithm is implemented as described in Chapter 5. The main features of the mentioned algorithm are described along with the optimisation

case. In principle, a basic genetic algorithm was implemented with three basic operators: combination, crossover, mutation. The parameters that govern the algorithm were determined after hundreds of runs in which the changes over the results were inspected and compared, such as population size, selection method, crossover method, mutation rate, selection pressure, population management, fitness scaling and termination method. In the following sections the research about these parameters is discussed.

11.6.1 Population size and population management

Out of all the parameters that can be altered when implementing a genetic algorithm, the population size has the strongest influence on both the efficiency of the optimisation and the quality of the results. In principle, two things can be said:

- as the size of the population increases it is obvious to expect an increase in the time that the algorithm needs to run a given amount of generations: a higher number of consultations have to be done in order to assess every individual in the population.
- at the same time, to obtain a solution of a given quality (measured by the fitness) a bigger population would mean a smaller number of generations. In a larger population, the chances that a fitter algorithm is generated randomly for the first generation are higher. There are more not-so-fit individuals with potentially high fitness in some portions of their genes that can profitably mate and crossover with other individuals. Besides, larger populations improve diversity, and therefore they cover better the design space, improving the chances of exploring more areas that may contain potentially good solutions.

During the optimisation process the genetic algorithm applies four basic steps: reproduction, crossover, mutation and fitness evaluation. Out of those, the last is the one that takes up most of the time in every generation. Unfortunately, due to the way the computer stores and manipulates the sheer amount of data (the knowledge database), duplicating the amount of individuals per generation increases the time needed for the assessment of the population in a factor higher than 2. On the other hand, a population twice as big does not double the quality of the results, but less.

Figure 11.16 and **11.17** show how duplicating the size of the population indeed improves the quality of the results, at the expense of a steep increase in the computation time, as shown in **Figure 11.18**. The main simulation parameters used for this study are:

- generations: 500,
- selection pressure: 4,
- selection method: roulette wheel selection,
- crossover method: 2 points crossover,

- mutation rate: adaptive,
- fitness scaling: none,
- random immigrants: none.

As can be seen, the maximum fitness achieved by the optimisation process increases with the size of the population. Smaller populations are too unstable and do not propagate good schemas through the generations. Also the diversity is too small to properly cover the design space, and find potentially good areas and exploit them. On the other hand, there is an upper limit for the improvement that is achieved by enlarging the population, since the closer the solution gets to the optimum, the hardest it is for the algorithm to further improve the solution.

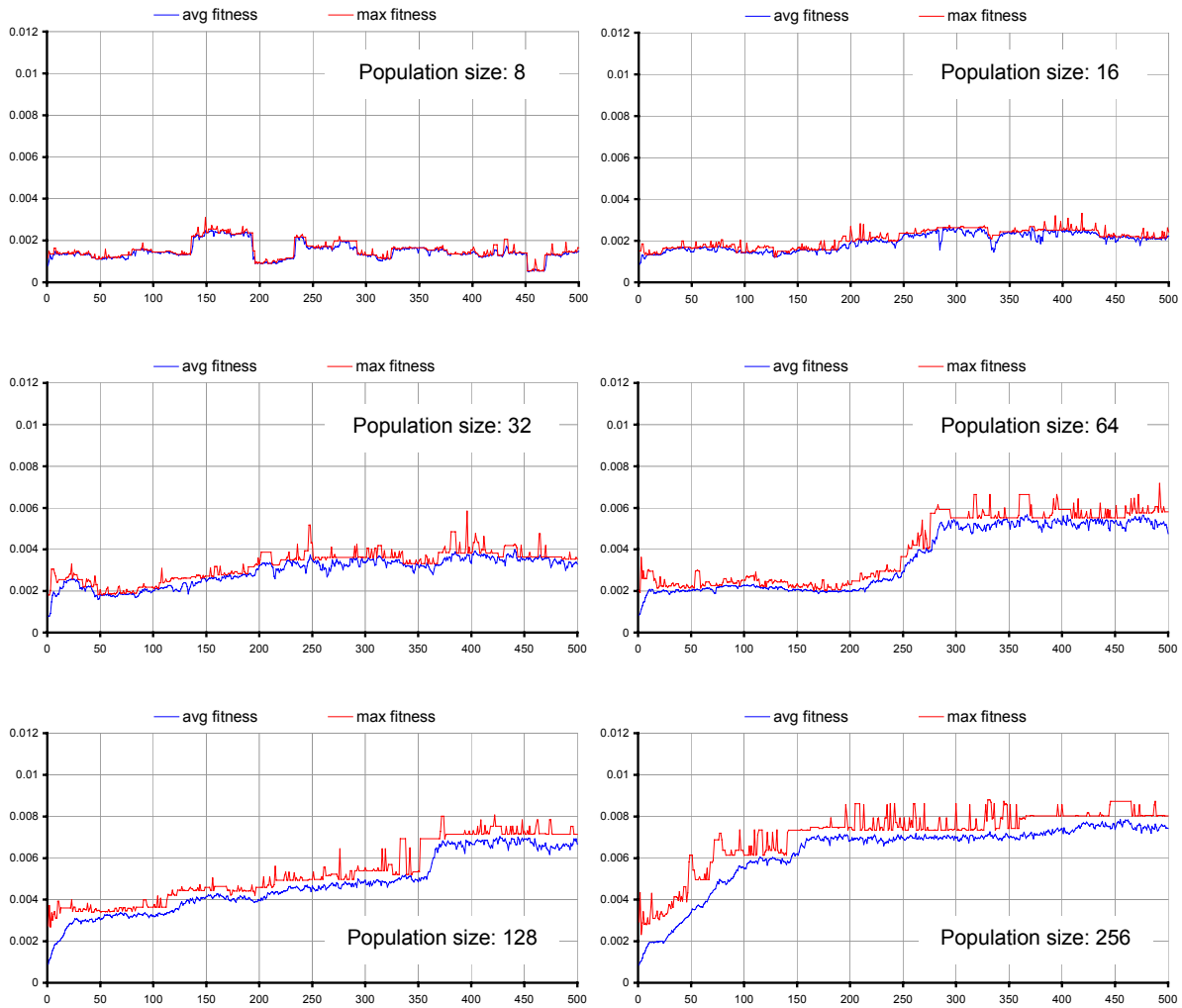


Figure 11.16 – Evolution of the fitness through the generations, for different sizes of population

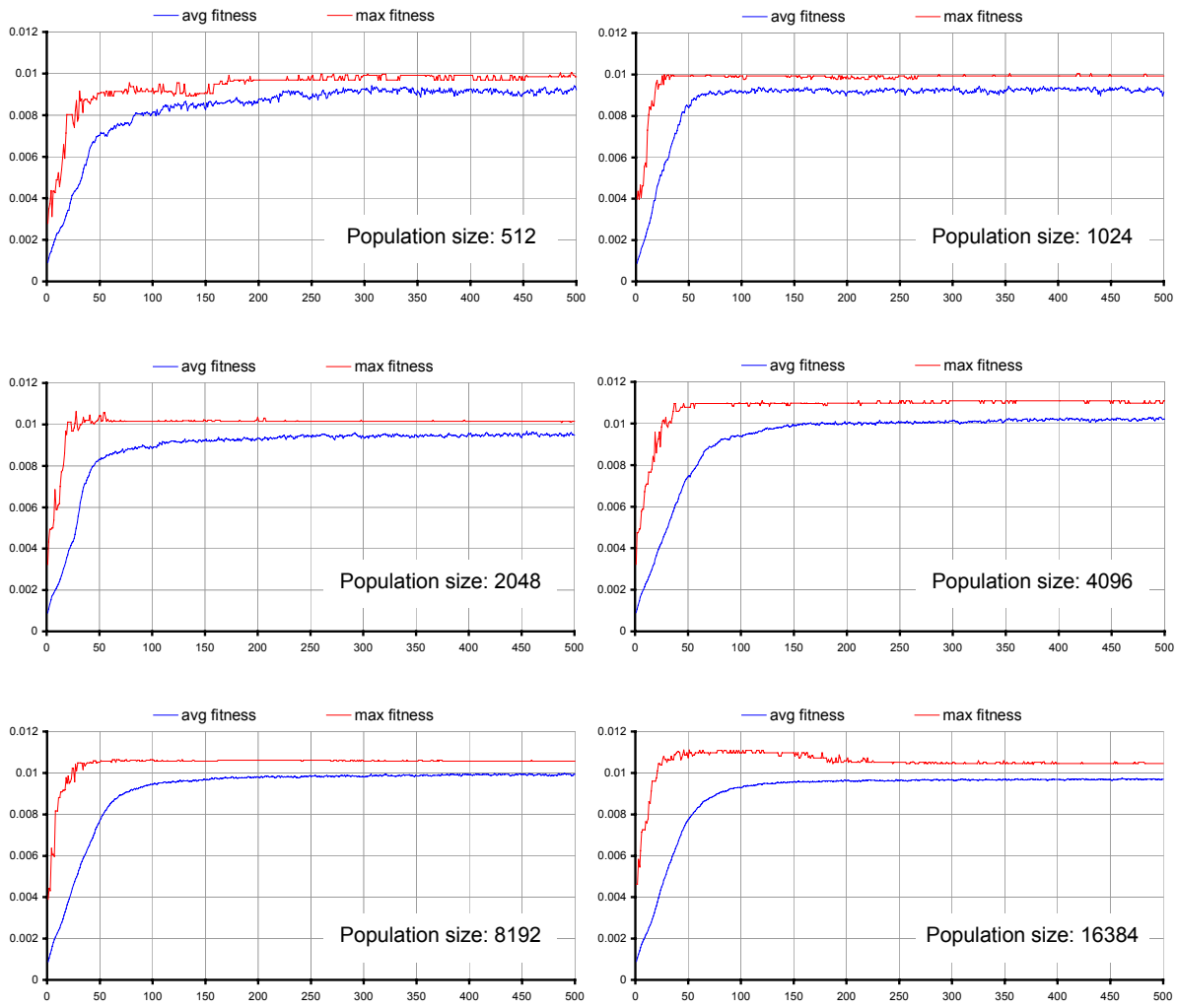


Figure 11.17 – Evolution of the fitness through the generations, for different sizes of population

Figure 11.18 shows the maximum fitness achieved by every population size. When analysing these results, it must be kept in mind that the genetic optimisation involves a great amount of stochastic processes, and therefore the results can vary, that is, they do not determine precisely the boundaries between small and large populations performance. However, it is clear that doubling the population does not double the fitness of the solution, and when the population is very large the extra increase in size is barely reflected on the results.

The exact figures of the improvement are shown in **Table 11.4**., where it can be seen that there exists a sort of limit value that the fitness can reach, around 0.011. With a population of 2048 individuals it is already sufficient to reach such value, and further increase in the population size brings along no advantage. However, the size of the population is influenced by the amount of optimisation parameters since they determine the size of the design space that has to be covered as much as possible with by the first generation.

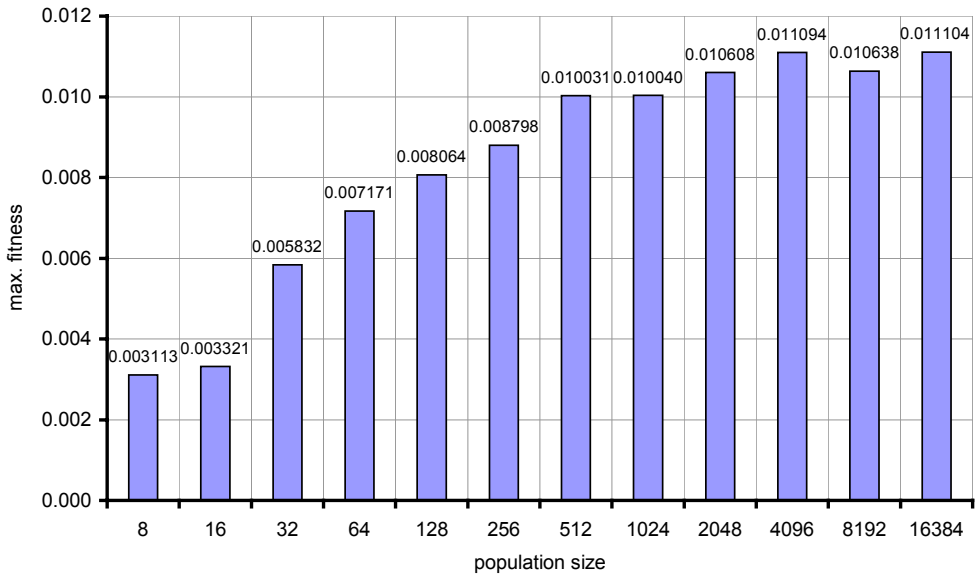


Figure 11.18 – Maximum fitness achieved for different sizes of population

population size	max. fitness	improvement
8	0.003113	-
16	0.003321	6.68%
32	0.005832	75.63%
64	0.007171	22.95%
128	0.008064	12.45%
256	0.008798	9.10%
512	0.010031	14.02%
1024	0.010040	0.09%
2048	0.010608	5.66%
4096	0.011094	4.58%
8192	0.010638	-4.11%
16384	0.011104	4.37%

Table 11.4 – Maximum Fitness for different sizes of population

Concerning the computational time, **Figure 11.19** shows the amount of time needed to run the 12 aforementioned optimisation cases and the increase in time with respect to the previous size of population. For example, if the population size is 16, doubling the population means an increment of only 19.51% in time. However, for a population size of 128, doubling the population size means 61.96% more time, and the larger the population, the larger the factor that multiplies the time requirements. When the population size is about 4000 individuals or more, doubling the size of the population tends to more than double the time used for the algorithm to run a given amount of generations. Therefore, it

is obvious that for small populations other processes have some weight on the total time, but as soon as the population size increases the calculation of the fitness becomes dominant, together with the problems to manipulate such amount of data by the computer.

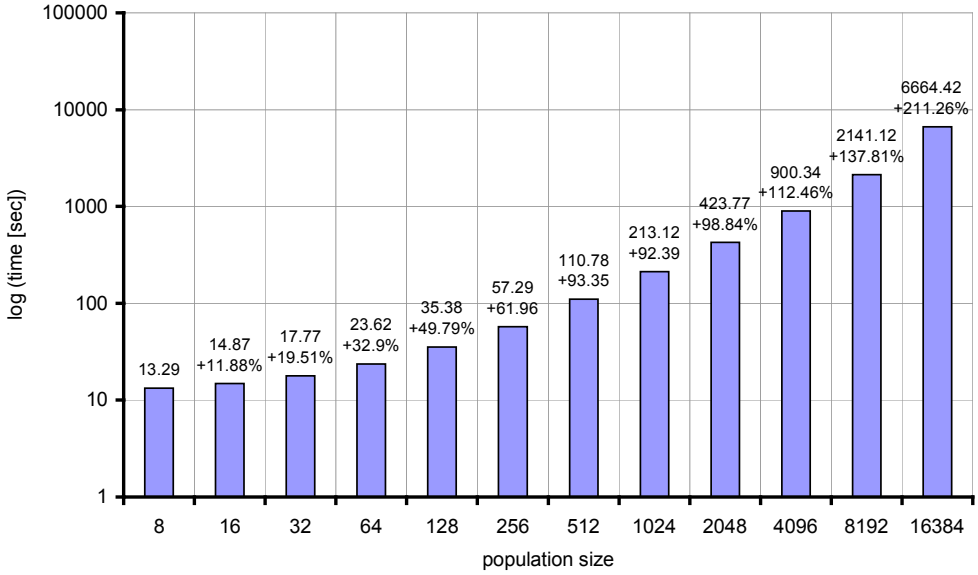


Figure 11.19 – Computation time of the optimisation process for different sizes of population

From **Figures 11.16** and **11.17** it can be seen that the highest results are achieved by the larger populations well at the beginning of the optimisation process. In particular, a population of 1024 or 2048 individuals achieved their best result in the first 100 generations. Obviously the diversity provided by large populations, which allows locating good areas in the design space, is more important than the evolution that can be done from less fit individuals. As tempting as it looks, it is not a good idea to terminate the process at this point since some more mutation could lead still to an improvement in the results. Unfortunately this means that the process needs a long time to be completed. Nevertheless, for populations up to 4000 individuals, doubling the population does not double the time, so a process with twice the population and half the generations would seem to lead to a better efficiency. Unfortunately, as **Figure 11.20** shows, the result of such procedure (where for a population size of 8 the number of generations is 500, for population size of 32 is 250, for population size of 64 is 125 and so forth) is that the fitness attained by a population of 8 individuals in 500 generations is about the same as what it was attained by a population of 4096 individuals in 1 generation. In fact the result does not vary too much from what it can be obtained by a random individual, as it can be seen from the last three populations sizes where only one generation, the first, was produced. In general, the fitness is about one third of what can be achieved after a proper optimisation.

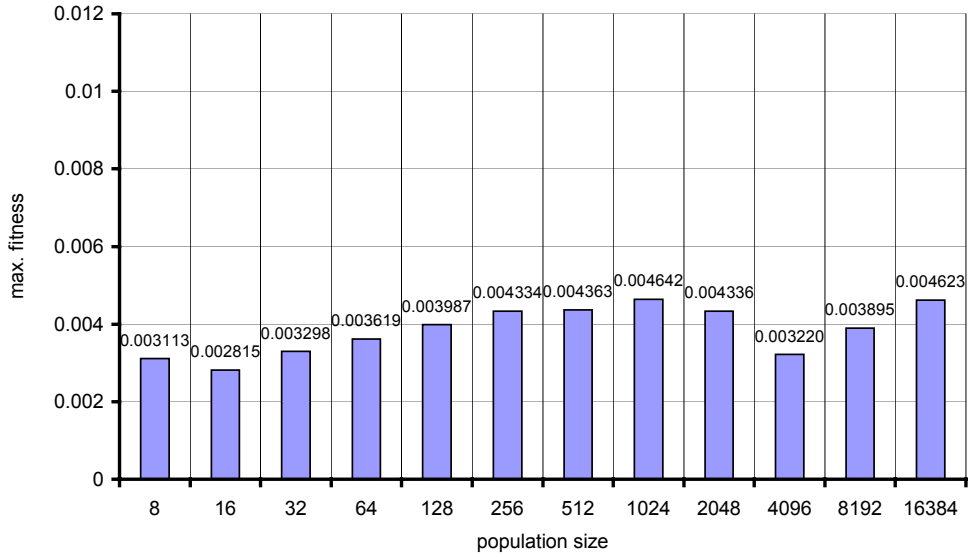


Figure 11.20 – Fitness vs. population size (halving the number of generations)

It can be concluded that the randomly generated individuals must be enough to allow the optimiser to comprehensively explore the search space and pinpoint those areas with more potential for improvement. After this first stage that comprises the first few generations, a large population is no longer needed and adds nothing but computation time. It is necessary then to introduce in the algorithm a parameter that allows the regulation (downsizing) of the population as the generations pass. This can be a way to make use of the diversity of an initially large population, and the short process time of a small population.

As a first trial, an initial population of 2048 individuals and a final population of 16 individuals are chosen, and 8 individuals are eliminated at each generation. The number 8 comes from a demand to the population size, that it must be divisible by the parameter expressing the selection pressure (4), times the amount of parents (2). Therefore, the population decreases linearly until it reaches the lowest limit, in generation 255, from which it remains constant until the last generation.

Figure 11.21 shows the results of a simulation in which the positive effect of the initially large population can be appreciated in the diversity present, which allows the algorithm to effectively locate potentially good areas of the search space and develop them. It is also obvious that reducing the population too much actually downgrades the quality of the solution. The maximum fitness achieved by this process is 0.105731 (very similar to the previous best: 0.011104), and the total time is 119.16 seconds, out of which the first 255 generations take 110.44 seconds.

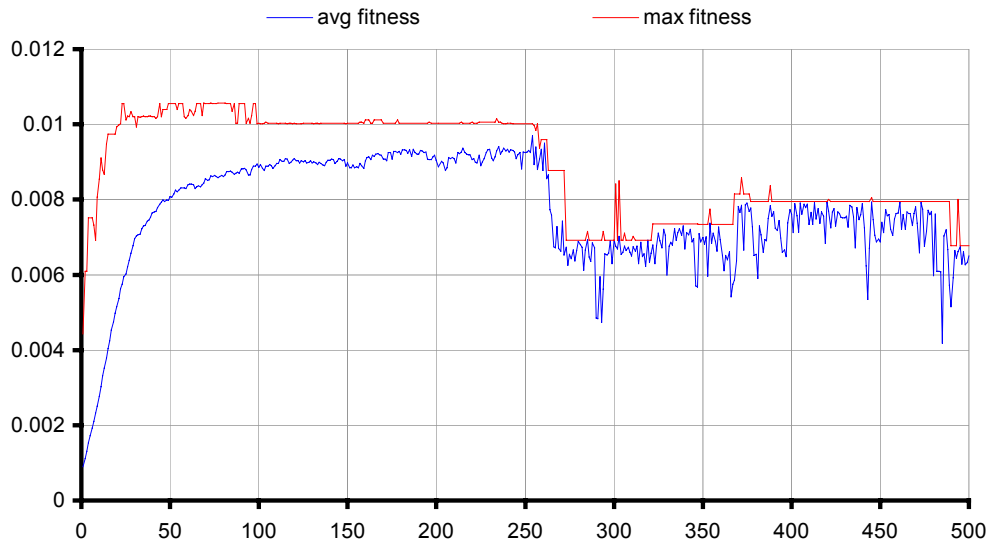


Figure 11.21 – Fitness evolution for decreasing population size (2048→8)

From **Figure 11.21** it can be seen that populations smaller than 128 individuals are too small and unstable to offer good performance for the algorithm, since they limit the ability of the population to propagate the good schemata through the generations. Therefore, 128 will be used as a lower limit for the population size and the upper limit increased to 4096 individuals in order to improve diversity. The number of generations is lowered to 250, which seems to be enough for the average fitness of the population to stabilise. The results of an optimisation process with these parameters can be seen in **Figure 11.22**. The maximum fitness is 0.011218, and the total time is 223.35 seconds.

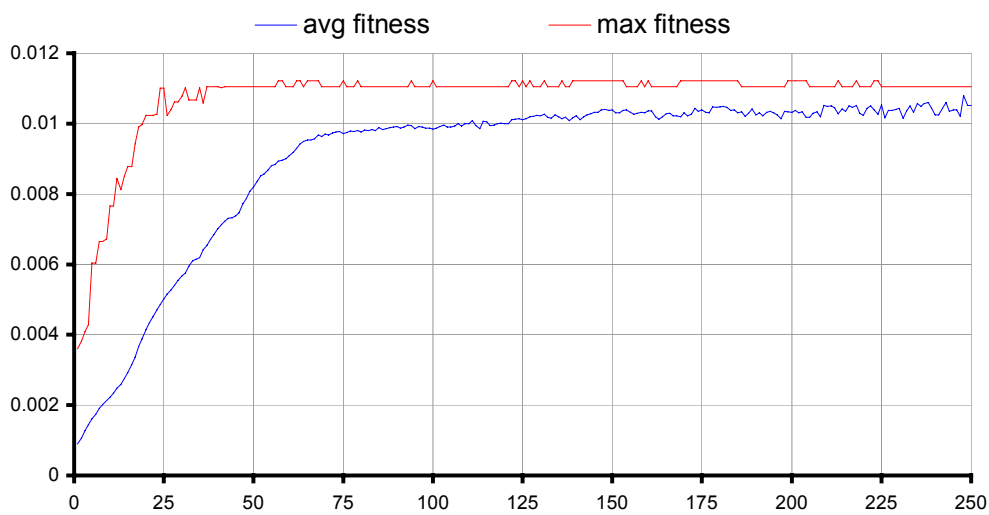


Figure 11.22 – Fitness evolution for decreasing population size (4096→128)

11.6.2 Selection method

As discussed in Section **5.4**, selecting individuals by the tournament selection method involves picking a number of individuals at random from the population to form a "tournament" pool, and then proceed deterministically mating the individuals from this pool. Such method can be useful if the fitness function is very difficult to calculate or time consuming but provided the knowledge database this is not the case, so this method is dismissed in favour of the roulette wheel selection method. What is left to be determined is whether it is worth it or even necessary to implement elitism.

Figure 11.23 depicts four optimisation processes where the population is 256 individuals in two cases, and 2048 in the others. The two charts on the left side do not implement elitism, while the two charts on the right do. In principle, elitism may lead the algorithm to fall into a local solution by focusing on rather fit individuals of the initial generations. It forces the algorithm to keep the best individual from one generation to the next one, which in turn may inhibit slower but perhaps potentially better schemata from propagating.. However, if the population is large enough this is more of an advantage because it helps the algorithm to progress without being misled by the stochastic side of the process that sometimes eliminates good solutions.

The time needed for the optimisation process, it is slightly increased in both cases from 57.27 to 59.47 seconds (3.79%) for a population of 256 individuals and from 423.77 to 425.54 seconds (0.42%) for 2048 individuals.

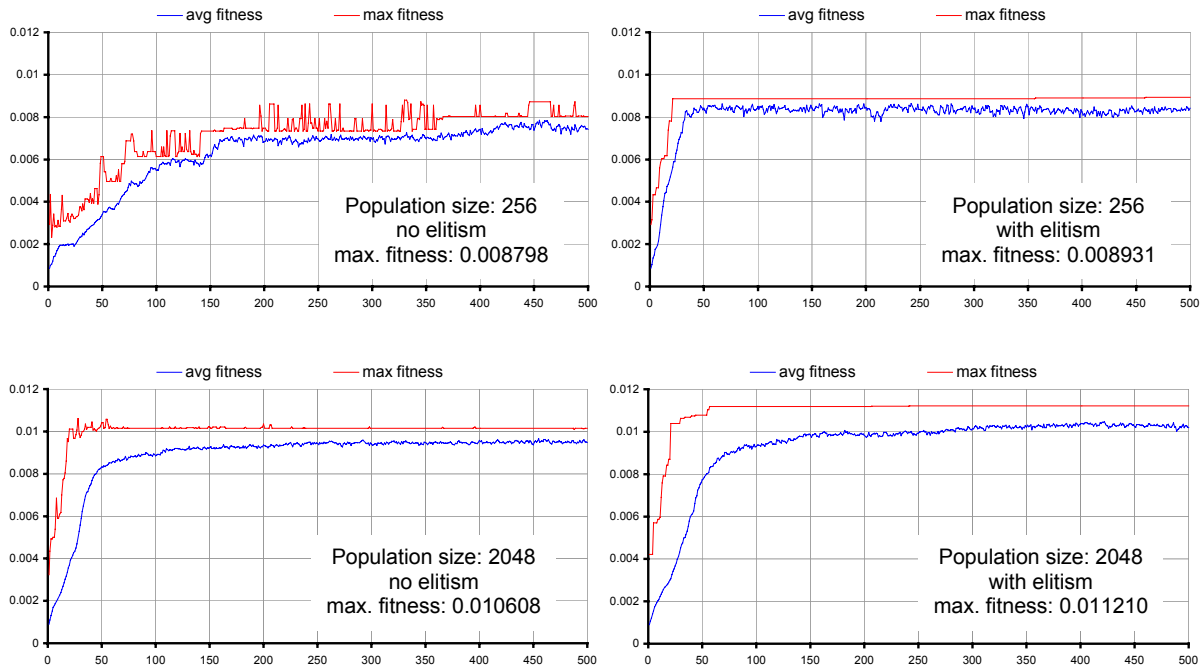


Figure 11.23 – Effect of elitism on the fitness

11.6.3 Crossover method

According to the details presented in Section **5.5**, out of the four methods presented two of them can be immediately dismissed: one-point crossover and uniform crossover, since both have the inconvenience that they tend to destroy schemata. Half-uniform crossover is extremely useful only when the coding of the genes is binary, otherwise the Hamming distance partly looses its meaning. In a non-binary encoding, especially in the first generations, the Hamming distance tends to be as high as the length of the gene string. Furthermore, on the last generations the Hamming distance is usually 0 or 1 since the individuals are very much alike, so no improvement is accomplished except for those changes introduced by the mutation operator.

In conclusion, the best choice is the two points crossover which is suitable for the codification method selected where each gene simply represents one of each of the 17 possible values between the lower limit for the modification factor, and 1 (the number 17 comes from the choice of the user of dividing the search space in 16 steps, as explained in Section **11.5**). For example, if the lower limit is $\frac{\sqrt{2}}{2}$, dividing the interval $\left(\frac{\sqrt{2}}{2};1\right)$ in 16 steps means that each gene can take a value according to the equation:

$$\text{gene} = \frac{\sqrt{2}}{2} + (n-1)\cdot\frac{1-\frac{\sqrt{2}}{2}}{16} \tag{11.1}$$

where n is any integer from 1 to 17. In this way, if the important DOFs are 11, as it resulted from the sensitivity analysis, the gene string contains 11 genes. For ease of access, storage and use (as index to consult the knowledge database) the values in the genes are not the modification factors themselves but the values of n. For example, one individual could be:

1 13 17 3 15 12 15 17 17 14 4 1

11.6.4 Mutation rate

In order to determine the best mutation rate, a rather small population (just 256 individuals) will be used for the study. The reason is that large populations do cover the search space in a comprehensive way and generally provide good results, but at the cost of increasing the time required for the process. The objective is to determine the adequate mutation rate that achieves the same results starting off from a smaller population, faster to handle by the computer.

Figure 11.24 depicts the importance of the mutation operator in three simulations. It shows the evolution of the maximum fitness for three populations when the mutation operator is not applied. The result of the process is completely dependent on the available pool of genes generated randomly for the

first generation. Even worse, the algorithm rapidly falls into a local solution and does not progress any further. Once the reproduction and crossover operators have been applied, the population consist of only one genetic code (the average and maximum fitness are the same) after a few generations, and the stochastic selection of the parents pool does eventually let the slightly fitter individuals die out.

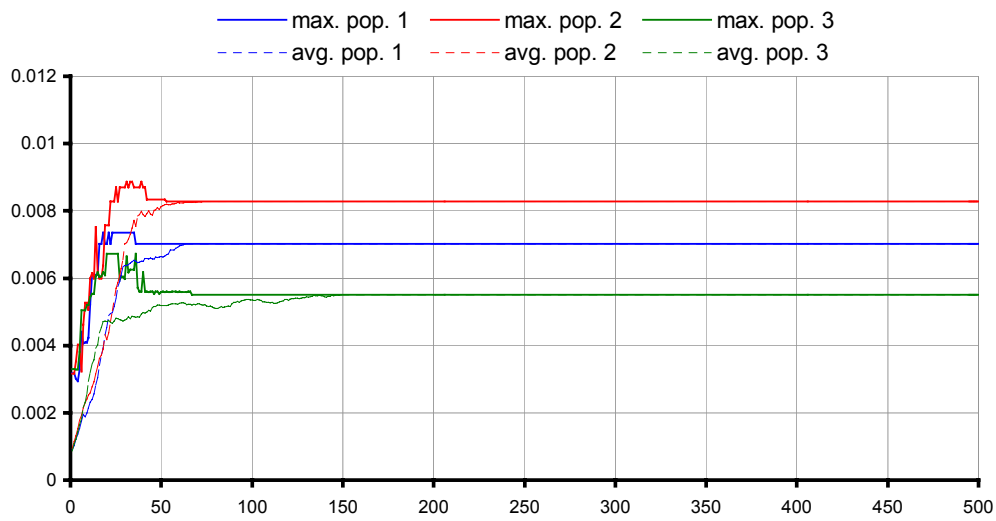


Figure 11.24 – Evolution of the fitness without mutation

Table 11.5 shows the result of 8 optimisation runs, with a population of 256 individuals. The first case does not use any mutation operator, so the algorithm must find the best individual exclusively by re-combination of the available pool of genes randomly generated for the first generation. In the other 7 cases the mutation rate is varied from 0.0001, 0.001, 0.01, 0.1, 0.2, 0.4 and 0.8.

mutation rate	max. fitness	improvement
0	0.00887012	
0.0001	0.00863105	-2.70%
0.001	0.00898068	4.05%
0.01	0.00978691	8.98%
0.1	0.01027875	5.03%
0.2	0.01032458	0.45%
0.4	0.00913058	-11.56%
0.8	0.00674379	-26.14%

Table 11.5 – Maximum fitness for different fixed mutation rates

It can be seen that introducing the mutation operator increases the fitness that the algorithm is able to reach. The results suggest that if the mutation rate is too low, there is not enough influence of this op

erator to allow any improvement. As the mutation rate increases, the algorithm is able to find solutions not present in the original pool of genes and to increase the fitness of the population up to a mutation rate of 0.1. Beyond that the rate is so high that it actually dominates the process, masking the improvements achieved by the reproduction and crossover operators: good individuals among the offspring are being mutated and therefore degraded, which lowers the quality of the population. Consequently, very few or none individuals are fit enough to become parents for the next generations, stopping the progress of the population. In any case, the mutation operator modifies the offspring population and not the parents, leaving these as fit as before. If a mutation does nothing but degrade an individual, the original parents that gave birth to such individual are still there in the gene pool to be mated once again in later generations.

This effect is also visible for large populations. **Figure 11.25** depicts the fitness progression of two populations of 2048 individuals. It is evident how by increasing the mutation rate, the average fitness of the population is lowered and the progress of the maximum fitness becomes slower. Horizontal segments in the red curves show that the pool of parents, which presumably contains the fittest individuals, does not change through the generations. It is important to notice that in both cases, however, the maximum fitness achieved by the algorithm is higher than those achieved in **Table 11.5**, due to the higher diversity provided by the larger populations.

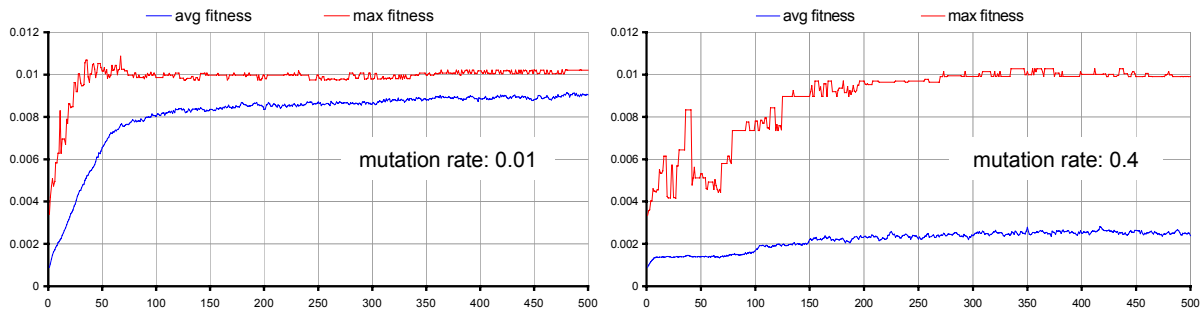


Figure 11.25 – Effect of mutation rate on the fitness

In order to improve the algorithm an adaptive mutation rate is implemented according to the equation presented in Paragraph **5.8.4**. The adaptability is implemented by adding a factor that is the ratio between mean and maximum fitness, to the power of the geometry factor n. The geometry factor n takes the value 4 since lower values have no effect, while higher values lead to a mutation rate too high that turns the genetic algorithm into a random walk over the search space. The proportionality constant k is varied in order to determine the best value for this parameter.

In **Table 11.6** it can be seen the results of implementing such adaptive mutation rate. As expected, increasing the mutation rate (which is increased with the parameter k) improves the maximum fitness

attained by the algorithm. Since the factor added to adapt the mutation rate is always smaller than 1 (typically it takes values between 0.1 and 0.7), the resulting mutation rate is slightly lower than the one without this factor. However, it is a variable value that fluctuates from generation to generation, adapting to the changing conditions of the population according to the mean and maximum value of the fitness. As a result (**Figure 11.26**) the mean fitness is significantly higher with respect to the maximum fitness when implementing the adaptive mutation rate, which does not mutate blindly in order to improve diversity. In turn, this means that the available pool of genes is of "better quality", facilitating the algorithm the improvement of the population. Therefore, the results obtained by implementing an adaptive mutation rate are superior. From **Table 11.4** and **11.5** it can be seen that the best fitness achieved by implementing an adaptive mutation rate (0.01103462) is 6.88% higher than that with fixed mutation rate (0.01032458).

k	max. fitness	improvement
0	0.0088701	
0.0001	0.0090202	1.69%
0.001	0.0090617	0.46%
0.01	0.00979902	8.14%
0.1	0.01058809	8.05%
0.2	0.01100632	3.95%
0.4	0.01103462	0.26%
0.8	0.01029899	-6.67%

Table 11.6 – Maximum fitness for different adaptive mutation rates

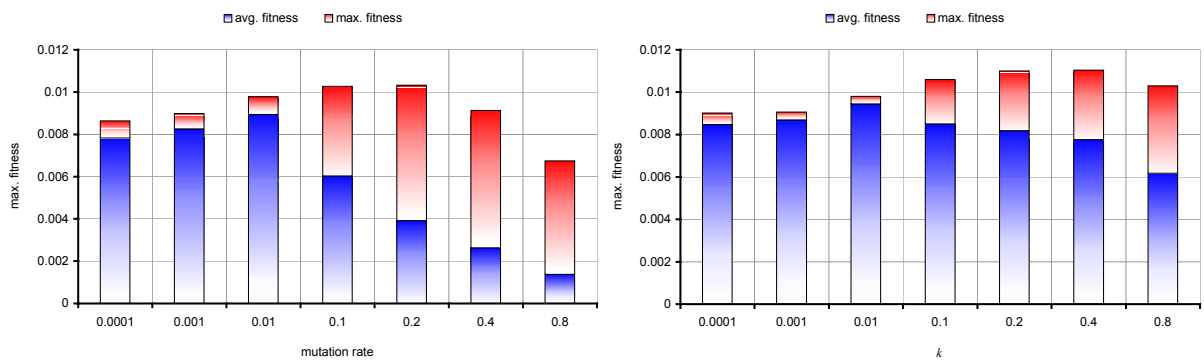


Figure 11.26 – Effect of fixed and adaptive mutation rate

11.6.5 Selection pressure

By changing the parameter that defines the selection pressure, the ratio of the size of the population with respect to the parents that survive from generation to generation is set. The higher this parameter,

the tougher it gets for the individuals to survive for the next generations, since the pool of genes that are selected for reproduction is smaller. On the other hand, those individuals that make it to the next generation are aimed to produce more offspring.

For the research to determine the best value of selection pressure, a small population size of 240 individuals is used. **Figure 11.27** shows the results of changing the selection pressure from 2 to 8, where the worst (blue bars) and best (red bars) results of 5 trials for each value of selection pressure are shown. The chart shows that increasing the selection pressure improves the results since it forces the struggle within the population to survive to the next generations. It can be noticed that, in general, the higher the selection pressure the bigger is the difference between the best and the worst result. A selection pressure of 4 delivers the same results as higher values but with lower dispersion, plus taking less time for the process since a lower selection pressure means less new individuals to generate for each generations and less new fitness values to determine.

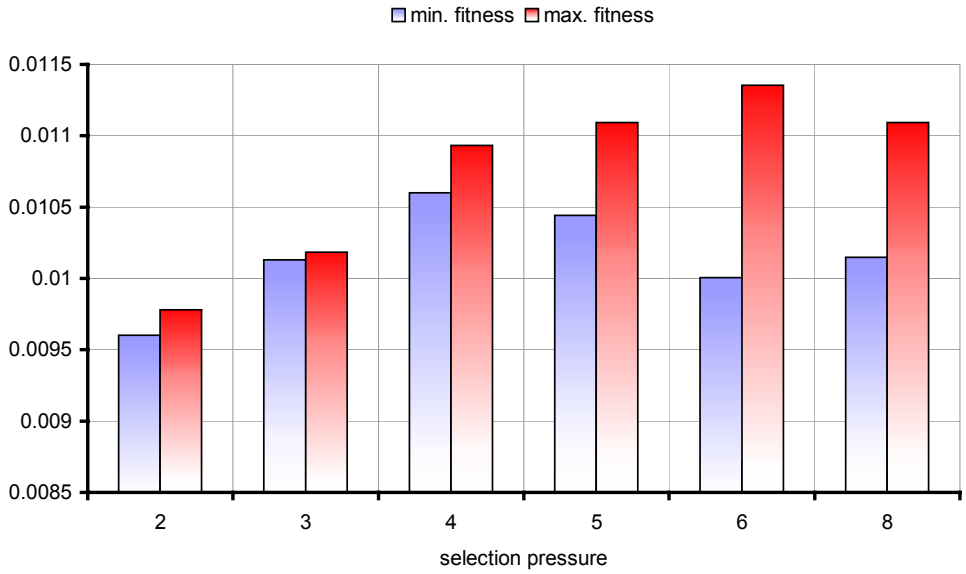


Figure 11.27 – Effect of selection pressure on the fitness

11.6.6 Fitness scaling

Scaling the fitness of the population provides a method to stop the algorithm from falling in a local solution by evening the differences at the beginning of the process, avoiding stagnation at the end of by enlarging differences, or both. **Figure 11.28** shows a reference run with no scaling (left) and one where the ranking method is applied (right). As can be seen, the ranking method is unsuccessful. The reason for such a low result can be found in **Figure 11.29**, where the relationship between the scaled and non-scaled fitness of the population at the 10th and at the 250th generation is depicted.

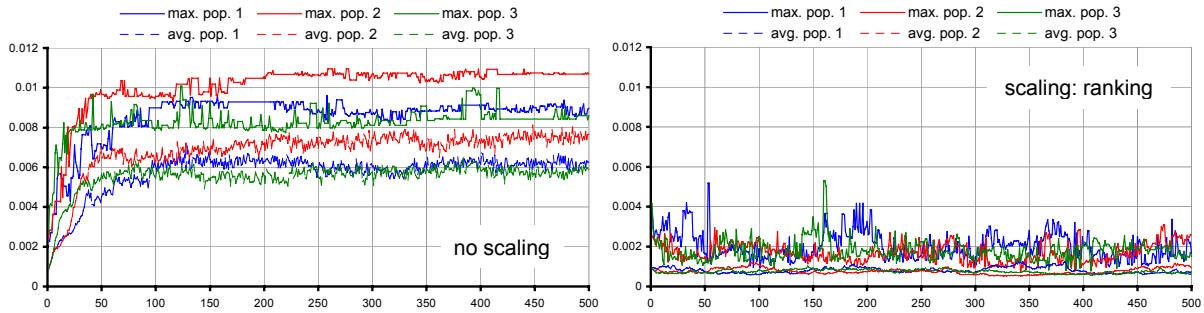


Figure 11.28 – Scaling by the ranking method

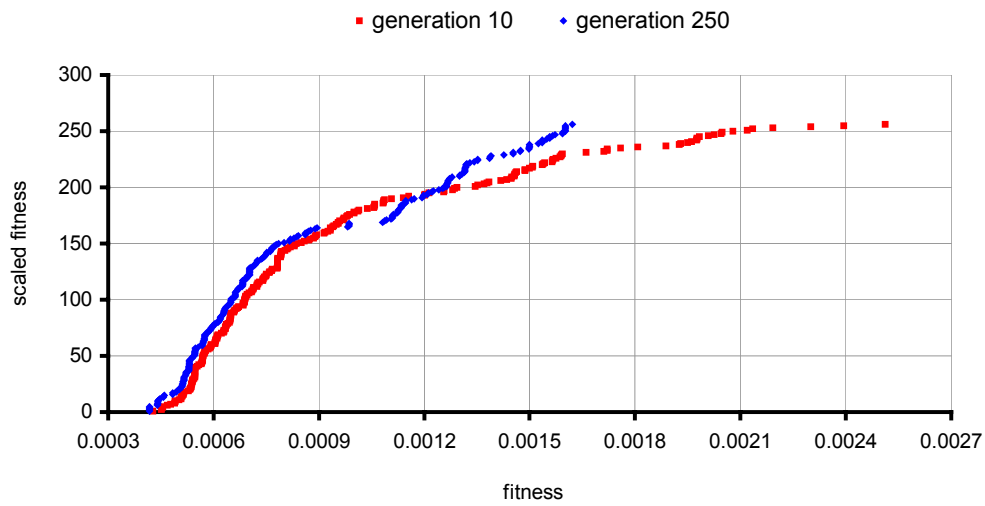


Figure 11.29 – Effect of the ranking scaling on the fitness across the generations

As expected the population at the beginning of the process has a low fitness, with most of its individuals in the lower part of the spectrum but with a few with fairly above-average fitness, up to 0.002512. When the scaling method is applied those few fit individuals are evened out with the rest of the population, making the algorithm harder to pick them for reproduction. For instance, the average of the non-scaled population is 0.000951, so the ratio of the maximum to the mean fitness is 2.64, while for the scaled fitness this ratio is only 2. The ranking method is suited for cases in which there are hyper-fit individuals at the early populations (with a ratio of maximum to mean fitness of orders of magnitude higher than here). Therefore, the fairly good individuals are not so good after being scaled, plus they are just a few, lowering their chances to survive and dying out. As it can be seen in the plot for generation 250, the population is rather similar to the on at the 10th generation with two differences: the fittest individuals are extinguished and the population is more homogeneously distributed. As expected the scaling method is preventing the algorithm from falling into a local solution, but unfortu

nately it is preventing it from falling into any solution at all, since the randomly generated population does not present hyper-fit individuals in a degree that need to be evened. The values are rather similar and they do not need to be evened.

In **Figure 11.30** it can be seen the effects of implementing exponential scaling. According to the equation presented in Paragraph **5.8.2**, k is a problem-dependent factor that outlines a homogeneous population. When $k = 1$, an individual with fitness higher than the mean fitness is considered hyper-fit, hence evened in some degree. The higher the value of k, the higher has to be the ratio between the individual's fitness and the population's mean fitness to be considered hyper-fit. Conversely, when the best individual has a fitness value lower than k times the mean fitness of that generation, this population is considered homogeneous and therefore its fitness scaled in order to increase the differences.

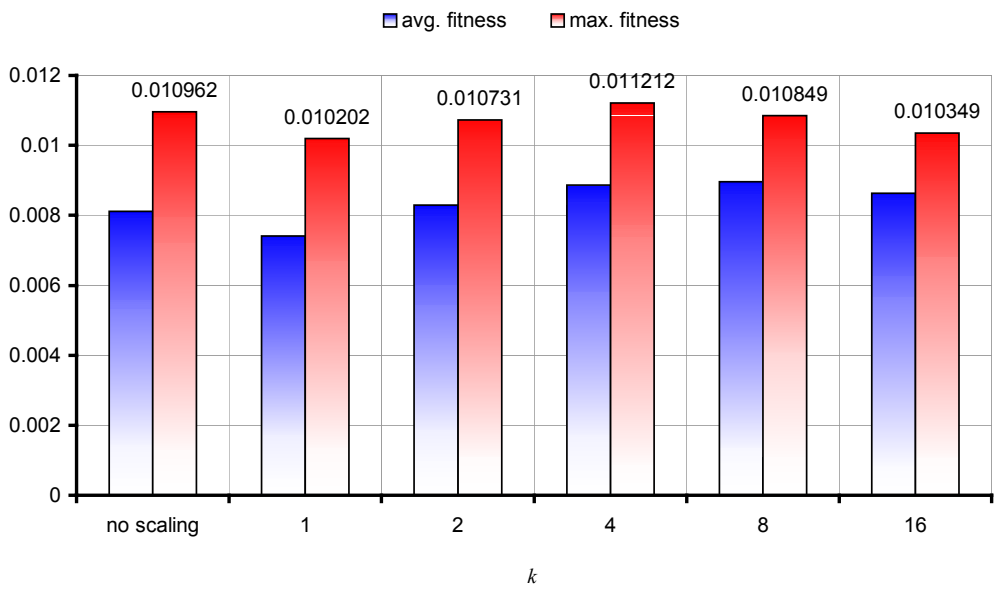


Figure 11.30 – Effect of the factor k on the results of the exponential scaling

As it can be seen from the chart, a value of $k = 4$ delivers the best performance, achieving better results than those without scaling. Recalling the ratio of the first generations measured when applying the ranking method, the usual value of the ratio between maximum and mean fitness is less than 3. Therefore, by setting $k = 4$, all the individuals with fitness higher than the mean fitness of the population, but still not 4 times higher, will be scaled up, enhancing their chances to be selected for reproduction. Setting k too high lowers the degree of scaling and the quality of the results starts declining.

The difference between the effect of both scaling methods can be seen by comparing the shape of the curves from **Figure 11.29** and **Figure 11.31**. On the latter one, the values above the mean fitness are scaled up and those below it are scaled down, while on the ranking method it is the opposite.

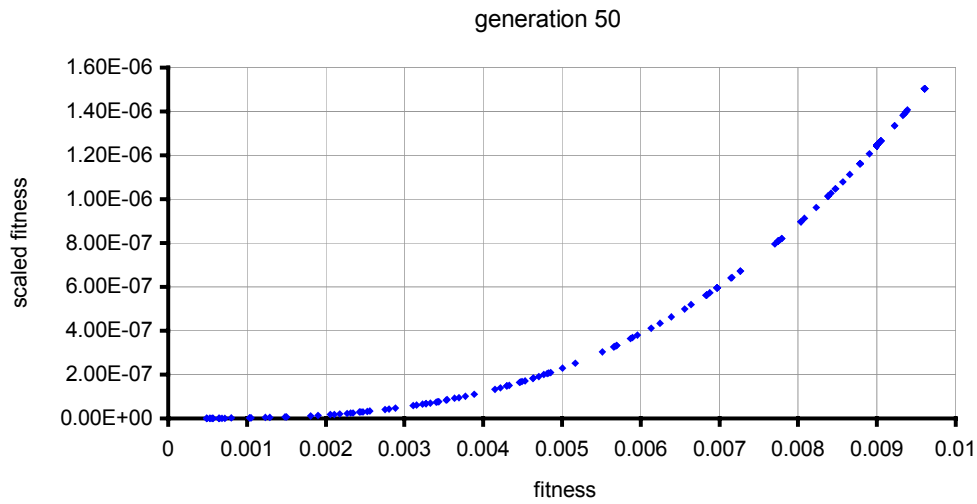


Figure 11.31 – Effect of the exponential scaling on the fitness across a generation

11.6.7 Termination method

The termination method selected is by the number of generations: 500. During the whole process the maximum fitness is tracked and the best individual stored, so if the maximum fitness of the last generation is lower than that of a previous one, the best ever is still the one reported.

11.6.8 Conclusions

As a result of the quest for the best methods and the adequate values for each of the parameters involved, the final shape of the genetic algorithm is as follows:

- generations: 500,
- population: variable (initial population: 2048; final population: 256),
- selection pressure: 4,
- selection method: roulette wheel selection with elitism,
- crossover method: two-points crossover,
- mutation rate: adaptive ($k = 0.5$),
- fitness scaling: exponential ($k = 4$),
- random immigrants: none.

The introduction of random immigrants was briefly considered, but no perceptible difference was found. The mutation rate is high with respect to the usual values suggested by literature **[4.1] [4.4] [4.5] [5.3] [5.4] [5.6]**. The relatively large population generated to start off the process provides the necessary diversity in the first generation to allow the algorithm to effectively scatter in the search space. The decrease of the population from generation to generation is managed by the decrement of the number of individuals (parents) appointed for reproduction. The decrement step is calculated as the size of the interval divided by the number of steps, $\frac{P_0 - P_f}{n_{gen}}$, where

P_0 = initial population

P_f = final population

n_{gen} = number of generations to run the process

The outcome number must be checked to be a multiple of twice the selection pressure, so the available pool of parents is always a number that allows the chosen selection pressure without inconsistencies for the algorithm.

The result of all these choices can be seen in **Figure 11.32** where three runs are depicted. The first point that is evident is the consistency of the results, that is the algorithm is repetitively able to find at least a very good solution, compared with all the previous results. Nevertheless, in any serious optimisation process, and especially due to the stochastic nature of the genetic algorithm, a key factor to achieve the best results is to make several runs of the algorithm, and select the genetic code that delivers the highest fitness. Each run takes approximately 180 seconds.

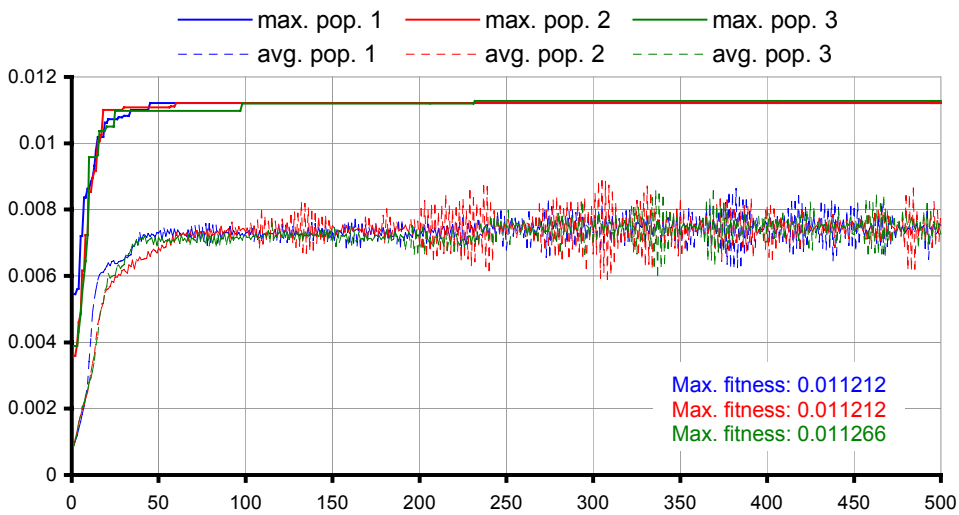


Figure 11.32 – Resulting optimisation runs

It is important to note that, since the fitness function is inversely proportional to the error of the measurements, the targets set for the optimisation process cannot be the reference values, but they must be varied in a small value in order to avoid the division by zero when the algorithm assesses the fitness of the individuals. The targets for this case were the reference values plus 0.5%.

11.7 Static analysis

As a result of the optimisation process run over the suspension assembly, the vector containing the 11 modification factors is:

$$\underbrace{0.707107,\ 0.890165,\ 0.981694,\ 0.725413,\ 0.963388,\ 0.963388,\ 0.945083}_{\substack{\text{lower control arm} \\ \text{(DOF: 1, 2, 8, 9, 13, 14 and 15)}}},\ \underbrace{1,\ 1,\ 1,\ 0.743718}_{\substack{\text{upper control arm} \\ \text{(DOF: 1, 7, 8 and 14)}}}$$

The correspondence of the modification factors with respect to the components and their DOF is indicated above. All the other DOF are multiplied by the minimum modification factor allowed, set by the user to 0.5. When this set of values is used to modify the stiffness matrices, the optimised components are used to run the simulations in ADAMS that allow determining the new values for the seven measurements. The results are shown in **Table 11.7**.

reference [°/kN]	0.2955	-2.4818	
	-0.1979	2.2768	
	0.1583	-0.1941	-0.2231
optimised [°/kN]	0.2970	-2.4960	
	-0.1990	2.2835	
	0.1592	-0.1953	-0.2241
variation	0.48%	0.57%	
	0.56%	**0.30%**	
	0.58%	**0.62%**	0.48%

Table 11.7 – Actual variation of the measurements after the optimisation process

As it was stated above, the target variation for the measurements is 0.5%. Since this target is set only having in account the particularity of the optimisation algorithm, and not based on any need, the maximum variation of 0.62% is of no concern. The reason for this discrepancy is that the optimisation process is based on the data stored in the knowledge database, which is built up under the assumption

of no epistasis. Internally, the algorithm uses the values taken from the database to estimate the performance of every individual in the population. In **Table 11.8** it can be seen the difference with the estimated value of the solution.

reference [°/kN]	0.2955	-2.4818	
	-0.1979	2.2768	
	0.1583	-0.1941	-0.2231
optimised estimated [°/kN]	0.2970	-2.4923	
	-0.1990	2.2876	
	0.1594	-0.1951	-0.2248
variation	0.48%	0.42%	
	0.54%	0.48%	
	0.70%	0.50%	0.78%

Table 11.8 – Estimated variation of the measurements after the optimisation process

Once the measurements are checked to be within the boundaries of what is acceptable (this depends on the particular suspension assembly and it must be revised by the user) the optimised components are statically analysed. Each component is subjected to the load cases described in Sections **9.1** and **9.2**. The results of the static analysis are given in **Table 11.9**.

	lower control arm (hardpoint 5003)			**upper control arm (hardpoint 6003)**	
	Δx [mm]	**Δy [mm]**	**Δz [mm]**	**Δx [mm]**	**Δy [mm]**
reference	0.0989	0.0254	0.0887	1.1748	0.0605
optimised	0.1066	0.0274	0.0993	2.3496	0.1093
variation	+7.74%	+7.75%	+11.96%	+100.00%	+80.79%

Table 11.9 – Results of the static analysis on the reference and optimised component

The table shows the displacement of the outer hardpoint of each component. It is important that the variation on the targets is positive, that is, the component can only get weaker, not stronger, because the latter would mean to add mass, which cannot be due to the characteristics of the software used in a later stage to perform the optimisation of the components based on the new targets. The old targets, used to design the components in the first place are the reference values for the displacements, while the new targets are the optimised ones. This values are passed on to the next stage called topology optimisation, which allow to go on the CAD file of the original component and, after an automated process, indicate where the mass can be taken out to achieve the new targets.

From **Table 11.9** it can be seen that, regardless the final value obtained for the fitness function by the genetic algorithm, the only numbers that really account in order to take weight out of the components are those representing the variation of the stiffness, that is, the increase in the displacements when applying the different load cases for the static analysis.

Unfortunately, there is not a univocal correlation between the potentially saved weight and the value of the fitness function (which takes into account both the weight of the components and the performance of the suspension), since individuals with lower fitness might in fact be lighter, but nevertheless perform worse for the performance of the suspension. The weight saving is neither univocally rrelated with the objective function (which accounts only for the weight of the components) since an individual might have a low value of the objective function but it achieves this by applying the small modification factors to those DOF that do not play an important roll in the static analyses performed later on, that is, the component "seems" lighter to the objective function, but this is not reflected in the static analysis. For example, the genetic algorithm presents a solution where all the modification factors but one take the lowest value, while the remaining modification factor takes the upper bound value (1), and the DOF affected by this modification factor is the one that rules the results in the static analysis. To the objective function, this individual looks the lightest, but in the static analysis there is no change in the targets that in turn are to be passed on to the topology optimisation process. As an example, **Table 11.10** shows the result of 5 optimisation processes.

	fitness	**lower control arm**			**upper control arm**	
		Δx [mm]	**Δy [mm]**	**Δz [mm]**	**Δx [mm]**	**Δy [mm]**
reference		**0.098934**	**0.025421**	**0.088671**	**1.174782**	**0.060470**
case_1	0.010874	0.115185 16.43%	0.030801 21.17%	0.095539 7.75%	2.349564 100.00%	0.109325 80.79%
case_2	0.010159	0.102658 3.76%	0.029596 16.43%	0.095539 7.75%	2.349564 100.00%	0.109325 80.79%
case_3	0.009943	0.110766 11.96%	0.034892 37.26%	0.099276 11.96%	2.349564 100.00%	0.109325 80.79%
case_4	0.010531	0.115185 16.43%	0.033442 31.55%	0.088671 0.00%	2.349564 100.00%	0.109325 80.79%
case_5	0.011266	0.106596 7.74%	0.027390 7.75%	0.099276 11.96%	2.349564 100.00%	0.109325 80.79%

Table 11.10 – Results of the static analyses on 5 cases

The column **fitness** in the table is the value of the objective function, and the next 5 columns the corresponding results in the static analysis, with each case showing the variation obtained. It can be seen that the upper control arm achieves always big variations that happen to be the same, allowing for great weight savings, while the lower control arm presents smaller variations and different from case

to case. From the column with the fitness, it can be seen that this value is not strictly related with the performance of the components in the static analysis. The 5th case, with the highest fitness value, is nevertheless rather conservative regarding the variations in stiffness of the components. The 3rd case, on the other hand, presents the lowest fitness value, but it achieves the highest variations in the static analysis. The 4th case must be dismissed since the topology optimisation process does not allow 0% variation as a target. The first three cases have too much variation in stiffness between the different directions of movement of the hardpoint. For instance, it is not realistic to try to achieve a small variation in the stiffness of a hardpoint in x direction, while going for a 40% increase in the y direction. Therefore, for the topology optimisation, the 5th case will be chosen based on the smaller variation in stiffness between the different load cases.

In conclusion it can be said that, although the genetic algorithm optimisation does provide an adequate solution for the problem, this may not be the one that in the static analysis delivers the highest variation in the displacements, which in turn allows the topology optimisation process to achieve the highest weight saving. Hence, the genetic algorithm must be implemented several times and multiple different solutions analysed, in order to find the one that offers more potential

12 TOPOLOGY OPTIMISATION

The topology optimisation is the stage in which the components are examined in order to find those elements in the mesh of the FEM that can be eliminated, thus weakening the structure, in order to adjust the stiffness of the component to the values specified by the multi-objective optimisation process driven by a genetic algorithm in the previous stage. In order to accomplish this goal, a software package called Altair OptiStruct is employed. The use of programming algorithms for solving problems in structural optimisation is well established and described in detail in the literature **[12.1]**.

As an introductory example of the capabilities of this software, the following case from online documentation **[12.2]** is presented in which an engine mount bracket was to be optimised. The goal was to minimise the mass of the bracket while maintaining the same stiffness and strength as an existing bracket design (**Figure 12.1**), with a mass of 950 grams.

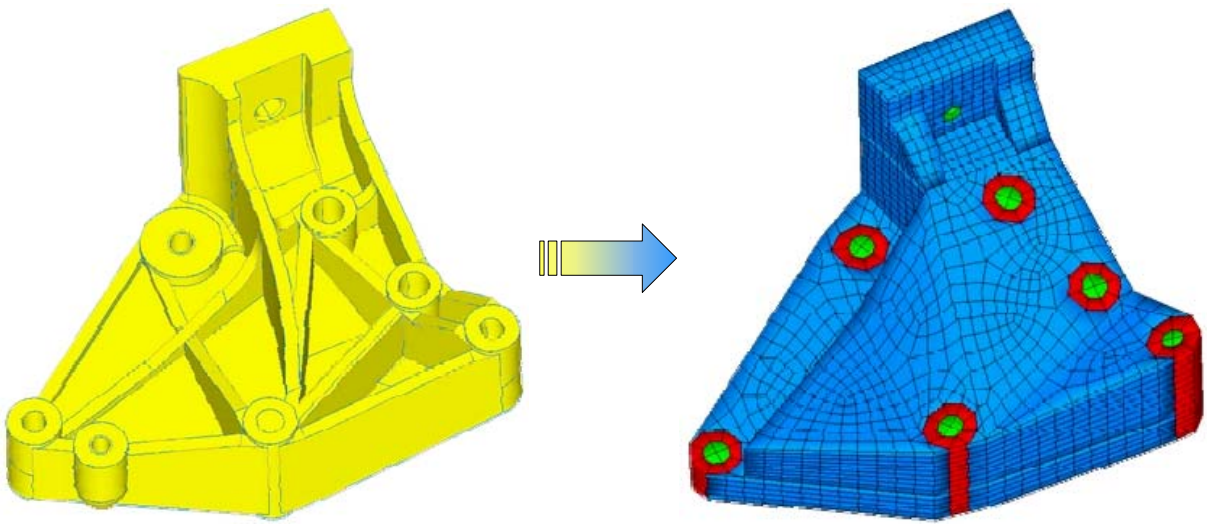


Figure 12.1 – Existing engine mount bracket and its design space FE model

The existing design was used to determine the package space for the OptiStruct analysis and used to build a solid finite element model, as shown in **Figure 12.1**. The blue area in the model was design space while the red and green areas were non-design space. Load cases were run on the bracket model in order to determine the performance of the part and set the stiffness targets. OptiStruct was used to make a topology optimisation analysis of the package space model. The optimisation yielded the density results show in **Figure 12.2**. After studying and interpreting the OptiStruct results, a new bracket

design was made. For verification of the results, a finite element model was made of the new design and a linear-static analysis was completed. The stiffness and strength of the new design was determined to be the same as the existing design, with a new mass of 730g, that is, 23.16% reduction.

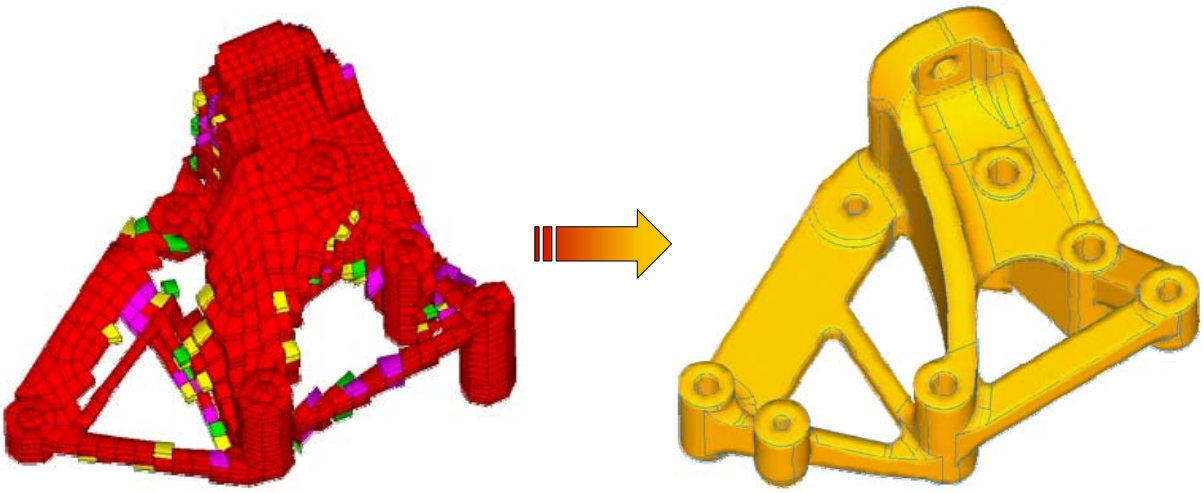


Figure 12.2 – Result of the topology optimisation and redesigned component

What OptiStruct does internally to achieve the result shown on the left side of **Figure 12.2** is a gradient based optimisation process **[12.1]** where the problem is posed as follows: there is a design space available to allocate material, with certain boundary conditions (constraints and loads, defined by the static analyses). The material inside the design space is assumed to be non-homogeneous, with a relative density ρ between 0 and 1. The elasticity properties are a function of density.

After building a finite element discretisation of the design space, a function between density and elasticity using a penalty factor is used to evaluate the solutions. The purpose of this penalty function is to bias the process towards solutions that offer 0 or 1 density, and not intermediate (unfeasible) values. At each iteration, the solution undergoes the static analysis in order to check the stiffness targets. The process can be schematically represented as in **Figure 12.3**.

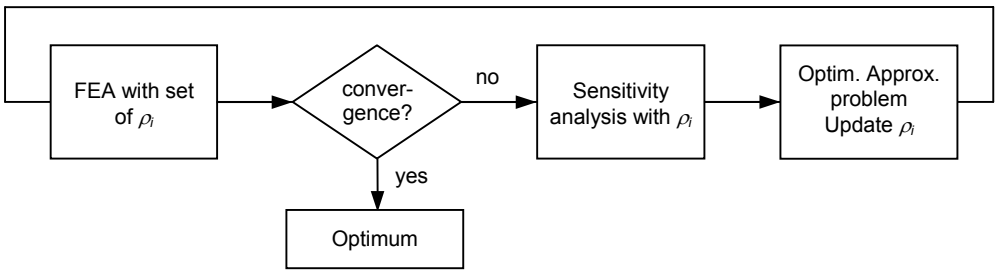


Figure 12.3 – Topology optimisation process

The main advantages of this method are:

- one design variable per element,
- easy to implement,
- and robustness.

When the displacements from **Table 11.9** are passed to the Topology Optimisation process, OptiStruct creates first a solution that minimises the mass and pursues the stiffness targets, which has a density distribution varying from 0 to 1.

The result of the topology optimisation process[10] on the lower control arm are shown in **Figure 12.4**, where it can be seen the areas with the lower relative density in blue, which are going to be removed in a later stage. As it was mentioned, the transition area of intermediate values of relative density is minimised in order to obtain a result as realistic as possible. The mass of the original lower control arm is 11.402 kg, while the mass attained by the topology optimised lower control arm is down to 10.063 kg, which represents a reduction of 1.339 kg or -11.74%. The areas right on and surrounding the hardpoints are non-design areas, to prevent the algorithm from weakening the attachment points to the chassis and wheelcarrier.

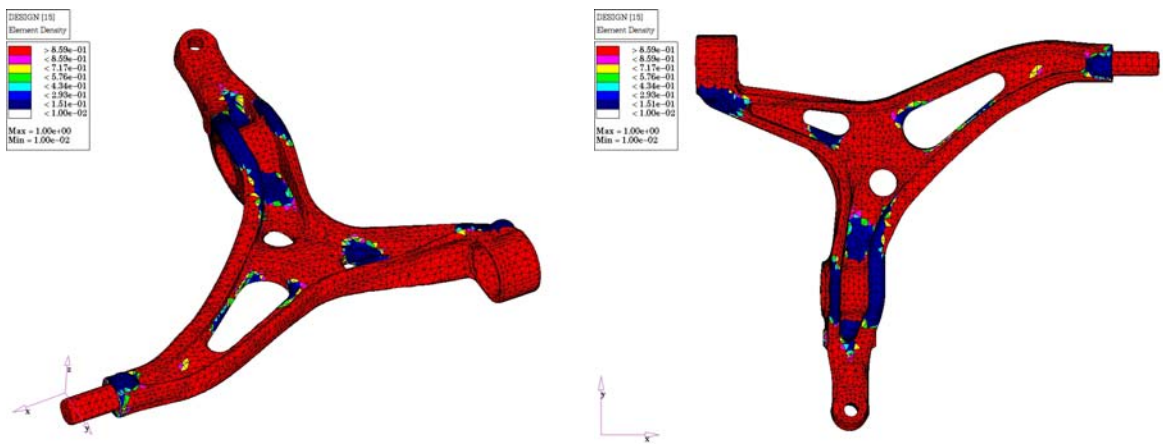


Figure 12.4 – Low density areas (blue) in the lower control arm

In the case of the upper control arm (**Figure 12.5**) the blue areas are concentrated in the ribs of the arms connecting the outer hardpoint with the two inner hardpoints. The mass of the original component is 0.783 kg, while the mass attained by the topology optimisation is down to 0.558 kg, which represents a reduction of 0.225 kg or -28.74%.

[10] In order to highlight the influence of different parameters in the process, the FE mesh used for the topology optimisation is particularly coarse, which may lead to non-optimum results.

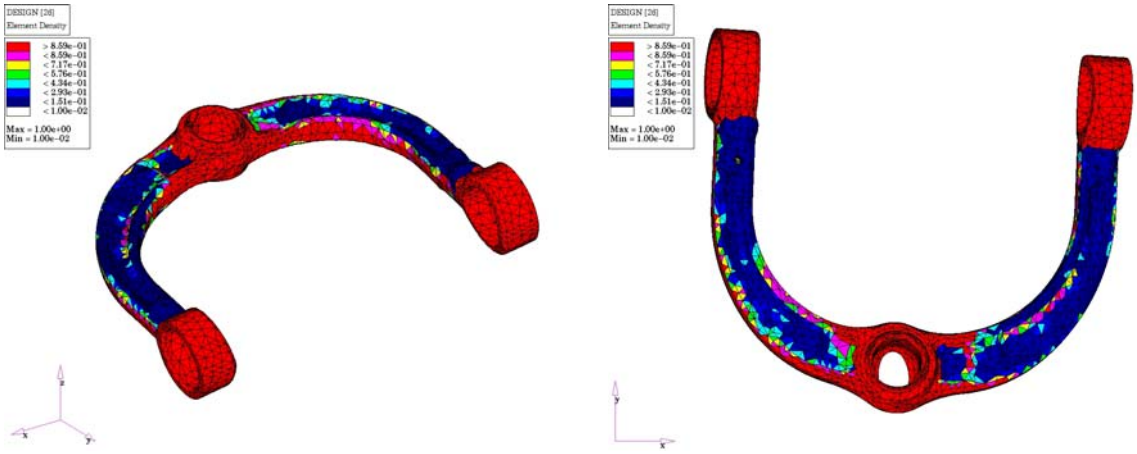


Figure 12.5 – Low density areas (blue) in the upper control arm

The performance of the two components on the static analysis is given in **Table 12.1**. The first line, **target**, refers to the values retrieved from the static analysis using the components based on the optimised stiffness matrices that resulted from the genetic algorithm. The second line, **optimised**, refers to the values retrieved from the static analysis using the components resulting from the topology optimisation process, in which their relative density varies from 0 to 1.

	lower control arm (hardpoint 5003)			upper control arm (hardpoint 6003)	
	Δx [mm]	**Δy [mm]**	**Δz [mm]**	**Δx [mm]**	**Δy [mm]**
target	0.106596	0.027390	0.099276	2.349564	0.109325
optimised	0.106600	0.027380	0.099230	2.344	0.1091
error	0.00%	-0.04%	-0.05%	-0.24%	-0.21%

Table 12.1 – Results of the static analysis on the topology optimised components

13 VERIFICATION

13.1 Relative density threshold

After the components have undergone the topology optimisation process, they must be analysed to verify that the target measurements have been achieved. However, in reality it is generally not possible to manufacture a component with heterogeneous density distribution as required. Therefore, there must be a threshold value to set the limit that defines which elements remain in the FE mesh and which are eliminated. For example, if the threshold is 0.6 those elements in the FE mesh with relative density lower than 0.6 are deleted, while the elements with density from 0.6 to 1 are all set to relative density 1 (**Figures 13.1** to **13.4**).

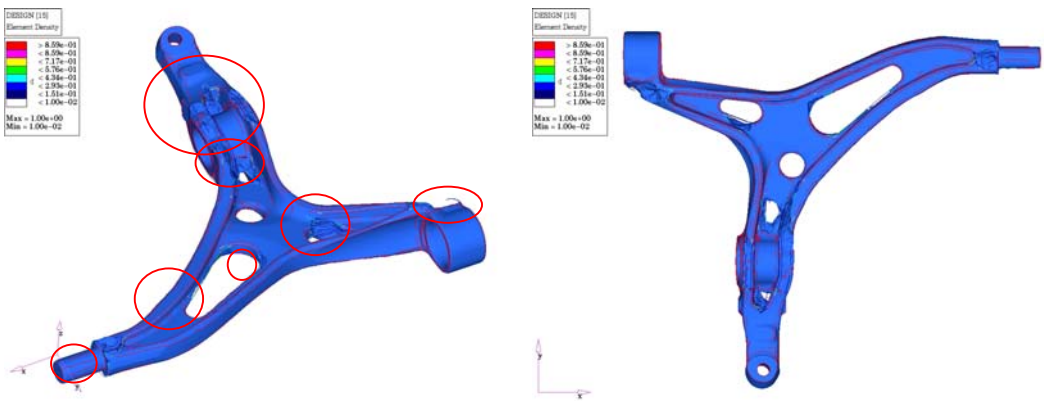


Figure 13.1 – Lower control arm, threshold value: 0.6

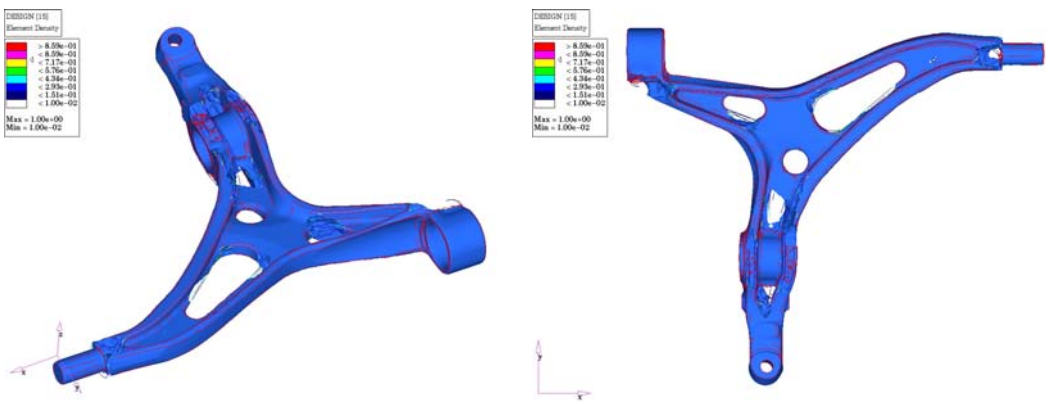


Figure 13.2 – Lower control arm, threshold value: 0.8

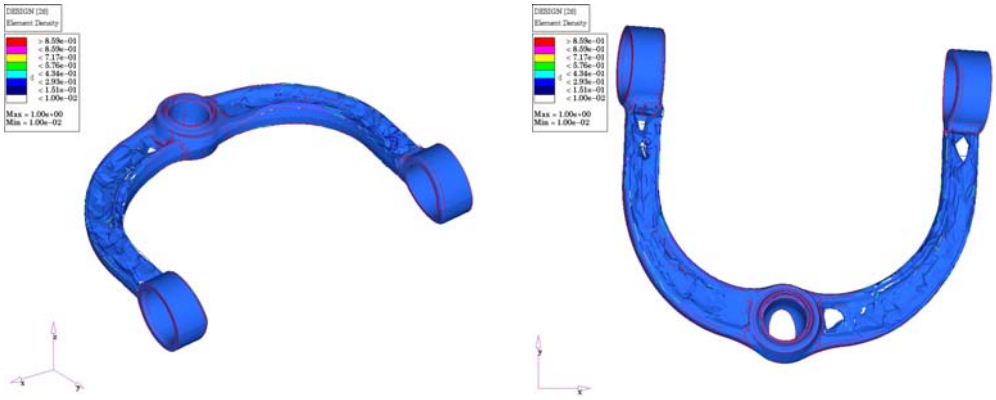


Figure 13.3 – Upper control arm, threshold value: 0.6

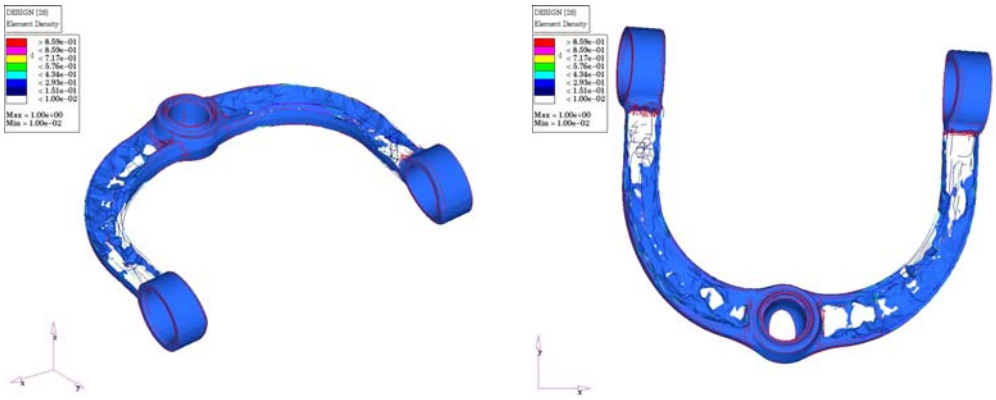


Figure 13.4 – Upper control arm, threshold value: 0.8

Clearly the elastic properties of the resulting component will differ from those achieved by the topology optimisation process. Furthermore, different threshold values lead to different stiffness values of the components.

13.2 Preparing the FE mesh

In order to analyse both the stiffness of the components that result from the topology optimisation and the compliances of the suspension assembly that makes use of them, the files containing the topology optimised components must be prepared. This intermediate step becomes necessary since by applying a threshold limit to the density, some elements in the FE mesh are erased. The main concern is that during this process some of the remaining elements in the mesh might be attached to it by only 2 nodes, 1 node or even detached from it, as shown in **Figures 13.5** and **13.6**.

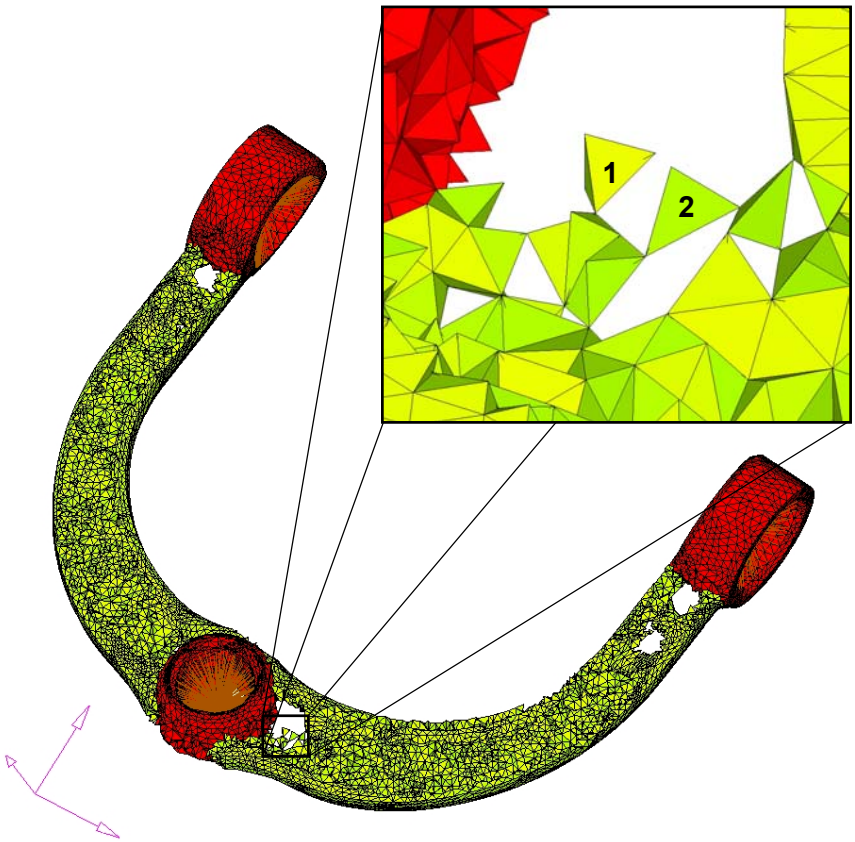


Figure 13.5 – Example of elements attached to the mesh by only one node (**1**) or by two nodes (**2**)

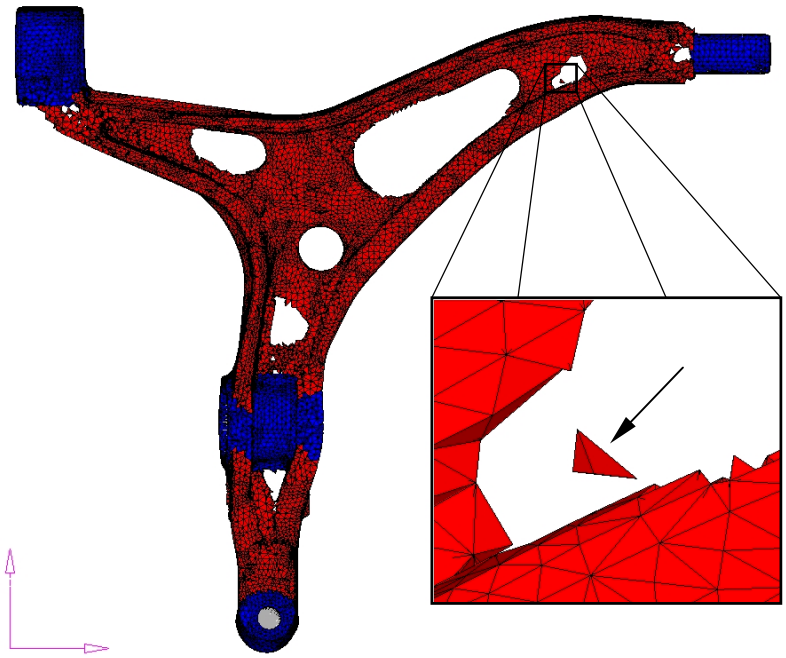


Figure 13.6 – Example of elements detached to the mesh

Mathematically, this situation has negative effects on the stiffness matrix and it can not be condensed, so the .pch file resulting from the SOL103 1st run is not correctly produced. This can be solved depending on the element situation. In the case of an element attached to the mesh by 2 nodes, a rigid element (called RBE2) must be placed connecting the problematic node to another one in the mesh, so to prevent the element to rotate around the axis made by the 2 attached nodes. In the other 2 cases (the element is attached to the mesh by only 1 node, or not attached at all) the element makes no contribution to the mesh stiffness so it can be simply erased. All of the mentioned operations are performed in an FE pre-processor such as Altair HyperMesh.

13.3 Component FE mesh check

Needless to say, visually checking a component mesh is not viable, so the way to systematically detect these problematic elements and the nodes is by trying to make a condensation of the component FE model (by running a SOL103 1st run) and looking into the log file (with extension .f06) coming out of the Nastran run. At the end of this file there is a list with the nodes belonging to the problematic elements. This process is once again carried out automatically by MATLAB through the file phase_7.m, as described next.

Step 33 – A dialog box (**Figure 13.7**) is displayed to confirm that the .bdf files corresponding to the topology optimised components have been checked against spurious lines sometimes saved with the file during the topology optimisation process, such as those containing load cases or keywords than can cause the process to fail.

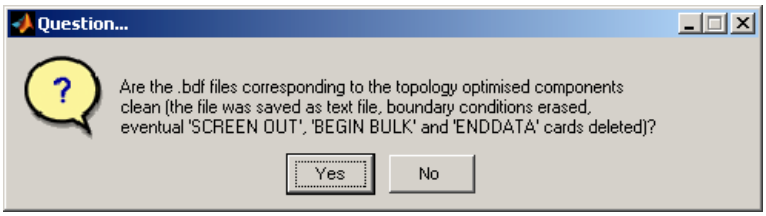


Figure 13.7 – Dialog box to confirm that the file is prepared

Step 34 – A dialog box (**Figure 13.8**) is displayed to select the .bdf files corresponding to the topology optimised components.

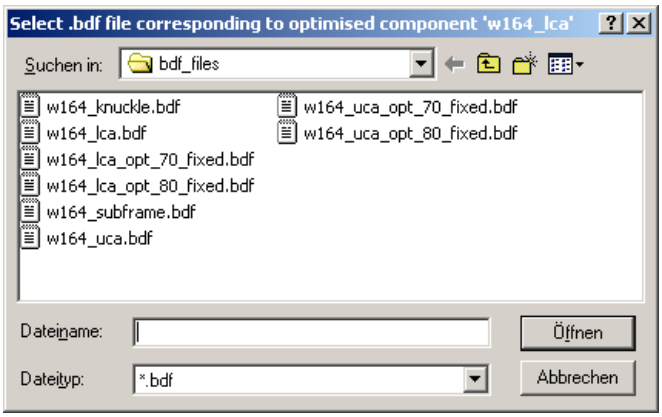


Figure 13.8 – Dialog box to select the files containing the optimised component

Step 35 – Based on the selected components and the information entered in phase_1, the file bdf2pch_template.dat is used to generate the file bdf2pch.dat, and this is executed in batch mode by Nastran.

Step 36 – The file bdf2pch.f06 is open and the problematic nodes, if existing, read. All the unnecessary files remaining from the process are then erased.

Finally the user receives a list of node numbers for every component that specifies those nodes that pose a problem to the condensation of the mesh, which help to pinpoint those elements defectively attached to the mesh. With the help of a FE pre-processor as mentioned, the elements can be erased or properly connected and the amended .bdf file of each component is saved.

13.4 Performance check

The last step in the procedure to close the loop and perform the check over the target cascade objectives can now be performed by MATLAB with the help of the m-file phase_8.m. All the information has already been entered, and only the names of the prepared files from the topology optimised components must be specified.

Step 37 – A dialog box (**Figure 13.9**) is displayed to select the .bdf files corresponding to the fixed topology optimised components. Together with the information entered in phase_1, all the necessary files to allow the simulations in ADAMS/Cars are created, as well as those for the static analysis.

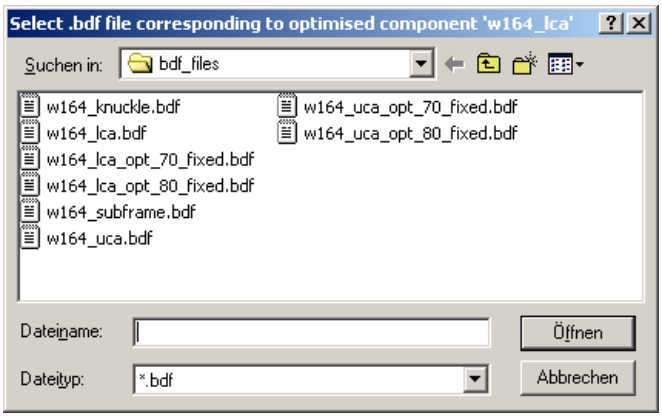


Figure 13.9 – Dialog box to select the file containing the optimised mesh, fixed

Step 38 – The simulations are performed and the measurements are read and stored in the table mea_top.

Step 39 – The load cases are applied to the topology optimised components by calling Nastran in batch mode. The results are stored in the array sol101_set_top.

Step 40 – The results are rearranged in the matrices sol101_top to improve the readability.

13.5 Stiffness targets

As indicated by the procedure, by raising the limit of ρ more elements are deleted from the mesh and the component becomes lighter but more elastic, results revealed in the outcome of the static analyses shown in **Table 13.1**.

The line labelled as **reference** states the original values of the static analysis and the weight of the component before the optimisation. The line labelled as **target** states the values of the static analysis that the optimised component can achieve without altering the performance of the suspension, according to the results from the genetic algorithm optimisation process.

The line labelled as **variable** ρ contains the results of the topology optimisation, where the elements of both FE models have relative densities varying from 0 to 1. It can be seen that the software is very capable of finding a solution for the optimisation problem, achieving a maximum deviation of the targets of –0.05% for the lower control arm, and –0.24% for the upper control arm (for the whole table, the variation is calculated with respect to the target values, not the reference) with important weight savings.

	lower control arm				upper control arm		
	Δx [mm]	**Δy [mm]**	**Δz [mm]**	**weight [kg]**	**Δx [mm]**	**Δy [mm]**	**weight [kg]**
reference	**0.098934**	**0.025421**	**0.088671**	**11.402**	**1.174782**	**0.060470**	**0.783**
target	**0.106596**	**0.02739**	**0.099276**	**-**	**2.349564**	**0.109325**	**-**
variable ρ	0.1066	0.02738	0.09923	10.063	2.344	0.1091	0.558
variation	0.00%	-0.04%	-0.05%	-11.74%	-0.24%	-0.21%	-28.74%
ρ > 0.6	0.103743	0.026677	0.095684	10.03	1.841149	0.090624	0.537
variation	-2.68%	-2.60%	-3.62%	-12.03%	-21.64%	-17.11%	-31.42%
ρ > 0.8	0.106646	0.027385	0.099647	9.688	4.783079	0.136080	0.475
variation	0.05%	-0.02%	0.37%	-15.03%	103.57%	24.47%	-39.34%

Table 13.1 – Results of the static analysis for the topology optimised components

The next two groups of lines highlight the differences in results by applying different threshold limits to the density of the elements that will be erased from the mesh. As it can be seen, applying a limit of 0.6 means that too many elements remain in both FE meshes (which are all taken to relative density 1), since the stiffness of the components is actually increased. When applying a limit of 0.8 the components become more flexible, although too flexible in the case of the upper control arm.

13.6 Compliances of the suspension

What really matters at the end of the optimisation process is that the compliances of the suspension remain intact or at least within certain limits, according to the criteria adopted by the user, regardless of the changes that the components have undergone in order to fulfil the stiffness targets. The margin of 0.5% allowed to the genetic algorithm optimisation for the measurements was related with the intricacies of the algorithm implementation rather than with any particular requirement on the suspension itself. However, it must be kept in mind that, for a weight reduction process like the one tackled in this thesis, at the end of it the compliances of the suspension must be verified.

Table 13.2 shows the result of three sets of simulations performed in ADAMS in order to determine the new performances of the suspension. The first set of results corresponds to the original suspension. The next two sets correspond to the performance of the suspension when using the components resulting from the topology optimisation with different threshold values for the relative density. The use of a threshold value of 0.6 produces acceptable results, while raising it to 0.8 has already shown to be

too aggressive with the upper control arm, so the compliances of the suspension reflect this by diverging too much from the targets.

	measurements [°/kN]			variation		
reference	0.295532	-2.481800		-	-	
	-0.197928	2.276770		-	-	
	0.158331	-0.194118	-0.223060	-	-	-
ρ = 0.6	0.298282	-2.519400		0.93%	1.52%	
	-0.195820	2.240600		-1.07%	-1.59%	
	0.158348	-0.194748	-0.222520	0.01%	0.32%	-0.24%
ρ = 0.8	0.309283	-2.676900		4.65%	7.86%	
	-0.185386	2.067360		-6.34%	-9.20%	
	0.157798	-0.195608	-0.220100	-0.34%	0.77%	-1.33%

Table 13.2 – Measurements based on the topology optimised components

As it was seen in **Table 13.1**, by the use of 0.8 as a threshold value the lower control arm delivers the closest performance to the stiffness targets for that component achieved by the genetic algorithm optimisation stage. Therefore, in **Table 13.3** are shown the results of combining this optimised component (lca) with the original upper control arm (uca).

	measurements [°/kN]			variation		
ρ_{lca} = 0.8 ρ_{uca} = orig.	0.296219	-2.489000		0.23%	0.29%	
	-0.198106	2.284010		0.09%	0.32%	
	0.158979	-0.194405	-0.224560	0.41%	0.15%	0.67%
ρ_{lca} = 0.8 ρ_{uca} = 0.7	0.300402	-2.546800		1.65%	2.62%	
	-0.194072	2.216040		-1.95%	-2.67%	
	0.158394	-0.194978	-0.222510	0.04%	0.44%	-0.25%

Table 13.3 – Measurements based on a combination of the original and the optimised components

The last group of values on the table allows discussing an important aspect of the topology optimisation. Since a threshold value of 0.8 is too much for the upper control arm (making it too weak) and 0.6 is too little (the component is still too stiff) a threshold value of 0.7 is used and, in combination with the lower control arm obtained by applying a threshold value of 0.8, the simulations in ADAMS are run and the compliances measured. As it can be seen, the variations are still too high. It would then be reasonable to seek for a threshold value between 0.7 and 0.8. Unfortunately, OptiStruct does not allow values other than multiples of 0.1, limiting in this way the possibilities for the quality of the results.

The best way to solve this matter is by refining the FE mesh before starting the process, bearing in mind right from the beginning the needs of the topology optimisation for a smoother mesh with much more elements. However, in this example it would mean re-starting the process all the way from phase_1. A proposed workaround to this issue is to adopt the model that results from applying the threshold value that delivers the closest results to the stiffness target, that is, 0.7, and to apply a small variation in the Young's modulus E in order to fine tune its stiffness according to the static analysis. The results of this procedure are shown in **Table 13.4**. The material of the upper control arm is aluminium, with a Young's modulus E_{Al} = 70.000 N/mm^2.

	Δx [mm]	Δy [mm]	weight [kg]	E/E_{Al}
reference	**1.174782**	**0.060470**	**0.783**	**1**
target	**2.349564**	**0.109325**	**-**	
uca (ρ = 0.8)	4.783079	0.13608	0.475	1
variation	103.57%	24.47%	-39.34%	
uca (ρ = 0.7)	2.215636	0.102058	0.509	1
variation	-5.70%	-6.65%	-34.93%	
uca (ρ = 0.6)	1.841149	0.090624	0.537	1
variation	-21.64%	-17.11%	-31.42%	
uca (ρ = 0.7)	2.35706	0.1085725	-	0.94
variation	0.32%	-0.69%	-	

Table 13.4 – Results of the static analysis on topology optimised upper control arm, with variation on the Young's modulus

13.7 Refining the FE mesh

The last two lines on the table show the stiffness values delivered by an upper control arm which its Young's modulus has been adjusted to 94% of the one from aluminium. If this component is used together with the lower control arm with a threshold value 0.8 (which gives the best results for this component), it is reasonable to expect that the compliances of the suspension are closer to the targets than those from any other combination of upper and lower control arms. The results of the simulation to measure the compliances are shown in **Table 13.5**, where it can be seen that the deviations from the targets are, in general, bigger for the modified than for the original Young's modulus. The reason behind this is that, although in **Table 13.4** the upper control arm with modified Young's modulus is closer than any other to the stiffness targets, the variation of the material's properties affects the over

all behaviour of the component and not only the stiffness of the outer hardpoint under specific load cases, as measured during the static analysis. This point highlights the fact that the static analysis itself is not comprehensive nor it covers every possible aspect of the behaviour of a component within a suspension. Hence, it can be said that the best way to deal with this issue is by starting off with the optimisation task with a FE mesh appropriate for the process. A finer mesh means a slightly longer time on the beginning of the process, specifically during the condensation process run in phase_1, but as an advantage it offers more possibilities to the topology optimisation process to achieve results closer to the stiffness targets, even after the threshold value for the relative density has been applied.

	measurements [°/kN]			variation		
ρ_{lca} = 0.8	0.300402	-2.546800		1.65%	2.62%	
ρ_{uca} = 0.7	-0.194072	2.216040		-1.95%	-2.67%	
E/E_{Al} = 1	0.158394	-0.194978	-0.222510	0.04%	0.44%	-0.25%
ρ_{lca} = 0.8	0.300746	-2.551600		1.76%	2.81%	
ρ_{uca} = 0.7	-0.193746	2.210750		-2.11%	-2.90%	
E/E_{Al} = 0.94	0.158308	-0.195035	-0.222210	-0.01%	0.47%	-0.38%

Table 13.5 – Measurements based on a modified Young's modulus

In order to highlight the advantages of such strategy, an optimisation process was run with a refined FE mesh in order to compare the results with those obtained in the previous chapter. The differences in the FE meshes for both the lower and upper control arms can be seen in **Table 13.6**.

	lca		uca	
	elements	nodes	elements	nodes
original (Tetra10)	25101	45095	9723	18633
refined (Tetra4)	194691	43557	177762	39509

Table 13.6 – Differences in the original and refined FE meshes

The change in element type from TET10 to TET4 is made because mathematically the later element type is better suited for topology optimisation than the former. TET10 is a type of element more (and sometimes the only) appropriate for non-linear analysis, and requires a geometrically increasing amount of time with respect to TET4, but delivers superior accuracy. TET4 is a simpler element, perfectly suited for linear analyses like the ones performed by Nastran during the static analyses. **Figure 13.10** shows the basic four-noded tetrahedral element, which takes into account only the corner nodes 1 to 4. Including the midside nodes 5 to 10, the ten-noded tetrahedral element is made, with the nu

meration system being only an example. The effect of these changes can be seen in **Tables 13.7** and **13.8**, where it can be appreciated that the components are slightly stiffer than before and the suspension is less compliant. This is due to the change in the type of element used in the mesh, which stiffens the mesh slightly; on the other hand, the increase in number of elements accounts for a weaker component, but it does not completely counteracts the effect of the change in the element type. As a combined result, the components show smaller displacements, during the static analysis.

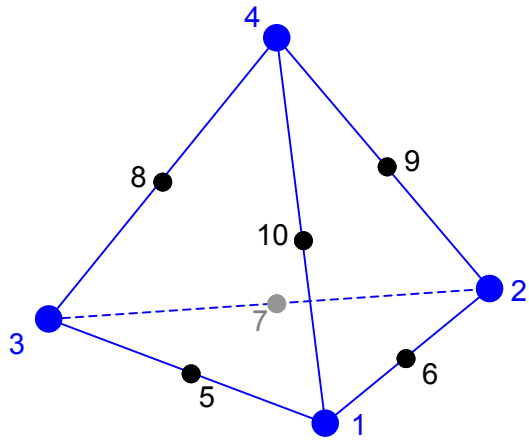


Figure 13.10 – TET4/TET10 element

	lca			uca	
	Δx [mm]	Δy [mm]	Δz [mm]	Δx [mm]	Δy [mm]
original	0.098934	0.025421	0.088671	1.174782	0.060470
refined	0.097974	0.024565	0.085923	1.155517	0.059622
variation	-0.97%	-3.37%	-3.10%	-1.64%	-1.40%

Table 13.7 – Variation of stiffness from the original to the refined FE meshes

original [°/kN]	0.2955	-2.4818	
	-0.1979	2.2768	
	0.1583	-0.1941	-0.2231
refined [°/kN]	0.2955	-2.4807	
	-0.1981	2.2764	
	0.1581	-0.1939	-0.2225
variation	-0.02%	-0.04%	
	0.07%	-0.02%	
	-0.17%	-0.09%	-0.27%

Table 13.8 – Variation of compliances in the suspension

After the optimisation process is run over the suspension assembly using the refined FE meshes for the components, the sensitivity analysis shows that the same 11 DOF are important for the measurements. These are optimised by the genetic algorithm, and the result is used to calculate the new stiffness targets, shown in **Table 13.9**, which are passed on to the topology optimisation stage.

	lca			uca	
	Δx [mm]	**Δy [mm]**	**Δz [mm]**	**Δx [mm]**	**Δy [mm]**
refined	0.097974	0.024565	0.085923	1.155517	0.059622
optimised	0.105563	0.029764	0.104108	2.311033	0.078435
variation	7.75%	21.17%	21.17%	100.00%	31.55%

Table 13.9 – New stiffness targets for the optimised components, based on the refined FE meshes

The results of the topology optimisation process are presented in **Table 13.10**. Given the wide difference between the target variation for the optimised upper control arm (3.17 times bigger in *x* than in *y* direction) as well as for the lower control arm (2.73 times bigger in *y* and *z* than in *x* direction), the topology optimisation process is not able to achieve the targets, specially for the lower control arm.

	lower control arm				upper control arm		
	Δx [mm]	**Δy [mm]**	**Δz [mm]**	**weight * [kg]**	**Δx [mm]**	**Δy [mm]**	**weight * [kg]**
refined	**0.097974**	**0.024565**	**0.085923**	**11.450**	**1.155517**	**0.059622**	**0.759**
target	**0.106596**	**0.02739**	**0.099276**	**-**	**2.349564**	**0.109325**	**-**
variable ρ	0.1066	0.02738	0.09923	9.863	2.308	0.0784	0.5917
variation	8.80%	11.46%	15.49%	-13.86%	99.74%	31.50%	-22.04%
ρ > 0.6	0.1035	0.02833	0.1002	9.763	1.9939	0.0735	0.5831
variation	5.65%	15.32%	16.60%	-14.73%	72.56%	23.27%	-23.18%
ρ > 0.7	0.1042	0.02883	0.1017	9.654	2.0986	0.07529	0.5745
variation	6.37%	17.35%	18.33%	-15.69%	81.61%	26.28%	-24.31%
ρ > 0.8	0.1052	0.02956	0.1037	9.519	2.2693	0.07778	0.5642
variation	7.36%	20.34%	20.63%	-16.86%	96.39%	30.46%	-25.67%

* due to the different amount of elements in the meshes, the volume enclosed for the same component is different, so it is the reported weight of the components.

Table 13.10 – Results of the static analysis for the topology optimised components

It is indeed very difficult to weaken a component in a way that the variation in its reaction in a given direction is very different from the variation in another direction. Hence, comparing with the target variations shown in **Table 13.9**, the lower control arm including all elements with relative density between 0 and 1 has a ratio of less than 2 when comparing the maximum (in *z*) with the minimum (in *x*) variation achieved. In principle, the upper control arm seems to achieve its targets much better.

The components are presented in **Figures 13.11** and **13.12**. Comparing these pictures with those in **Figures 12.4** and **12.5**, it can be seen how the finer mesh effectively reduces the transition areas. Comparing the variations in weight achieved by the components with all relative densities and the components with relative density greater than 0.8, the variation in the coarser mesh is 14.87% (upper control arm) and 3.73% (lower control arm), while for the finer mesh is just 4.65% (upper control arm) and 3.49% (lower control arm).

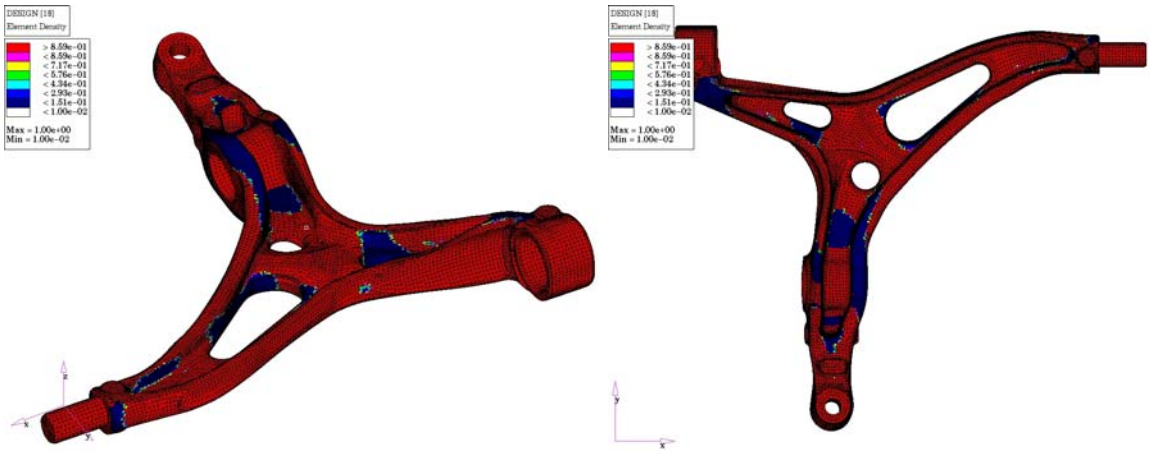


Figure 13.11 – Low density areas (blue) in the lower control arm

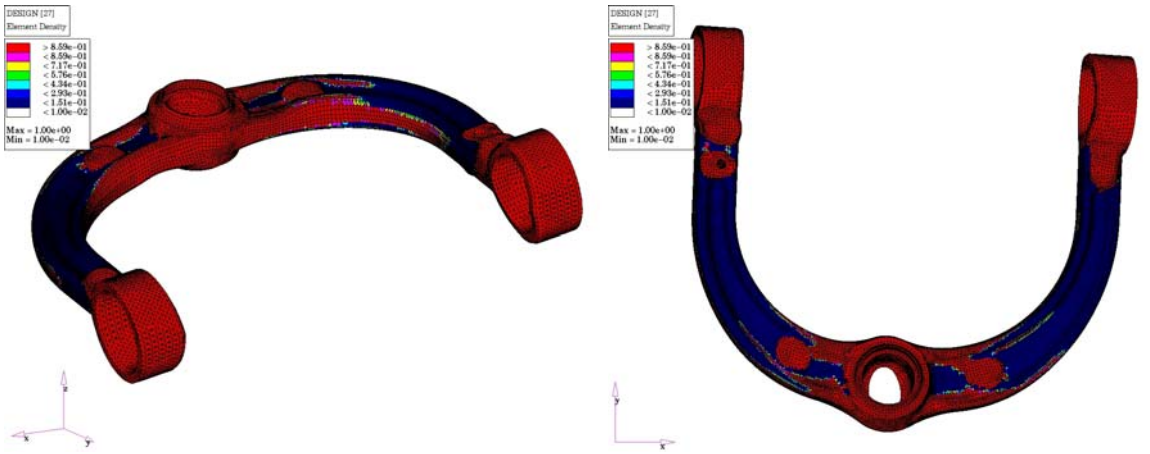


Figure 13.12 – Low density areas (blue) in the upper control arm

The effect of the different threshold limits for the relative density on the suspension compliances can be seen on **Table 13.11**. The relationship between the threshold limit and the variation in the measurements is very obvious: the higher the limit, the higher the variation in the compliances, because more elements are eliminated and so the component is further weakened. Therefore, it is up to the user to determine how much performance is to be sacrificed in exchange for a reduction in weight.

	measurements [°/kN]			variation		
refined	0.295474	-2.480700		-	-	
	-0.198066	2.276420		-	-	
	0.158062	-0.193946	-0.222460	-	-	-
0.6	0.297881	-2.513500		0.81%	1.32%	
	-0.196112	2.249860		-0.99%	-1.17%	
	0.159191	-0.194748	-0.224770	0.71%	0.41%	1.04%
0.7	0.298282	-2.518600		0.95%	1.53%	
	-0.195751	2.245300		-1.17%	-1.37%	
	0.159351	-0.194806	-0.225100	0.82%	0.44%	1.19%
0.8	0.299084	-2.528500		1.22%	1.93%	
	-0.194983	2.235440		-1.56%	-1.80%	
	0.159557	-0.194978	-0.225560	0.95%	0.53%	1.39%

Table 13.11 – Estimated variation of the measurements after the optimisation process

14 CONCLUSIONS

14.1 Scope of the results

From the results obtained after the topology optimisation phase presented in **Table 13.1**, for the example used for the development of the optimisation process the weight savings are somewhere between 11.74% and 15.03% for the lower control arm, and between 28.74% and 39.34% for the upper control arm, depending on the threshold limit selected for the relative density. The difference of the results achieved by both components highlights the importance of each of them for the overall performance of the suspension assembly. Since the upper control arm admits a bigger reduction in stiffness than the lower control arm, it is evident that its roll in the suspension assembly is just one of positioning the upper hardpoint of the wheelcarrier, without taking up important loads. On the other hand, the lower control not only must help positioning the lower hardpoint of the wheelcarrier, but also cope with the loads transmitted by the suspension system. Usually, in a double wishbone suspension lay-out like the one of the development case, the forces acting on the lower control arm are four times higher than those acting upon the upper control arm.

It important to highlight that the criterion used to determine the modifications of the stiffness targets of the components is not comprehensive; it lacks of a unified view of all the requirements that the components must comply with, in order to be effectively useful and a realistic option. These extra criteria mainly involve targets for durability (fatigue) that are not taken into account in this thesis: all the effort is directed towards finding out the right targets from a stiffness point of view, which is carried out in a static analysis. It does not involve any dynamic analysis concerning the eigenvalues (natural frequencies of the component), change in the centre of mass, etc., all of which have an impact in the natural frequency of the suspension which, in turn, in its ability to keep the wheel in contact with the road and keep the vehicle within the required levels of NVH. The ability of the suspension assembly to absorb energy under plastic deformation (for instance during a crash) is closely related to its behaviour during an accident, and this criterion has also been overlooked as well as any other safety concerns.

For all of this, it can be said that the results presented in **Tables 13.1** and **13.10** in the columns **weight** represent a lower bound to what can be achieved in terms of weight reduction. Further work in the remaining areas of the components requirements would most probably increase the weight of the components, in a hitherto undetermined degree.

14.2 Improving the results

Down to what the developed optimisation process does have into account, it is particularly important that the static analysis cases are carefully considered, in order to effectively cover the spectrum of use of a component within an assembly. If the series of static analysis that set the stiffness targets are not comprehensive, they can lead the topology optimisation process to take material out of the component where it is needed by it in order to manage the applied loads and deliver the right performance within the assembly. The topology optimisation will then deliver a component that does meet the new stiffness targets, but when the component is tried out in the suspension assembly, the latter does not achieve the target compliances.

A similar effect could be seen when modifying the Young's modulus of the upper control arm, where the stiffness targets where met but, once in the virtual suspension rig in ADAMS, the assembly does not perform as expected. Not only has the stiffness of the component in the desired hardpoints and directions been changed, but also in all the other parts and directions. The set of load cases used for the static analysis proves, from this point of view, to be not sufficient to univocally define the scope of influence of the component in the suspension assembly, and more load cases should be used. In any case, the best solution, as stated in the previous chapter, is to start off from a finer mesh in the FE model that allows the topology optimisation process to achieve higher quality in the results, by decreasing the size of the transition elements, some of which are erased when setting the threshold value for the relative density. By using a finer mesh, the differences obtained with different threshold values are reduced and, more important, the differences between the output model with relative densities from the topology optimisation, and that with a threshold value already applied.

14.3 More than a Cost Reduction Engineering tool?

There is an important question that arises and must be answered: given that the optimiser handles the DOF that can reduce or increase the compliances of the suspension assembly, can this process be used to tune its performance? The answer to this question requires a careful breakdown analysis.

In principle, the method is developed as a cost reduction engineering tool, not a compliances tuning tool. The reasons do not lay in the code itself, because the targets are set in phase_5, when setting up the parameters for the genetic algorithm optimiser and, in principle, the algorithm can handle variations in the targets. To study the possibilities, six cases are simulated in which the measurements have:

- a target of a very small variation[11] in all the compliances but one (camber under lateral force), which should achieve a 4% increase,

[11] As explained in Paragraph **11.6.8**.

- a target of a very small variation in all the compliances but one (camber under lateral force), which should achieve a 1% reduction,
- a target of a very small variation in all the compliances except those measurements taken under lateral force, which should achieve a 4% increase,
- a target of a very small variation in all the compliances except those measurements taken under lateral force, which should achieve a 1% reduction,
- all the measurements should achieve a 4% increase.
- all the measurements should achieve a 1% reduction,

The results of these optimisation cases can be seen in **Table 14.1**.

	0.295474	-2.480700				
	-0.198066	2.276420				
	0.158062	-0.193946	-0.222460			
	0.297136	-2.496700		0.56%	0.64%	
3,2 + 4%	-0.199160	2.285050		0.55%	0.38%	
	0.158927	**-0.195493**	-0.223320	0.55%	**0.80%**	0.39%
	0.297079	-2.498600		0.54%	0.72%	
3,2 - 1%	-0.199120	2.287900		0.53%	0.50%	
	0.158824	**-0.194634**	-0.223560	0.48%	**0.35%**	0.49%
	0.297365	-2.494700		0.64%	0.56%	
3,: + 4%	-0.199378	2.286650		0.66%	0.45%	
	0.162646	**-0.196410**	**-0.231950**	**2.90%**	**1.27%**	**4.27%**
	0.297136	-2.498700		0.56%	0.73%	
3,: - 1%	-0.199017	2.282820		0.48%	0.28%	
	0.157265	**-0.194118**	**-0.220090**	**-0.50%**	**0.09%**	**-1.07%**
	0.304069	**-2.578300**		**2.91%**	**3.93%**	
+ 4%	**-0.201206**	**2.352020**		**1.59%**	**3.32%**	
	0.166920	**-0.198473**	**-0.241060**	**5.60%**	**2.33%**	**8.36%**
	0.298396	**-2.513300**		**0.99%**	**1.31%**	
- 1%	**-0.195974**	**2.253270**		**-1.06%**	**-1.02%**	
	0.157231	**-0.194978**	**-0.219820**	**-0.53%**	**0.53%**	**-1.19%**

Table 14.1 – Optimisation analyses with tuned compliances

The reason to use 4% increase on the measurements but only 1% reduction is because the compliances, in general, are increased when the DOF are weakened, as it is shown in **Table 11.2**. Hence, to stiffen the performance of the suspension and reduce the compliances is much more difficult for the optimiser, since it has fewer parameters to tune which, on top, do have influence in the rest of the measurements that are pretended to be kept unaltered or with a small variation at the most. A straightforward solution for this issue would be to allow the optimiser to use modification factors higher than

1, but this is not viable due to the limitation imposed but the topology optimiser. Modification factors higher than 1 would mean to strengthen the component, which implies adding mass to it, operation that the mentioned optimiser does not handle.

As it can be noted, none of the analyses is actually successful. The reason for the first two analyses is that the DOF that could modify the camber compliance under lateral force have influence too in other measurements, and the margin to tune this particular compliance without modifying the others is very narrow. In the next two cases, by trying to tune all the compliances measured under lateral acceleration the results are improved. This is because every ADAMS simulation has a set of most influential DOF, which can be common to other simulations and, in this particular case, the DOF 14 of the upper control arm is not important for the measurements taken under acceleration or braking forces, as it was determined during the sensitivity analysis. In the last two cases, trying to tune the whole suspension in a certain direction (stiffer or more compliant) proves to be again rather difficult, since the influence of the DOF over the measurements is, most of the time, at odds.

In the case of DOF 2 from the lower control arm, it is very influential when applying a lateral force to the suspension, increasing the compliances when weakened, but during braking and acceleration it decreases them. Also from the lower control arm, DOF 8 has the same effect in every measurement when a lateral force is applied to the assembly, but the opposite during braking and acceleration. On the other hand, weakening DOF 1 from the upper control arm increases the toe and longitudinal wheelcentre compliances under acceleration, but decreases them under braking. In short, this slightly detailed analysis shows clearly that due to the fact that some DOF "cooperate" with each other to modify some measurements, and some are at odds, constraining the optimisation algorithm, a solution that satisfies the constraints when they involve tuning one particular measurement is rather complicated, when not impossible.

In short, the answer to the question is *no*, the developed process is not a compliance tuning tool. The interrelationship between the DOF involved in each measurement is rather complicated, and leaves the optimiser with extremely little margin to play with the parameters and achieves, in general, poor results. Part of the reason is that the genetic algorithm introduces all the constraints (the requirements for the measurements) as some sort of Lagrange multipliers in the figure of the so called penalty function, which decreases the fitness of an individual in a given amount. It must be taken into account, however, that this function condenses all the deviations from the measurements into one single number, making it impossible for the algorithm to distinguish too solutions, for instance, one in which two targets are away by 2%, and one in which one target is off by 1% and the other by 3%. For example, the fifth analysis in **Table 14.1**, where an increase of 4% in the compliances was the constraint, the average of all the increments is 4.007%, that is, extremely close, and still the variations are as high as 8.36%, which is outside the range of acceptable results.

What the developed process manages to do is to pinpoint those DOF in individual components that are key to the performance of the suspension to which they belong, studies their behaviour, and finds the right combination of possible ways to weaken the component without compromising the way the assembly performs. All of this from a static standpoint, taking into account the elasticities of the real components, and running the modifications in a mathematical way. For this, a pre- and post-processing of the components into and from a mathematical model is automatically carried out. As a result a lighter, yet as stiff suspension assembly is delivered.

14.4 Further work

Through out this thesis the different stages of the optimisation process have been introduced and analysed, and the shortcomings discussed. Therefore, a few comments about possible lines of work in order to perfect or enhance the capabilities of the process or to foresee its field of application are necessary.

During the sensitivity analysis, all DOF are perturbed and the change in the compliances measured. It is senseful to take in account what those DOF mean for the suspension. For example, where a balljoint is used, it is utterly obvious that the rotational DOF of that hardpoint make no contribution to the performance of the suspension, so the three rotational DOF of that hardpoint (such 5003 in **Figure 11.15**) can be skipped from the sensitivity analysis, hence speeding up the process. The same can be said about the rotational DOF in x direction of hardpoints 5001 and 5002 (in the same figure). Even when they typically contribute with as much as 25% of the spring rate of the suspension, compared with the stiffness of the control arms those values are insignificant. The relevance of the mentioned DOF can be seen in **Table 11.2** and **Figures 11.3** to **11.9**, where the change in compliances provoked by the perturbation on DOF 4 and 10 (rotation around x of hardpoints 5001 and 5002) DOF 16, 17 and 18 (rotation around x, y and z respectively of hardpoint 5003) are negligible.

In order to enable the option of considering a DOF unimportant by the software and override the sensitivity analysis, the corresponding changes in the code must be implemented.

Regarding the optimisation method itself, Gradient Based methods, despite their limitations, are not completely ruled out and it can be worth it to research deeper in order to determine their implementation. This could lead to better results or perhaps the same but at less computational expense.

With the application field in mind, the developed tool opens the possibility to re-analyse suspension assemblies and actually any other assembly with flexible components that has to fulfil compliance-type targets. The assembly can already fulfil its targets and deliver the required performance; however, by eliminating spurious bits of mass, hence lightening it, the performance can be further improved and the final product perfected. Furthermore, the output files of the topology optimisation stage, once veri

fied, can in principle be used in a fatigue analysis. This means that besides the stiffness aspect of the components, other demands can eventually be added to the procedure to extend its capabilities.

APPENDIX A – RECIPROCITY THEOREM

For two identically sized forces applied at the distinct points A and B on a linear structure, Maxwell's Reciprocity Theorem states that the displacement at A caused by the force at B is the same as the displacement at B caused by the force at A. As a result, the stiffness matrix of a linear system is symmetric, that is, $k_{ij} = k_{ji}$. A second way to put this theorem is to consider two forces successively applied on a system, and verifying that the total work done is constant, despite the order in which the two forces are applied. Thus, $k_{ij} = k_{ji}$.

For the proof of this theorem, consider the work done by forces f_1 and f_2, where the order of loading is *1* followed by *2* and then by its reverse. Define u_{ij} as the displacement at point i due to load at point j, and k_{ij} as the stiffness at point i due to load at point j.

By applying f_1 at the point i, the work done is

$$W = \int_0^u f_1 \cdot du \,. \tag{A.1}$$

In a linear system, $f = k \cdot u \Rightarrow u = \dfrac{f}{k}$, thus

$$W = \frac{1}{2} k_{ii} u_{ii}^2 = \frac{1}{2} \frac{f_1^2}{k_{ii}} \,. \tag{A.2}$$

By applying f_2 at point j, the work done is $\dfrac{1}{2}\dfrac{f_2^2}{k_{jj}}$. However, point i undergoes further displacement $u_{ij} = \dfrac{f_2}{k_{ij}}$, and the additional work done by f_1 ($= k_{ij} \cdot u_{ij}$) becomes

$$W = \int_0^u f_1 \cdot du = \frac{1}{2} k_{ij} u_{ij}^2 = \frac{1}{2} f_1 u_{ij} = \frac{1}{2} \frac{f_2 \cdot f_1}{k_{ij}} \,. \tag{A.3}$$

Thus, the total work done is

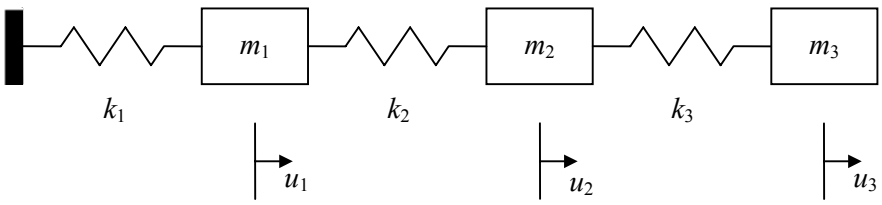
$$W = \frac{1}{2}\left(\frac{f_1^2}{k_{ii}} + \frac{f_2^2}{k_{jj}} + \frac{f_2 \cdot f_1}{k_{ij}} \right). \tag{A.4}$$

Reversing the order of loading, the total work done is

$$W = \frac{1}{2}\left(\frac{f_2^2}{k_{jj}} + \frac{f_1^2}{k_{ii}} + \frac{f_1 \cdot f_2}{k_{ji}} \right). \tag{A.5}$$

Because the work done in the two cases must be equal, from (A.4) and (A.5) then $k_{ij} = k_{ji}$.

Example:



For the system showed above there are 3 DOFs. In order to determine the stiffness matrix, consider first the case where $x_1 = 1$ and $x_2 = x_3 = 0$. The forces required at 1, 2, and 3 are:

$$f_1 = k_1 + k_2 = k_{11} \tag{A.6}$$

$$f_2 = -k_2 = k_{21} \tag{A.7}$$

$$f_3 = 0 = k_{31} \tag{A.8}$$

In a second case, $x_2 = 1$ and $x_1 = x_3 = 0$. Then:

$$f_1 = -k_2 = k_{12} \tag{A.9}$$

$$f_2 = k_2 + k_3 = k_{22} \tag{A.10}$$

$$f_3 = -k_3 = k_{32} \tag{A.11}$$

Finally, $x_3 = 1$ and $x_1 = x_2 = 0$. Then:

$$f_1 = 0 = k_{13} \tag{A.12}$$

$$f_2 = -k_3 = k_{23} \tag{A.13}$$

$$f_3 = k_3 = k_{33} \tag{A.14}$$

The stiffness matrix can now be written as

$$K = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} = \begin{bmatrix} k_1 + k_2 & -k_2 & 0 \\ -k_2 & k_2 + k_3 & -k_3 \\ 0 & -k_3 & k_3 \end{bmatrix} \tag{A.15}$$

and its equation of motion becomes

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{Bmatrix} \ddot{u}_1 \\ \ddot{u}_2 \\ \ddot{u}_3 \end{Bmatrix} + \begin{bmatrix} k_1 + k_2 & -k_2 & 0 \\ -k_2 & k_2 + k_3 & -k_3 \\ 0 & -k_3 & k_3 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \end{Bmatrix}. \tag{A.16}$$

As explained, Maxwell's Reciprocity Theorem states that the displacement of m_1 (u_1) caused by force F applied at m_2 is the same as the displacement of m_2 (u_2) caused by the force F applied at m_1. For simplicity, both displacements will be calculated under cuasi-static conditions so $\ddot{x}$ may be neglected. The equation of motion (A.16) is reduced to

$$\begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \end{Bmatrix}. \tag{A.17}$$

The displacement of m_1 caused by the force at m_2 is

$$\begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ F \\ 0 \end{Bmatrix} \Rightarrow F = u_1 k_{21} + u_2 k_{22} + u_3 k_{23} \Rightarrow u_1 = \frac{F - u_2 k_{22} - u_3 k_{23}}{k_{21}}. \tag{A.18}$$

It can be seen from the figure that when a force is applied at m_2,

$$u_2 = u_3 \tag{A.19}$$

and

$$u_2 = F \frac{k_1 + k_2}{k_1 k_2} \tag{A.20}$$

so

$$u_1 = F \frac{k_1 - \frac{k_1 + k_2}{k_2}\left(k_{22} + k_{23}\right)}{k_1 k_{21}}. \tag{A.21}$$

But it was previously determined that $k_{22} = k_2 + k_3$ and $k_{23} = -k_3$ so

$$u_1 = -F \frac{k_2}{k_1 k_{21}}. \tag{A.22}$$

The displacement of m_2 caused by the same force at m_1 is

$$\begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} F \\ 0 \\ 0 \end{Bmatrix} \Rightarrow F = u_1 k_{11} + u_2 k_{12} + u_3 k_{13} \Rightarrow u_2 = \frac{F - u_1 k_{11} - u_3 k_{13}}{k_{12}}. \tag{A.23}$$

It can be seen from the figure that when a force is applied at m_1, $u_1 = u_2 = u_3$ and $u_1 = \frac{F}{k_1}$, so

$$u_2 = F \frac{k_1 - k_{11} + k_{13}}{k_1 k_{12}}. \tag{A.24}$$

It was determined that $k_{11} = k_1 + k_2$ and $k_{13} = 0$ so

$$u_2 = -F \frac{k_2}{k_1 k_{12}} . \tag{A.25}$$

If both displacements u_1 and u_2 in (A.22) and (A.24) are equal when the force F is applied to m_2 and m_1 respectively, then k_{12} must be equal to k_{21}.

APPENDIX B – PROPERTIES OF DETERMINANTS

The determinant of a matrix A,

$$A = \begin{pmatrix} a_1 & a_2 & \cdots & a_n \\ b_1 & b_2 & \cdots & b_n \\ \vdots & \vdots & \ddots & \vdots \\ z_1 & z_2 & \cdots & z_n \end{pmatrix} \tag{B.1}$$

is commonly denoted $\det(A)$ or $|A|$.

A 2×2 determinant is defined to be

$$\det\begin{pmatrix} a & b \\ c & d \end{pmatrix} \equiv \begin{vmatrix} a & b \\ c & d \end{vmatrix} \equiv ad - bc\,. \tag{B.2}$$

A $n \times n$ determinant can be expanded by minors to obtain

$$\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix} = a_{11}\begin{vmatrix} a_{22} & \cdots & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{n2} & \cdots & a_{nn} \end{vmatrix} - a_{12}\begin{vmatrix} a_{21} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n3} & \cdots & a_{nn} \end{vmatrix} + \ldots \pm a_{1n}\begin{vmatrix} a_{21} & \cdots & a_{2(n-1)} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{n(n-1)} \end{vmatrix}. \tag{B.3}$$

If α is a constant and A an $n \times n$ square matrix, then

$$|\alpha A| = \alpha^n |A|\,. \tag{B.4}$$

Given an $n \times n$ determinant, the additive inverse is

$$|-A| = (-1)^n |A|\,. \tag{B.5}$$

This means that the determinant of a matrix inverse can be found as follows:

$$|I| = |AA^{-1}| = |A||A^{-1}| = 1\,. \tag{B.6}$$

Where I is the identity matrix, so

$$|A| = \frac{1}{|A^{-1}|}\,. \tag{B.7}$$

The determinant of a matrix transpose equals the determinant of the original matrix,

$$|A| = |A^T|\,. \tag{B.8}$$

Important properties of the determinant include the following, which include invariance under elementary row and column operations:

- Switching two rows or columns changes the sign.
- Scalars can be factored out from rows and columns.
- Multiples of rows and columns can be added together without changing the determinant's value.
- Scalar multiplication of a row by a constant c multiplies the determinant by c.
- A determinant with a row or column of zeros has value 0.
- Any determinant with two rows or columns equal has value 0.

APPENDIX C – MATRIX CONDENSATION

Condensation by Symmetric Gauss Elimination

Although in principle the static condensation of a stiffness matrix K_{ii} pursues the same objective as the Gauss-Jordan elimination process, computer programmes implement it in a different way, i.e., the equations of the substructure are not actually rearranged, and the explicit calculation of the inverse of K_{ii} is avoided. Therefore, this method has an extra advantage since the stiffness matrix of an unconstrained body is singular. The procedure may be in fact coded as a variant of symmetric Gauss elimination, in which one by one, the columns and rows representing the internal nodes are eliminated from the matrix.

Given the stiffness matrix K, the node p represents an internal node. Therefore, p is also the index of row and column representing this node, which will be eliminated. The resultant matrix has n-1×n-1 elements,

$$\begin{bmatrix} k_{1,1} & k_{1,2} & \cdots & k_{1,n} \\ k_{2,1} & k_{2,2} & \cdots & k_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ k_{n,1} & k_{n,2} & \cdots & k_{n,n} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \rightarrow \begin{bmatrix} k'_{1,1} & k'_{1,2} & \cdots & k'_{1,n-1} \\ k'_{2,1} & k'_{2,2} & \cdots & k'_{2,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ k'_{n-1,1} & k'_{n-1,2} & \cdots & k'_{n-1,n-1} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \end{bmatrix} = \begin{bmatrix} f'_1 \\ f'_2 \\ \vdots \\ f'_{n-1} \end{bmatrix} \tag{C.1}$$

where

$$k'_{ij} = k_{ij} - \frac{k_{pj} \cdot k_{ip}}{k_{pp}} \tag{C.2}$$

and

$$f'_i = f_i - \frac{f_p \cdot k_{ip}}{k_{pp}} \;\; \forall i, j \neq p \,. \tag{C.3}$$

For example, let be considered the following stiffness equations of a component with 4 nodes and only one degree of freedom each (i.e. translation in x direction):

$$\begin{bmatrix} 6 & -2 & -1 & -3 \\ -2 & 5 & -2 & -1 \\ -1 & -2 & 7 & -4 \\ -3 & -1 & -4 & 8 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \\ 4 \\ 0 \end{bmatrix} \tag{C.4}$$

Let the two nodes u_3 and u_4 be classified as interior and are to be statically condensed out. To eliminate u_3, perform symmetric Gauss elimination of the third row and column:

$$\begin{bmatrix} 6 & -2 & & -3 \\ -2 & 5 & & -1 \\ & & \bigcirc & \\ -3 & -1 & & 8 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \\ u_4 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \\ \\ 0 \end{bmatrix} \tag{C.5}$$

which results in

$$\begin{bmatrix} 6-\frac{(-1)\cdot(-1)}{7} & -2-\frac{(-2)\cdot(-1)}{7} & -3-\frac{(-4)\cdot(-1)}{7} \\ -2-\frac{(-1)\cdot(-2)}{7} & 5-\frac{(-2)\cdot(-2)}{7} & -1-\frac{(-4)\cdot(-2)}{7} \\ -3-\frac{(-1)\cdot(-4)}{7} & -1-\frac{(-2)\cdot(-4)}{7} & 8-\frac{(-4)\cdot(-4)}{7} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_4 \end{bmatrix} = \begin{bmatrix} 3-\frac{4\cdot(-1)}{7} \\ 6-\frac{4\cdot(-2)}{7} \\ 0-\frac{4\cdot(-4)}{7} \end{bmatrix}. \tag{C.6}$$

That is

$$\begin{bmatrix} \frac{41}{7} & -\frac{16}{7} & -\frac{25}{7} \\ -\frac{16}{7} & \frac{31}{7} & -\frac{15}{7} \\ -\frac{25}{7} & -\frac{15}{7} & \frac{40}{7} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} \frac{25}{7} \\ \frac{50}{7} \\ \frac{16}{7} \end{bmatrix} \tag{C.7}$$

so

$$\begin{bmatrix} 41 & -16 & -25 \\ -16 & 31 & -15 \\ -25 & -15 & 40 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 25 \\ 50 \\ 16 \end{bmatrix}. \tag{C.8}$$

Repeating the process for the third row and column to eliminate u_4 so

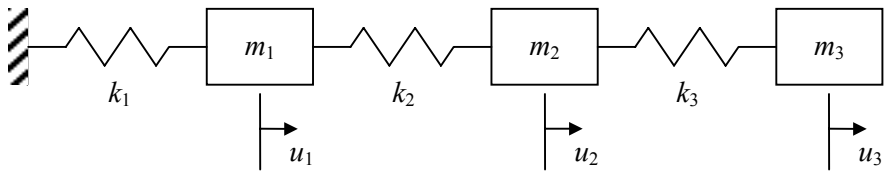
$$\begin{bmatrix} 203 & -203 \\ -203 & 203 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 280 \\ 448 \end{bmatrix}. \tag{C.9}$$

(C.9) are the condensed stiffness equations for the superelement, where only the boundary nodes appear. Obviously this procedure is much simpler than going through the explicit matrix inverse. Another important advantage of Gauss elimination is that equation rearrangement is not required even if the condensed degrees of freedom do not appear sequentially. For example, let be supposed that the assembled substructure contains originally eight DOF and that the ones to be condensed out are numbered 1, 4, 5, 6 and 8. Then Gauss elimination is carried out over those equations only, and the condensed 3×3 matrix and 3×1 force vector are extracted from original rows and columns 2, 3 and 7.

The symmetric Gauss elimination procedure is primarily useful for macroelements and mesh units, since the number of stiffness equations typically does not exceed a few hundreds, which permits the use of full matrix storage. For substructures containing thousands or millions of degrees of freedom,

such as in an aeroplane model, the elimination is carried out using more sophisticated sparse matrix algorithms; for example that described in **[2.2]**.

Example of application:



For the system showed above there are 3 dof, and its stiffness matrix has been already determined as

$$K = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} = \begin{bmatrix} k_1 + k_2 & -k_2 & 0 \\ -k_2 & k_2 + k_3 & -k_3 \\ 0 & -k_3 & k_3 \end{bmatrix}$$

For the proposed system, the equivalent stiffness k_e coefficient when a force is applied at m_3 is

$$k_e = \frac{k_1 \cdot k_2 \cdot k_3}{k_1 k_2 + k_1 k_3 + k_2 k_3}$$

Note that the stiffness matrix is not singular; instead, $\det(K) = k_1 \cdot k_2 \cdot k_3$. This is because there is an external constraint applied to the system, given by the link between m_1 and the wall. Performing Gauss elimination to nodes 1 and 2 sequentially over the stiffness matrix,

$$\begin{bmatrix} \text{[illegible]} & \text{[illegible]} & \text{[illegible]} \\ \text{[illegible]} & k_2 + k_3 & -k_3 \\ \text{[illegible]} & -k_3 & k_3 \end{bmatrix} \Rightarrow \begin{bmatrix} k_2 + k_3 - \dfrac{(-k_2)\cdot(-k_2)}{k_1 + k_2} & -k_3 - \dfrac{0\cdot(-k_2)}{k_1 + k_2} \\ -k_3 - \dfrac{(-k_2)\cdot 0}{k_1 + k_2} & k_3 - \dfrac{0 \cdot 0}{k_1 + k_2} \end{bmatrix} =$$

$$= \begin{bmatrix} \text{[illegible]} & \text{[illegible]} \\ \text{[illegible]} & k_3 \end{bmatrix} \Rightarrow \begin{bmatrix} k_3 - \dfrac{(-k_3)\cdot(-k_3)}{\dfrac{k_1 \cdot k_2 + k_1 \cdot k_3 + k_2 \cdot k_3}{k_1 + k_2}} \end{bmatrix} = \begin{bmatrix} \dfrac{k_1 \cdot k_2 \cdot k_3}{k_1 k_2 + k_1 k_3 + k_2 k_3} \end{bmatrix}$$

The last result is k_e as it was known from a system of springs arranged in series.

BIBLIOGRAPHY

[1.1] ***Fundamentals of Vehicle Dynamics***, Chapter 7: Suspensions – Thomas D. Gillespie – Society of Automotive Engineers, Inc.

[1.2] ***Fundamentals of Motor Vehicle Technology, 4th edition***, Chapter 50: Camber, castor and swivel-axis inclination – V.A.W. Hillier – Stanley Thornes (Publishers) Ltd., September 2004

[1.3] ***Pointed the right way*** – John Hagerman – Grassroots Motorsports Magazine, January 2001

[2.1] ***Concepts and Applications of Finite Element Analysis***, Chapter 2: The Stiffness Method and the Plane Truss – Robert D. Cook – Wiley, 2001

[2.2] ***Introduction to Finite Element Methods***, Chapter 2: The Stiffness Method: Breakdown; Chapter 11: Superelements and Global-Local Analysis – Carlos A. Felippa – Aerospace Engineering Sciences Department of the University of Colorado, 2004

[2.3] ***Theory of Vibrations with Applications***, Chapter 2: Free Vibration, pages 18-23; Chapter 5: Systems with Two or More Degrees of Freedom, pages 130-133; Chapter 6: Properties of Vibrating Systems, pages 175-176, Reciprocity Theorem; Chapter 10: Introduction to the Finite Element Method, pages 301-305 – William T. Thomson – Chapman & Hall, 1993

[2.4] ***Determinant***, Eric W. Weisstein, Mathworld, A Wolfram Web Resource http://mathworld.wolfram.com/Determinant.html

[4.1] ***Multidisciplinary non-linear optimisation with Optimising Structural Language optiSLang*** – Christian Bucher*, Johannes Will, Jörg Riedel* – DYNARDO GmbH, Weimar, Germany, 2003 – * Bauhaus Universität Weimar, Weimar, Germany

[4.2] **Altair HyperStudy v7.0-100** – Help topic "*Optimisation*" – Altair Engineering Inc. 2005

[4.3] ***On the robustness of a simple domain reduction scheme for simulation-based optimisation*** – N. Stander, K. Craig – Int. Journal for Computer-Aided Engineering and Software, 2002

[4.4] ***Evolutionary Algorithms in Theory and Practice***, p. 137 – Thomas Bäck – Oxford University Press, January 1996

[4.5] ***An Evolutionary Algorithm for Multiobjective Optimisation: The Strength Pareto Approach*** – E. Zitzler, L. Thiele – Technical Report 43, Computer Engineering and Communication Networks Lab (TIK) – Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, May 1998

[5.1] ***A Study into Ant Colony Optimisation, Evolutionary Computation and Constraint Programming on Binary Constraint Satisfaction Problems*** – J.I. van Hemert, C. Solnon – Report SEN-E0408 August 2004

[5.2] ***Evolutionary Computing and Autonomic Computing: Shared Problems, Shared Solutions?*** – A.E. Eiben – Vrije Universiteit Amsterdam, April 2005

[5.3] ***Genetic Algorithms Digest*** – David E. Goldberg, University of Alabama – Volume 4, Issue 4, February 1990

[5.4] ***Genetic Algorithms in Search, Optimisation and Machine Learning*** – David E. Goldberg, University of Alabama – Addison-Wesley Publishing Company, Inc., 1989

[5.5] ***Error-detecting and error-correcting codes*** – Richard W. Hamming – Bell System Technical Journal 29(2): 147-160, 1950

[5.6] ***Genetic Algorithms: A Survey*** – M. Srinivas & L. Patniak – IEEE Computer, June 1994

[5.7] ***Artificial Life. Where Biology Meets Computer Science*** – Jean-Philippe Rennard – Grenoble Graduate School of Business, 2001

[6.1] ***Bisection*** – Eric W. Weisstein – Mathworld, A Wolfram Web Resource http://mathworld.wolfram.com/Bisection.html

[7.1] ***Genetic Algorithms Digest*** – Philip Resnik, US Navy – Volume 3: Issue 4, January 1989

[12.1] ***Topology Optimization – Theory, Methods and Applications***, second edition – M. P. Bendsøe, O. Sigmund – Springer-Verlag Berlin Heidelberg, 2003 and 2004

[12.2] ***Topology Optimisation of an Engine Mount Bracket using Altair OptiStruct*** – www.altair.de/pdf/software/Engine-Mount-Bracket-Design.pdf