UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Ingegneria Industriale DII

Corso di Laurea Magistrale in Ingegneria Meccanica

Experimental and numerical investigation for in-depth characterisation of impact-dynamic properties in third generation Advanced High-Strength Steels

Relatore: Prof. Mauro Ricotta

Correlatori: Prof. Patricia Verleysen, Ing. Antonio Pontillo

Department of Electromechanical, Systems and Metal Engineering, Ghent University

Laureando: Claudio Lonardi

Anno Accademico 2020/2021

A Carla, mia nonna

Sommario

I materiali usati nell’industria automobilistica devono soddisfare requisiti sempre più stringenti, come la leggerezza, la resistenza all’urto e la duttilità. Una valida soluzione risiede nei 3rd generation Advanced High-Strength Steels (AHSS), molti dei quali godono del TRIP effect, per cui l’austenite residua, normalmente metastabile a temperatura ambiente, si trasforma in martensite durante la deformazione plastica del materiale, migliorandone le proprietà meccaniche. Sebbene la relazione tra la stabilità dell’austenite residua e lo stato di tensione sia stata ampiamente analizzata in condizioni quasi-statiche, pochi studi dinamici esistono a riguardo, motivo dell’origine del progetto europeo *Dynaustab*.

Il lavoro qui presentato è parte di tale progetto ed è stato svolto presso la Ghent University all’interno del programma Erasmus+. In particolare, esso è composto da una parte sperimentale e da una parte numerica, entrambe mirate a migliorare le qualità degli esperimenti dinamici, realizzati tramite il principio della Split-Hopkinson Bar. In prove di questo tipo, il provino è vincolato tra due lunghe barre di alluminio e, a causa di un impatto esterno contro una delle due, è percorso da un’onda elastica. Attraverso un sistema di estensimetri montati sulle barre, è possibile risalire allo stato di tensione e di deformazione imposto al provino.

Poiché si osservano discordanze tra le misure dei diversi estensimetri, lo scopo della parte sperimentale del lavoro è la calibrazione diretta di tali strumenti di misura, prendendo a riferimento una cella di carico. I risultati mostrano come tutti gli estensimetri, resistivi e a semiconduttore, sottostimino le misure di circa il 5 % e 22 %, rispettivamente. Per gli estensimetri resistivi, i motivi sono da ricercare nel possibile disallineamento tra la direzione di misura e l’asse longitudinale della barra, mentre per quelli a semiconduttore nella non-linearità del ponte di Wheatstone, che a riposo risulta non bilanciato.

La parte numerica del lavoro ha lo scopo di ottimizzare una particolare geometria del provino, variando due parametri dimensionali della regione centrale: la larghezza e il raggio di raccordo. Lo scopo è quello di ottenere alti valori del fattore di triassialità, i più costanti possibile durante la prova dinamica e nella regione centrale del provino. I risultati dell’ottimizzazione mostrano come la geometria ottimale sia quella caratterizzata da alti valori della larghezza e bassi valori del raggio di raccordo.

Abstract

Materials used in the automotive industry must meet increasingly stringent requirements, such as lightness, crashworthiness and ductility. A viable solution is represented by 3rd generation Advanced High-Strength Steels (AHSS), many of which are characterised by the TRIP effect, whereby retained austenite, typically metastable at room temperature, transforms into martensite during plastic deformation of the material, improving its mechanical properties. Although the relationship between the stability of retained austenite and stress state has been widely investigated in quasi-static conditions, few dynamic counterparts exist, which lies at the origin of the European project *Dynaustab*.

The work here presented is part of this project and has been developed at Ghent University within the Erasmus+ programme. In particular, it is composed of two parts, one experimental and the other numerical, both aimed at improving the quality of dynamic tests, performed through the Split-Hopkinson Bar principle. In such experiments, the sample is sandwiched between two long aluminium bars. Due to an external impact against one of the two bars, an elastic wave propagates through it. By means of a system of strain gauges mounted on the bars, it is possible to evaluate the stress and deformation state imposed to the sample.

Since discrepancies are observed between the measurements of the different strain gauges, the experimental part consists in the direct calibration of these measuring instruments, by considering a load cell as reference. The results show that all strain gauges, both resistive and semiconductor, underestimate the measurements by approximately 5 % and 22 %, respectively. For the resistive strain gauges, the reasons are to be found in the possible misalignment between their measuring direction and the longitudinal axis of the bar, while for the semiconductor ones in the non-linearity of Wheatstone bridge, which is not balanced under unstrained conditions.

The numerical part of the work aims to optimize a particular sample geometry by changing two dimensional parameters of the central region: the width and the fillet radius. The purpose is to obtain high values of stress triaxiality, as constant as possible during the dynamic test and in the central region of the sample. The optimization results show that the optimal geometry is characterised by high values of width and low values of fillet radius.

Table of contents

1. Introduction

The choice of materials in the automotive industry is driven by increasingly stringent requirements. In particular, the body in white material must be lightweight to minimize carbon dioxide emissions, while at the same time guaranteeing excellent crashworthiness, necessary for passengers' safety, and high ductility, so that complicated design can be obtained during production. A viable solution is represented by 3rd generation Advanced High-Strength Steels (AHSS), many of which are characterised by Transformation-Induced Plasticity (TRIP) effect, consisting in the transformation of retained austenite to martensite during plastic deformation of the material. The resulting excellent mechanical properties are influenced by the stability of retained austenite, which is in turn dependent on stress and deformation state. Although these relationships have been widely investigated for quasi-static conditions, few dynamic counterparts exist, which lies at the origin of the European project *Dynaustab*.

A fundamental part of this program is carried out at Ghent University and consists in the analysis of the dependence between the stability of retained austenite and stress triaxiality, a dimensionless parameter expressing the relative degree of hydrostatic stress in a given stress state. The dynamic tests, necessary to investigate this dependence, are performed through the Split-Hopkinson Tensile Bar (SHTB) present in the *DyMaLab* laboratory of Ghent University. In this apparatus, the sample is sandwiched between two long aluminium bars, called "input" and "output" bar, respectively. When the input bar is subjected to an external impact, an elastic wave propagates through it reaching first the sample and then the output bar. By means of strain gauges mounted on both bars, it is possible to measure the amplitude of strain waves travelling through them and, by applying one-dimensional wave propagation theory, to evaluate the stress state of the sample. The main issue observed in such measurements lies in discrepancies between the values read by the different types of strain gauges, resistive and semiconductor, which are mounted on the output bar. The first part of the work here presented aims to investigate, through an experimental study, the reasons of such discrepancies.

Another aspect to consider during dynamic tests is the value of stress triaxiality. In fact, in order to obtain a precise correlation between the stability of retained austenite and this parameter, the latter must remain as constant as possible during the test and in a specific region at the centre of the sample, where the microstructure is analysed after the experiment by means of X-ray diffraction (XRD) techniques. In addition, another requirement is to have values of stress triaxiality as high as possible. Since the value of this parameter is influenced by sample geometry, a geometric optimization is a fast and efficient way to meet the mentioned requirements, which is the reason why the second part of the work is numerical in nature.

In the second chapter of the thesis, an overview of the materials used in the automotive industry is provided, focusing in particular on their evolution over the years and the requirements they must meet. Subsequently, the main reasons for the birth of *Dynaustab* project are listed. In the third chapter, the experimental apparatus of *DyMaLab* laboratory is described and the fundamentals of one-dimensional wave propagation theory are reported. In the fourth chapter, the experimental work of strain gauges calibration is presented. The mechanical system designed for this purpose and the procedures followed in the test are described in detail. Then, the obtained results are discussed. In the fifth chapter, the numerical work of sample geometric optimization, performed in the finite element analysis software Abaqus FEA, is described. After a thorough explanation of the way the scripts developed in MATLAB and Python work, the optimal geometry is presented and discussed. Finally, in the sixth chapter, the conclusions of the works carried out are reported and some insights for possible future developments are given.

2. Overview of automotive materials

2.1 Material requirements for car body in white (BIW)

The material composing the body in white (BIW) of a car must satisfy several requirements.

The first one regards the safety, which means that during a crash the material has to be strong enough to absorb the impact energy and not to crumple, so that it can protect the passengers. As second requirement, its formability must be high in order to enable the more complicated designs during the production process. Lastly, the third requirement is environmental. Nowadays, due to the wide exploitation of nature carried out by humans, the air pollution is getting more and more severe, amplifying the so-called "greenhouse effect": due to some gas emissions, the thermal energy coming from the solar radiation is partly trapped in the Earth's atmosphere, causing a global warming that is harmful to our planet. Since one of the main substances responsible for this phenomenon is carbon dioxide, highly released by means of transport, the world government authorities have been trying for many years to reduce their average emissions through increasingly stricter regulations. The latest one issued by the European Parliament states that starting from 2021 the average emissions of all the cars in UE shall not exceed 95 g/km [1], which corresponds to a reduction of 27% with respect to the amount allowed in the previous regulation [2].

It should be noted that this requirement is amplified by the increasing development of pure and hybrid electric vehicles, characterised by well to wheel emissions of carbon dioxide lower than those of thermal vehicles. Hence, in order to compete against the former, the latter need to reduce the carbon dioxide emissions as much as possible [3]. Since in a car the fuel consumption, therefore the amount of the gas emissions, are proportional to its mass, its material has to be light enough. Several studies prove how a 10% reduction of the car mass brings to a 6 ÷ 8% reduction of fuel consumption and consequently of carbon dioxide emissions [4]. Other requirements, definitely not least, are the possibility of large-scale production, the weldability and the corrosion resistance, each of them achievable through a reasonable cost.

It is clear that many of all aforementioned requirements are conflicting, which implies a constant research for an optimal solution as a compromise. For this reason, the automotive industry has always been characterised by an ongoing innovation.

2.2 History

Steel began to be the main material in the automotive industry in the 1920s, fully replacing the wood that previously had been in part used for the BIW [5]. The fundamental characteristic that made it become the automotive leader material was the great combination of high mechanical strength and low cost. Nowadays, it represents about the 65% of the car weight and is the primary component of the BIW. However, over the last hundred years, it has been subjected to a deep evolution aiming to research solutions as optimal as possible that satisfied the requirements mentioned in §2.1, in particular the most conflicting ones: the high mechanical strength, necessary for passengers' safety, and the great formability, allowing complicated design. For this reason, in Figure 2.1Figure 2.1 the main steel types, which have been used over the years for the BIW and will be presented below, are represented in an ultimate tensile strength (UTS) vs total elongation graph. At first sight, it is easy to observe that all the steels are arranged following a sort of hyperbole, proving the conflicting nature of these two mechanical properties.

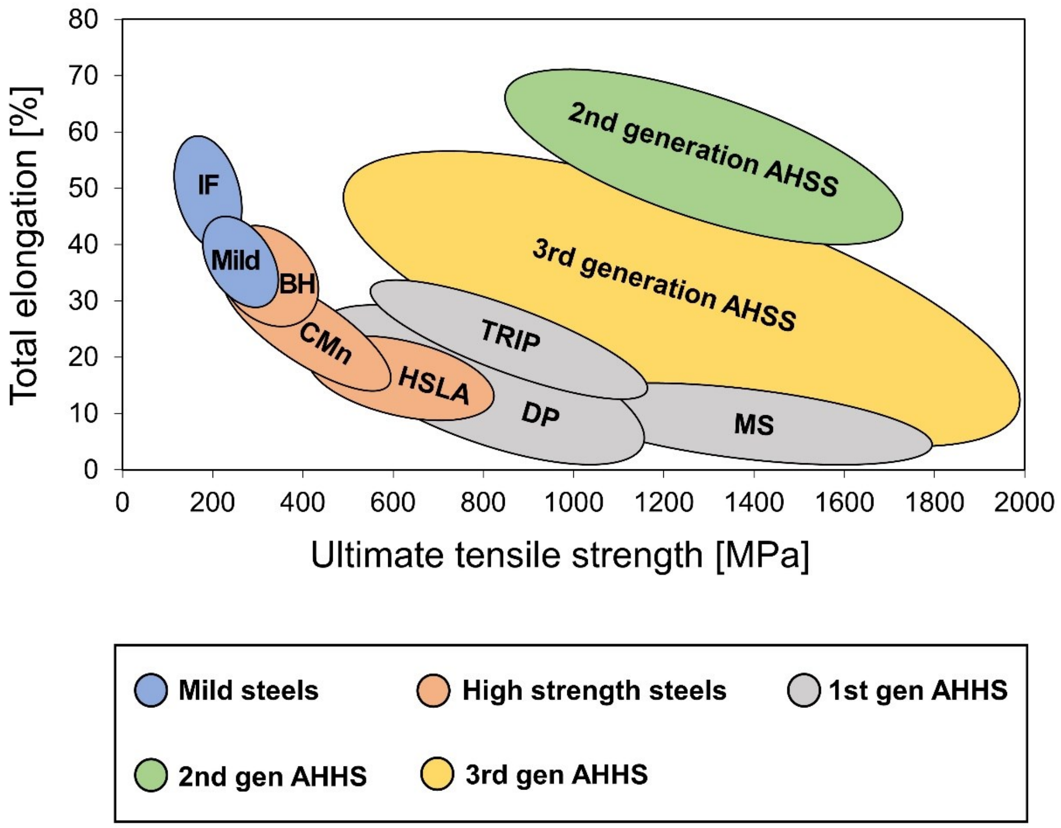


Figure 2.1 *Overview of the automotive steels classes, arranged in an ultimate tensile strength – total elongation graph. The 3rd generation yellow area refers to both current and possible future developments.*

The first special steels developed were the mild steels, also known as low-carbon (LC) steels, first used in the 1980s. They are composed of a single phase, ferrite, with a small carbon content, generally lower than 220 ppm in weight, that can reach even lower values (< 30 ppm in weight) for a particular type, the interstitial free (IF) steels. Although the total elongation of this steel family is relevant (until 60%), its UTS is relatively low (< 280 MPa). To overcome this issue, a new design solution was implemented for the BIW, which started to be fabricated in two parts, each of them satisfying a different requirement: the external one, composed of a thin formable layer, enabled complicated designs, whereas the internal one, not visible and composed of a structure made of beams and pipes, gave a high mechanical strength [6].

Subsequently, the 1990s saw the development of the so-called conventional high strength steels (HSS), in which the UTS was significantly increased (200 ÷ 800 MPa), at the expense of a decrease in the total elongation (10 ÷ 40%). Once again, they are single-phase steels and the strengthening mechanism is achieved by means of solid-solution hardening. This family includes bake hardening (BH), carbon manganese (CMn) and high-strength low-alloy (HSLA) steels.

The next step was to improve, with the same UTS, the total elongation, representing the critical point of HSS. Thus, were born the Advanced High-Strength Steels (AHSS) which, thanks to the ULSAB programme, promoted in 1995 and involving 35 steel companies from 18 different countries, massively replaced the materials previously used for the BIW. These steels have a UTS greater than 500 MPa and a composite-like microstructure. In fact, they consist of two or more phases that synergically give the material better properties than those of the individual phases themselves.

The first to be developed were the 1st generation AHSS, to which belong the Dual-Phase (DP) steels, based on studies carried out few decades before [7]. They are composed of ferrite and martensite, where the former gives formability to the material and the latter, varying from 5 to 50 % in volume, gives the mechanical strength. In this way, it became possible to reach UTS between 550 and 1000 MPa and total elongation slightly higher than that achievable by the HSS, with the same UTS.

Later, developing the studies of Wassermann [8], Transformation-Induced Plasticity (TRIP) steels began to be produced. These steels are based on the so-called “TRIP effect”: when the steel contains a sufficiently metastable austenite and is plastically deformed, the austenite loses the stability and turns into martensite. The austenite stability is meant as its tendency to preserve the chemical composition, in fact, from a thermodynamic point of view, a substance is stable if it is in its lowest energy state, that is in chemical equilibrium with the environment. The formed martensite brings an additional work hardening which raises the UTS (590 ÷ 1180 MPa) and delays necking, thus increasing the total elongation that, with the same UTS, is in fact greater than that of DP steels.

However, since the austenite is normally unstable at room temperature, a special thermal process, reported in Figure 2.2, is necessary.

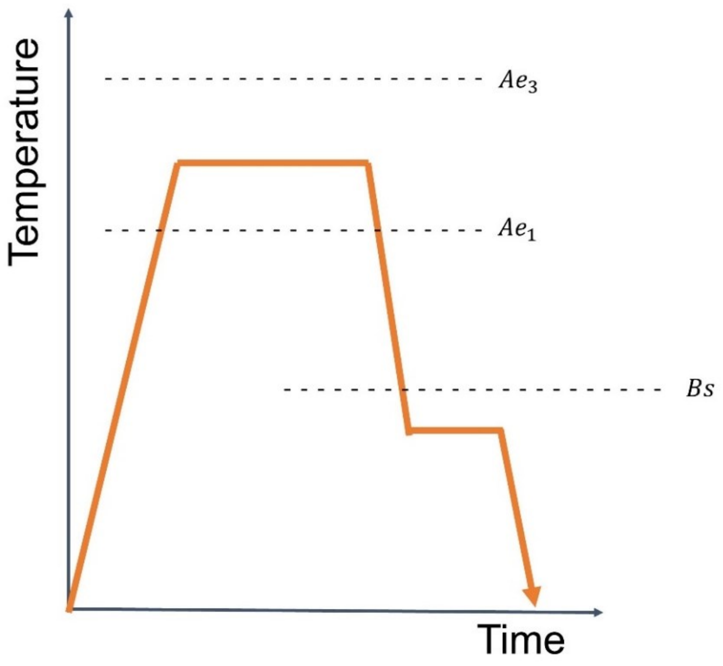


Figure 2.2 *Representation of thermal process adopted for TRIP steels.*

An austenitization at a temperature between Ae_1 and Ae_3, that is within the intercritical range where austenite and ferrite are in equilibrium, produces a microstructure in which the volume amount of each phase is generally 50%. Then, the material is quenched to a temperature below Bs to avoid any perlite precipitation and kept at this temperature, so that the austenite can transform to low-carbon bainite. In this step, thanks to the presence of some alloy elements such as silicon and aluminium, the carbide formation is suppressed and the untransformed austenite experiences a carbon enrichment.It follows that, after the subsequent cooling to room temperature, the austenite is no longer unstable. The final microstructure is typically composed of about 50% of ferrite, 35 to 40% of bainite and 10 to 15% of austenite. However, the volume fraction of retained austenite that can be obtained, so the martensite amount after the transformation, are low due to the limited amount of carbon. This results in a limited work hardening which badly affects especially the total elongation.

Other 1st generation AHSS are the martensitic steels (MS), mostly formed by a martensitic structure surrounded by small amounts of ferrite and bainite. They have the highest UTS (900 ÷ 1700 MPa) of all the automotive steels but, at the same time, a very low total elongation, which is generally a critical issue of all the 1st generation steels.

For this reason, starting in the 2000s, the 2nd generation AHSS began to be developed, still exploiting the TRIP effect like those of the previous generation. The substantial difference, however, lies in the relevant amount of alloy elements such as manganese (up to 30%), which stabilises austenite at room temperature and allows it to reach much higher percentages, even up to 100%. It follows that under plastic deformation a high amount of martensite, proportional to the amount of austenite, is generated, thus giving the material a great mechanical strength; at the same time, the necking is delayed and therefore the total elongation is just as great. As shown in Figure 2.1, the UTS - total elongation combination is the best among all automotive steels, however there are two major limitations, which are low weldability and high cost, both due to the massive amount of expensive alloying elements such as manganese, necessary for the stabilisation of austenite at room temperature.

To overcome the low total elongation of 1st generation AHSS and the high cost of 2nd generation AHSS, in the mid-2000s the 3rd generation AHSS began to be developed. First of all were the Quench and Partitioning (Q&P) steels, based on the approach proposed by Speer et al. in 2003 and shown in Figure 2.3 [9].

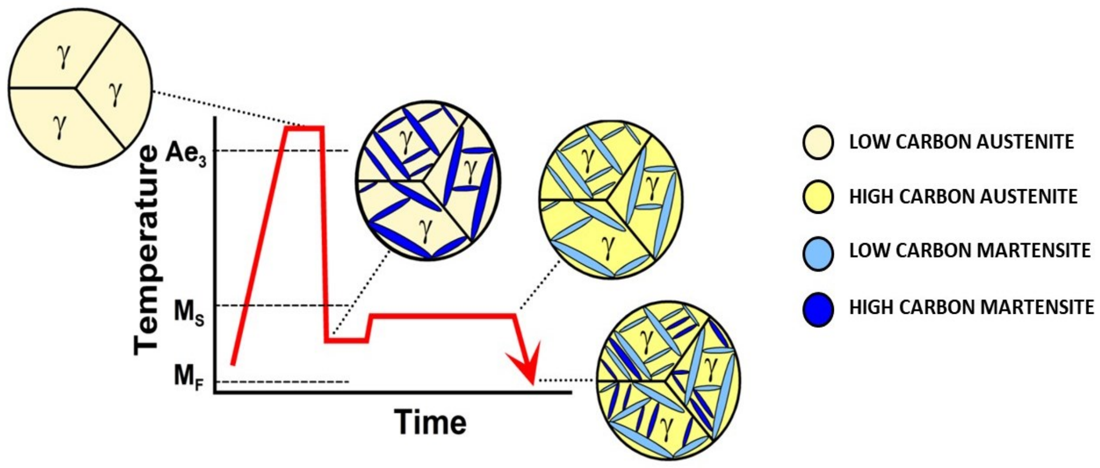


Figure 2.3 *Representation of the quenching and partitioning process with the microstructure evolution.*

A cold rolled thin sheet is austenitized above Ae_3 and then quenched at a temperature between M_s and M_f, so that a controlled portion of austenite turns into martensite. Subsequently, an isothermal process called "partitioning" is performed at a higher temperature and the diffusion of carbon from supersaturated martensite takes place. However, if normally at this temperature bainite would form, some alloying elements such as silicon and molybdenum retard this transformation by directing carbon to austenite. This implies that:

1. Austenite is more stable at room temperature, which enhances the TRIP effect.
2. Martensite is annealed, which improves its damage resistance properties.

Finally, a cooling to room temperature is performed. It is clear how this process avoids stabilising the austenite by a further addition of high-cost alloying elements, such as manganese, or of carbon, which would penalize the weldability. The resulting steels are characterised by values of total elongation up to 20% and UTS between 1000 and 1500 MPa, which makes them much more performing than those of the previous generations.

Another type of 3rd generation AHSS is represented by the medium-Mn steels. The lower amount of manganese (4 ÷ 12 wt%), with respect to the 2nd generation AHSS, makes the stabilisation of austenite at room temperature through alternative ways necessary. This is achieved by a special thermal process, called Austenite-Reverted-Transformation (ART) annealing, that is an intercritical annealing between Ae_1 and Ae_3, shown in Figure 2.4 [10]. Before undergoing this heat treatment, the steel is usually composed of ferrite, austenite and martensite thanks to previous processes. During ART annealing, by means of the reversed transformation from martensite to austenite, the latter is enriched in carbon and manganese, which provide its stabilisation at room temperature. It has been proved that the resultant mechanical strength and total elongation can reach 1600 MPa and 30% respectively.

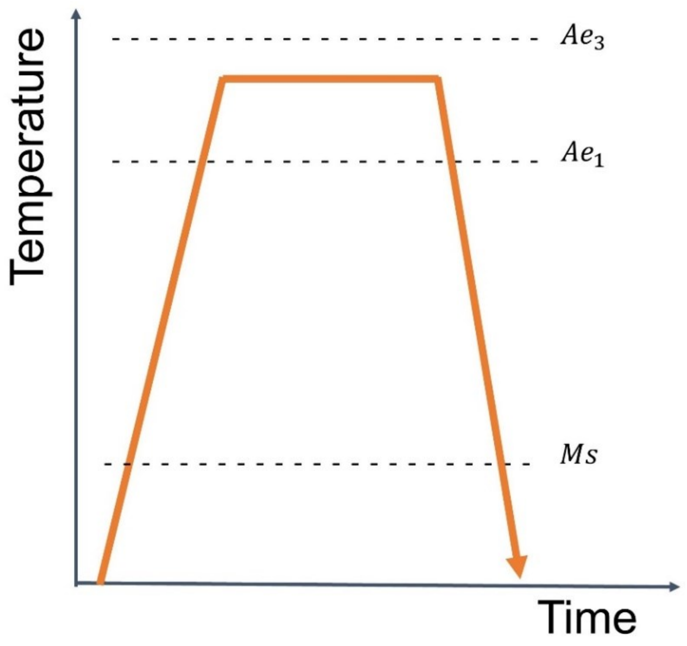


Figure 2.4 *Representation of ART annealing process used for medium-Mn steels.*

Over the last 20 years, thanks to great improvements in the research, the materials previously used for BIW have been almost completely replaced by AHSS, in particular by the 3rd generation ones, as shown in Figure 2.5, representing the BIW of the Volvo V60 in 2011.

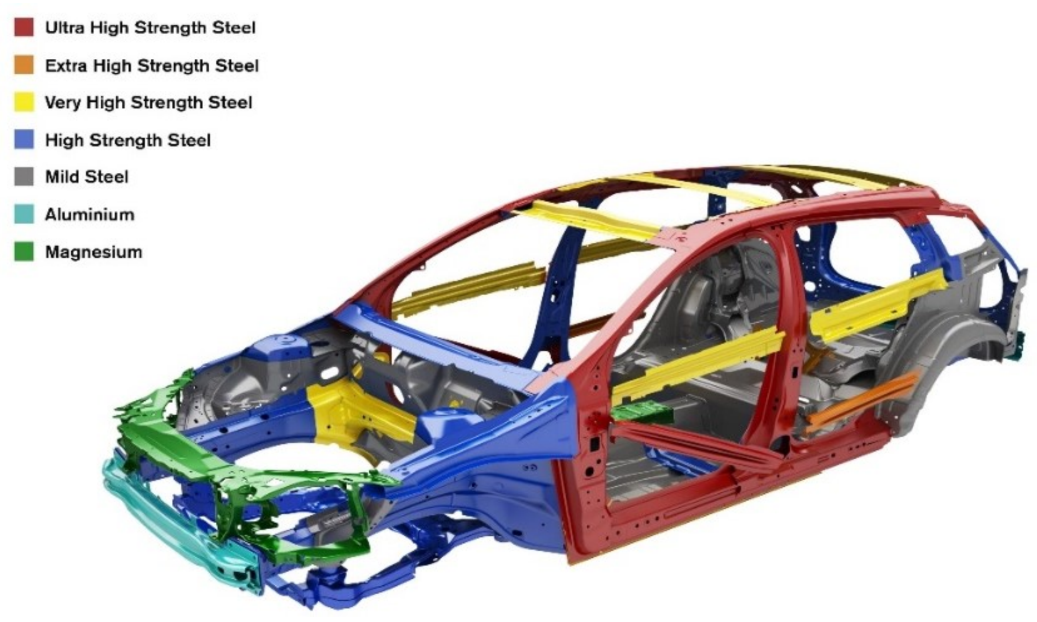

🟥 Ultra High Strength Steel
🟧 Extra High Strength Steel
🟨 Very High Strength Steel
🟦 High Strength Steel
⬛ Mild Steel
🟦 Aluminium
🟩 Magnesium

Figure 2.5 *The body in white composition of the Volvo V60 (2011). Ultra, Extra and Very High-Strength Steels belong to Advanced High-Strength Steel family.*

2.3 State of the art

One of the factors determining the excellent mechanical properties of 3rd generation AHSS is the stability of retained austenite, which is meant, as already mentioned, as its tendency to preserve the chemical composition. In the past years, many studies have been carried out to clarify the factors affecting the austenite stability and the following have been found to be the most relevant ones:

— stress state
— temperature
— strain rate
— chemical composition
— microstructural parameters.

With regard to stress state, since the austenite to martensite transformation is linked to a volume increase, a compression load inhibits the transformation, stabilising the retained austenite.

Jacques et al. investigated this dependence statically testing two different materials, both TRIP steels: High Silicon Steel (HSi), a conventional TRIP-assisted multiphase steel with a silicon content of 1.5 wt%, and Low Silicon Steel (LSi), a TRIP-aided steel with a lower content of silicon and a Dual-Phase-like chemical composition [11]. Some relevant results are shown in the graph of Figure 2.6, reporting the ratio α'/γ_0 of martensite to austenite amount as a function of plastic equivalent strain ε_e^p. The curves are parametric with respect to different values of stress triaxiality T, a dimensionless parameter expressing the relative degree of hydrostatic stress in a given stress state.

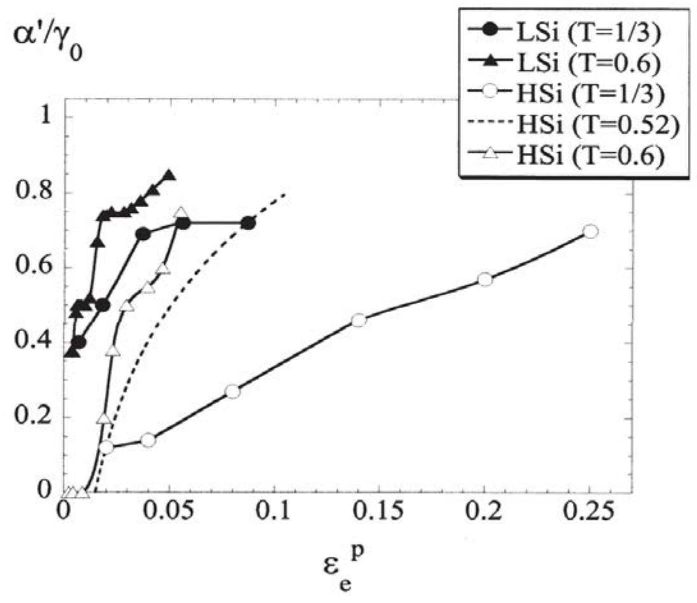


Figure 2.6 *Experimental data reporting the ratio α'/γ_0 of martensite to austenite amount as a function of plastic equivalent strain ε_e^p. The results have been obtained for two different materials (HSi and LSi) and for different values of stress triaxiality T [11].*

It is possible to observe that both steels exhibit the same trend, that is with increasing stress triaxiality the martensitic transformation rate increases, thus decreasing the austenite stability. Moreover, the effect of triaxiality T on α'/γ_0 is more relevant in the transformation of the last 50% of retained austenite. Finally, the whole transformation occurs in the first 10% of effective plastic strain ε_e^p, except for the HSi loaded in uniaxial tension ($T = 1/3$).

Some years later Jacques et al. themselves further investigated the problem by statically testing two other TRIP-steels with different values of stress triaxiality. The results showed that the transformation rate is not, as observed in the previous study, a monotonically increasing function of stress triaxiality because, after a certain value of the latter, the former starts to decrease [12]. This proves that stress triaxiality is not the only loading factor affecting the martensitic transformation rate.

As far as temperature is concerned, Blondé et al. carried out tensile experiments on TRIP-steels at different temperatures, observing an increase of the stability of retained austenite when the material is heated up, as shown from the graph in Figure 2.7 [13].

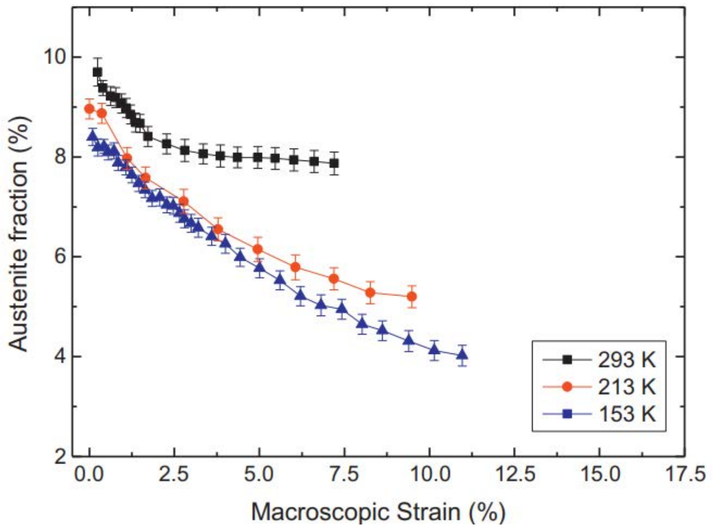


Figure 2.7 *Experimental data reporting the austenite fraction as a function of macroscopic engineering strain. The curves are parametric with respect to temperature [13].*

With regard to the dependence between the stability of retained austenite and strain rate, Zou et al. investigated it by loading in uniaxial tension QP980 steel samples over a wide range of strain rates and temperatures. Precisely, the strain rate range (0.0002 ÷ 175 s^{-1}) was chosen to study the problem both statically and dynamically and, in particular, at strain rates corresponding to real-case scenarios, that are sheet metal forming (2 s^{-1}) and car collision conditions (175 s^{-1}). The results showed how the effect of strain rate on the mechanical stability of retained austenite is non-monotonic. In fact, at low strain rates (0.0002 ÷ 0.1 s^{-1}) the stability of retained austenite increases with increasing strain rate, vice versa for high strain rates (0.1 ÷ 175 s^{-1}), as shown in the graph of Figure 2.8, reporting the volume fraction of retained austenite f_γ as a function of true strain at different strain rates [14].

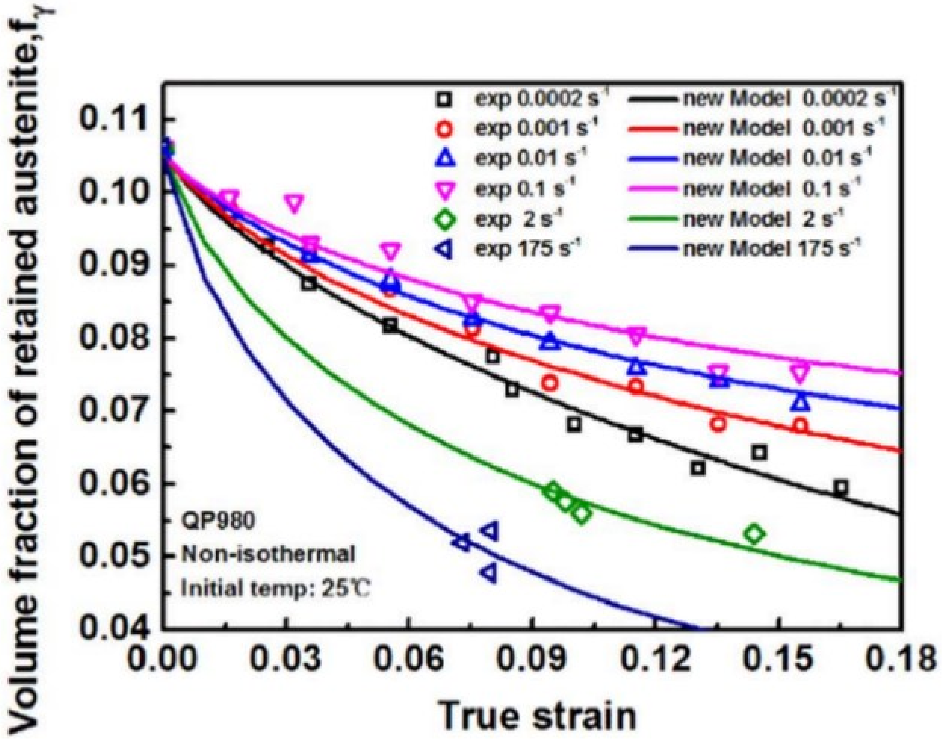


Figure 2.8 *Experimental data representing the volume fraction of retained austenite as a function of true strain at different strain rates. After the measurements, curve fitting, represented by the solid lines, was carried out [14].*

This non-monotonically trend results from the combination of two different effects, both caused by the increase in strain rate:

— Mechanical effect, that is more martensite nucleation sites are generated so the martensitic transformation is accelerated, penalising the stability of austenite.
— Thermal effect, that is temperature increases because of the adiabatic heating and the local heating which is generated by the exothermic martensitic transformation and, as mentioned before, the stability of austenite is improved.

The thermal and the mechanical effect would be dominant respectively at low (0.0002 ÷ 0.1 s^{-1}) and high strain rates (0.1 ÷ 175 s^{-1}), which justifies the trend observed. Similar considerations were carried out by another study [15].

For what mentioned above, it is clear that in addition to the dependence of the retained austenite stability on stress state, temperature and strain rate, it is also necessary to investigate the way these factors interact with each other. Although such studies have been extensively carried out for quasi-static cases, few exist for the dynamic counterpart, corresponding to real-case scenarios like crash impacts, where high strain rates and many different stress states occur.

From these needs, the *Dynaustab* project, of which the current work is part, was born. Funded by the Research Fund for Coal and Steel (RFCS), the project aims to investigate the stability of retained austenite under such conditions.

3. Experimental apparatus

3.1 Fundamentals

The experimental apparatus on which the work presented in chapter §4 is focused is the Split-Hopkinson Tensile Bar (SHTB), located in the *DyMaLab* laboratory of the Faculty of Engineering and Architecture at Ghent University.

The Split-Hopkinson Tensile Bar, from now on referred to simply as "Hopkinson Bar", is an apparatus useful for testing the mechanical response of materials under dynamic conditions, that is at strain rates in the range of $10^2 \div 10^4$ s^{-1}.

Before 1949, when the first Hopkinson Bar was built by Kolksy, who exploited the theoretical principles developed by Hopkinson in 1914, materials were dynamically tested by the direct impact of a hammer against the sample [16], a procedure which presented, however, two main issues:

1. The conditions of the sample were not well controllable.
2. Detailed information about the state of the sample could not be recorded.

The Hopkinson Bar was able to overcome these two issues with an innovative design, schematically illustrated in Figure 3.1, which refers to compression loading.

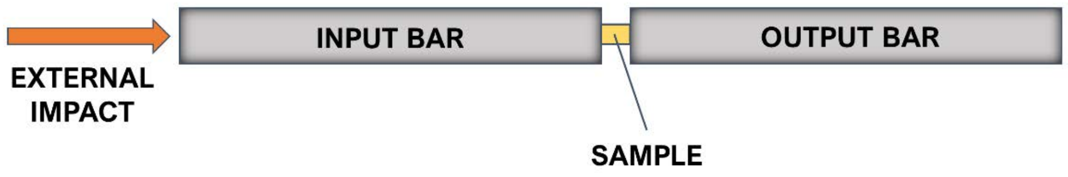


Figure 3.1 *Schematic representation of the Hopkinson Bar loaded in compression.*

Instead of loading the sample by a direct impact, in the Hopkinson Bar the sample is sandwiched between two elastic rods and one of the two is subjected to an external impact. The first bar, between the external impact point and the sample, is called “input bar”, while the other is called “output bar”. After the external impact against the input bar, an elastic wave, called “incident wave”, travels from left to right, with reference to Figure 3.1. Once it reaches the input bar-sample interface, the incident wave is subjected to two different phenomena:

1. A reflection, giving rise to the reflected wave which travels through the input bar back to the external impact point, from right to left with reference to Figure 3.1.
2. A transmission, giving rise to the transmitted wave which crosses the sample, loading it, and propagates through the output bar, from left to right with reference to Figure 3.1.

By means of a suitable instrumentation, presented in §3.2, it is possible to record the amplitude of the strain waves travelling through the input and the output bar and, through them, to derive the loading state of the sample, as explained in §3.3. Therefore, it is clear that, unlike the techniques characterised by a direct impact against the sample, the Hopkinson Bar allows to control and quantify the impact event.

3.2 Components

The Hopkinson Bar is schematically represented in Figure 3.2, which again refers to compression loading.

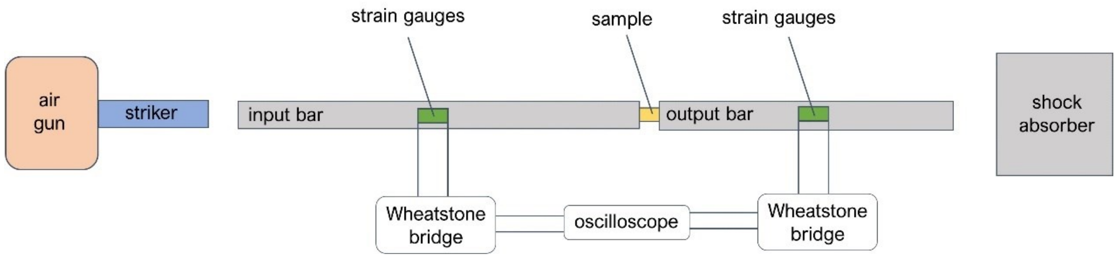


Figure 3.2 *Schematic representation of the Hopkinson Bar present in DyMaLab. The configuration refers to compression loading.*

The whole apparatus consists of the following components.

— Loading device, composed of the air gun and the striker. The latter, partially surrounded by the former, is pushed against the incident bar, under the action of compressed air filling the air gun. The impact velocity, a relevant parameter for the purposes of the experiment, can be controlled by suitably varying the air pressure inside the air gun.

— The input and the output bar. As already discussed in §3.1, the input bar is loaded by the striker and, due to the impact, is crossed by the incident wave. This wave, after reaching the input bar-sample interface, is partly reflected back to the input bar and partly transmitted through the sample and then through the output bar. The input and output bars of *DyMaLab* are 6000 mm and 3000 mm long, respectively, while their diameter is 25 mm. The material they are made of is 5083 aluminium, ensuring that during the experiments these two mechanical components exhibit a linear elastic behaviour, which makes it possible to use simple linear relationships between the recorded strain and stress.

— Shock absorber, whose function is to brake the output bar, which would otherwise continue its motion.

— Measuring system. Four pairs of strain gauges, mounted on the input and output bar, record the strain on them. Then, each of the four Wheatstone bridges amplifies the signal acquired by the pair of strain gauges it is connected to and transfers the signal to an oscilloscope.

The experimental apparatus present in *DyMaLab* is shown in Figure 3.3, where all the aforementioned components are indicated.

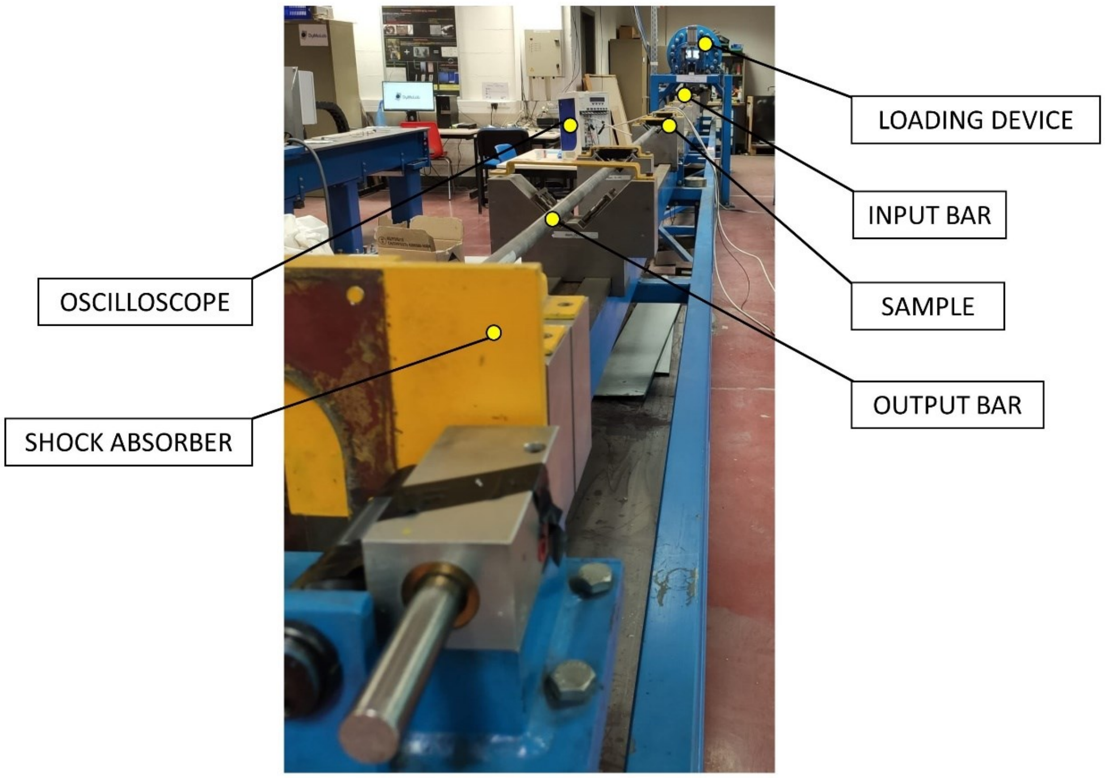


Figure 3.3 *Hopkinson Bar present in DyMaLab.*

3.3 Loading state of the sample

As mentioned in §3.2, the measuring system records the strain on the input and output bars. Since the material of the two bars ensures that during the experiment they exhibit a linear elastic behaviour, the strain to which they are subjected can be traced back to stress through linear relationships. However, it is not immediate to obtain the loading state on the sample from this information.

The step lies in applying the one-dimensional wave propagation theory to the Hopkinson Bar [17]. Several assumptions underlie this theory:

- Homogeneous material.
- Plane, parallel cross-sections remain plane and parallel.
- Uniaxial longitudinal stress state.
- Uniform distribution of stress along the cross-section.
- Absence of any body forces.
- Bar of semi-infinite length.

Referring to Figure 3.4, let $u(x,t)$ be the longitudinal displacement of the bar at the cross-section of coordinate x and at a generic time instant t.

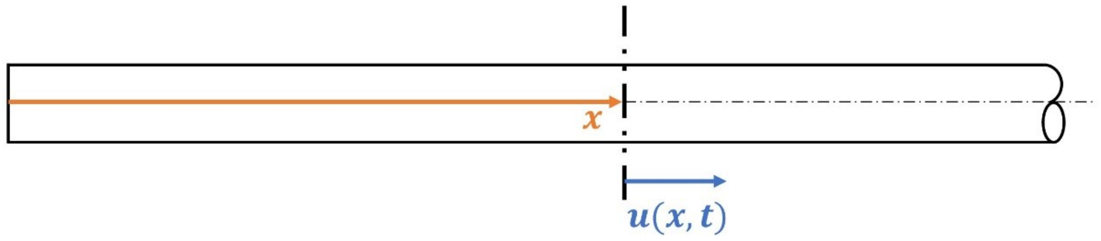


Figure 3.4 *Bar of semi-infinite length with displacement $u(x,t)$ of a generic cross-section.*

Respecting the aforementioned assumptions, it is possible to impose the mechanical equilibrium along x on a generic bar segment of infinitesimal length, obtaining with some mathematical steps the general equation of the waves

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{c_0^2}\frac{\partial^2 u}{\partial t^2}, \tag{3.1}$$

where $c_0 = \sqrt{E/\rho}$ represents the propagation speed of the elastic wave in the bar, E the Young's modulus of the bar material and ρ its density.

The solution $u(x,t)$ of (3.1), in the absence of wave dispersion, is given by d'Alembert formula.

$$u(x,t) = f(x - c_0 t) + g(x + c_0 t) \tag{3.2}$$

$f(x - c_0 t)$ e $g(x + c_0 t)$ represent generic waves propagating, at speed c_0, along the positive and negative x-direction, respectively. Thus, the solution $u(x,t)$ is given by the overlapping of two different waves travelling along opposite directions.

By deriving (3.2), with respect to x, the longitudinal strain $\varepsilon(x,t)$ of the bar can be obtained.

$$\varepsilon(x,t) = \frac{\partial u}{\partial x} = f'(x - c_0 t) + g'(x + c_0 t) \tag{3.3}$$

$f'(x - c_0 t)$ and $g'(x + c_0 t)$ indicate the derivative, with respect to the corresponding entire argument, of $f(x - c_0 t)$ and $g(x + c_0 t)$, respectively.

Then, by deriving (3.2) with respect to t, the velocity $v(x,t)$ of the material particle hit by the wave can be written as follows.

$$v(x,t) = \frac{\partial u}{\partial t} = c_0[-f'(x - c_0 t) + g'(x + c_0 t)] \tag{3.4}$$

In relation to the Hopkinson Bar, it is now possible to obtain (3.2), (3.3) and (3.4) for each of the two bars. With reference to Figure 3.5, let L_s and A_s be the length of the sample and the area of its cross-section, respectively. Moreover, let ε_i, ε_r, ε_t be the amplitudes of the incident, reflected and transmitted strain waves in the bars at the cross-sections 1 and 2, located at the input bar-sample and output bar-sample interafaces, respectively. Finally, let the subscript "1" indicate the input bar quantities corresponding to cross-section 1 and the subscript "2" the output bar quantities corresponding to cross-section 2.

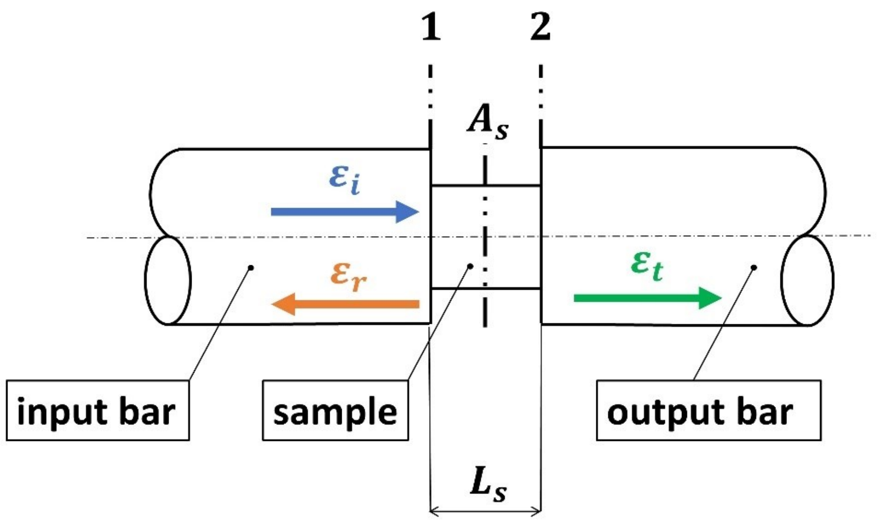


Figure 3.5 *Incident, reflected and transmitted waves at sample-bars interfaces.*

Considering that the functions $f(x - c_0 t)$ e g$(x + c_0 t)$ represent elastic waves which propagate along the positive and negative x-direction, respectively, and calling u_i, u_r and u_t the longitudinal displacements of the bars particles due to elastic waves at the cross-sections 1 and 2, it follows that:

- $f(x - c_0 t) + g(x + c_0 t) = u_i + u_r$ for the input bar at cross-section 1.
- $f(x - c_0 t) + g(x + c_0 t) = u_t + 0$ for the output bar at cross-section 2.

(3.2), (3.3) e (3.4) become, for the two bars:

$$u_1^{INPUT}(x,t) = u_i + u_r, \tag{3.5}$$

$$\varepsilon_1^{INPUT}(x,t) = \varepsilon_i + \varepsilon_r, \tag{3.6}$$

$$v_1^{INPUT}(x,t) = c_0(-\varepsilon_i + \varepsilon_r), \tag{3.7}$$

$$u_2^{OUTPUT}(x,t) = u_t, \tag{3.8}$$

$$\varepsilon_2^{OUTPUT}(x,t) = \varepsilon_t, \tag{3.9}$$

$$v_2^{OUTPUT}(x,t) = c_0(-\varepsilon_t). \tag{3.10}$$

By means of (3.5) ÷ (3.10) it is possible to trace back to the loading state of the sample, in particular to stress, force, strain rate and strain.

3.3.1 Stress

By imposing the mechanical equilibrium on the sample along x, it follows that

$$F_{1,s} = F_{2,s} \tag{3.11}$$

where $F_{1,s}$ and $F_{2,s}$ represent the magnitude of the forces loading the sample at the cross-sections 1 and 2, respectively.

Because of Newton's third law, $F_{1,s}$ and $F_{2,s}$ are also the forces magnitudes on the input and the output bar at the cross-section 1 and 2, respectively. Thus, (3.11) can be rewritten by expressing each of these forces as the product of the stress σ on the bar by the area A of the corresponding cross-section.

$$\sigma_1^{INPUT} \cdot A_1 = \sigma_2^{OUTPUT} \cdot A_2 \tag{3.12}$$

Considering that both bars have equal diameter ($A_1 = A_2 = A_b$) and that their linear elastic behaviour makes possible to express the longitudinal stress σ as the product of the longitudinal strain ε and the Young's modulus E, the following equation is obtained.

$$\varepsilon_1^{INPUT} = \varepsilon_2^{OUTPUT} \tag{3.13}$$

Because of (3.6) and (3.9), it follows that

$$\varepsilon_i + \varepsilon_r = \varepsilon_t, \tag{3.14}$$

which expresses, in terms of strain, the condition of mechanical equilibrium.

The longitudinal stress σ_s on the sample is assumed to be the average of the stresses on it at the cross-sections 1 and 2. With some simple mathematical steps already seen in this chapter, the following equation is obtained.

$$\begin{aligned}\sigma_s &= \frac{1}{2}\cdot\frac{F_{1,s}+F_{2,s}}{A_s} \\ &= \frac{1}{2}\cdot\frac{\sigma_1^{INPUT}\cdot A_1+\sigma_2^{OUTPUT}\cdot A_2}{A_s} \\ &= \frac{E}{2}\cdot\frac{A_b}{A_s}\cdot(\varepsilon_1^{INPUT}+\varepsilon_2^{OUTPUT})\end{aligned} \tag{3.15}$$

Given (3.6), (3.9) and (3.14), the final expression of σ_s results in (3.16).

$$\sigma_s = E\cdot\frac{A_b}{A_s}\cdot\varepsilon_t \tag{3.16}$$

3.3.2 Force

The force F_s loading the sample is nothing but the product between the stress σ_s of (3.16) and the cross-sectional area of the sample A_s.

$$F_s = \sigma_s\cdot A_s = E\cdot A_b\cdot\varepsilon_t \tag{3.17}$$

3.3.3 Strain rate

The expression of strain rate $\dot{\varepsilon_s}$ of the sample can be obtained by deriving the longitudinal strain ε_s with respect to time t .

$$\dot{\varepsilon_s} = \frac{\partial\varepsilon_s}{\partial t} = \frac{\partial}{\partial t}\left(\frac{u_{1,s}-u_{2,s}}{L_s}\right) \tag{3.18}$$

For the congruence condition, the longitudinal displacements of the sample $u_{1,s}$ and $u_{2,s}$ must be the displacements of the input bar at the cross-section 1 and of the output bar at the cross-section 2, respectively. Therefore, (3.18) becomes:

$$\dot{\varepsilon_s} = \frac{\partial}{\partial t}\left(\frac{u_1^{INPUT}-u_2^{OUTPUT}}{L_s}\right) = \frac{v_1^{INPUT}-v_2^{OUTPUT}}{L_s} \tag{3.19}$$

Considering (3.7) and (3.10), it follows that

$$\dot{\varepsilon_s} = \frac{c_0(-\varepsilon_i + \varepsilon_r + \varepsilon_t)}{L_s}. \tag{3.20}$$

Then, by combining (3.20) and (3.14), the final expression of $\dot{\varepsilon_s}$ can be obtained.

$$\dot{\varepsilon_s} = \frac{2c_0}{L_s}\varepsilon_r \tag{3.21}$$

3.3.4 Strain

The longitudinal strain ε_s is nothing but the integral over time of strain rate $\dot{\varepsilon_s}$.

$$\varepsilon_s = \int_0^t \dot{\varepsilon_s}\, dt = \frac{2c_0}{L_s}\int_0^t \varepsilon_r\, dt \tag{3.22}$$

It is therefore clear that through the knowledge of ε_i, ε_r e ε_t, recorded by the strain gauges mounted on the bar, it is possible to come to the loading state on the sample through (3.16), (3.17), (3.21) and (3.22), obtained by applying the one-dimensional wave propagation theory to the two bars. However, in the data processing the length L_s considered in (3.21) and (3.22) is related to the central region of the sample, called "gauge region", because it is assumed that the whole deformation occurs there. In a real-case scenario this assumption is not true because the deformation of the transition zones, connecting the gauge region to the rest of the sample, is not neglectable, so (3.21) and (3.22) lead to an overestimation of strain rate and strain, respectively. Moreover, another assumption of these two formulae is that strain is homogeneously distributed in the gauge section, but especially for notched samples this is not true. For these reasons, only stress and force are evaluated according to the equations presented in this chapter.

4. Strain gauges calibration

4.1 Overview

An electric strain gauge is a sensor able to measure the strain of an object. To do so, it converts the strain to an electrical signal. By means of proper correlations, it is possible to obtain the input signal, that is the strain, from the output one.

4.1.1 Resistive strain gauges

A resistive gauge can convert the strain signal to a change in resistance. Regarding the theoretical principles on which it is based, consider a wire made of conductor material of resistivity ρ and let L and A be its length and cross-sectional area, respectively. The resistance R of the wire can be expressed as follows.

$$R = \frac{\rho L}{A} \tag{4.1}$$

Differentiating the (4.1) and considering that the longitudinal strain ε of the wire can be expressed as

$$\varepsilon = \frac{dL}{L}, \tag{4.2}$$

the following can be obtained.

$$\frac{dR}{R} = \frac{d\rho}{\rho} - \frac{dA}{A} + \varepsilon \tag{4.3}$$

Then, expressing the cross-sectional area A in terms of the wire diameter D and considering that ε is linked to the lateral strain ε_{tr} through Poisson's ratio ν, it follows that

$$\frac{dA}{A} = -2\nu\varepsilon. \tag{4.4}$$

Thus, if (4.3) and (4.4) are combined in terms of finite differences, the following can be obtained.

$$\frac{\Delta R}{R} = \frac{\Delta\rho}{\rho} + \varepsilon(1 + 2\nu) \tag{4.5}$$

From (4.5), it turns out that both the strain on the wire and the change in resistivity of its material causes a change in resistance. However, for the temperature ranges which usually characterise the measurement environment of these gauges, the resistivity ρ is approximately constant, so the relative change in resistance $\Delta R/R$ can be assumed to be linearly dependent on the strain ε. Hence, it is possible to define a proportional relationship between these two quantities by means of the gauge factor K.

$$K = \frac{\Delta R/R}{\varepsilon} \tag{4.6}$$

The typical design of a resistive gauge is shown in Figure 4.1.

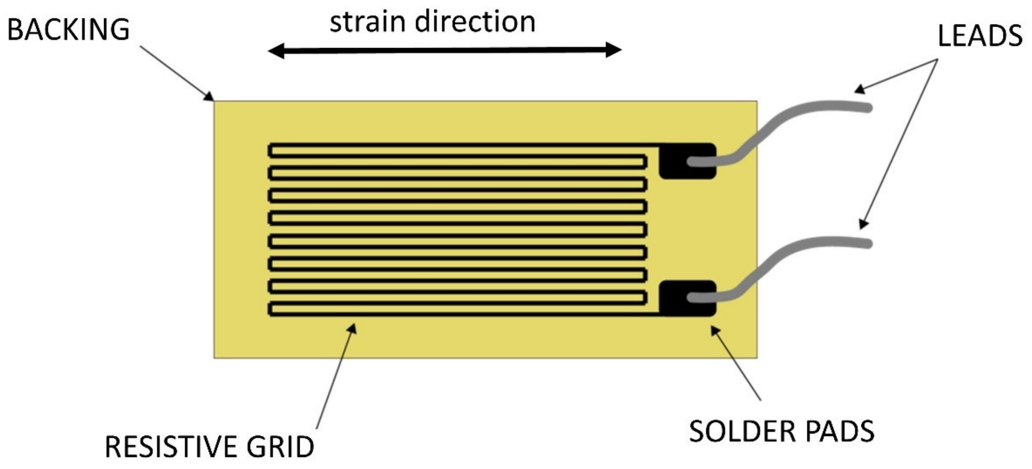


Figure 4.1 *Typical design of a resistive gauge.*

A grid made of conductor material, usually constantan alloy, is cemented on a flexible backing support and connected to an external electrical circuit by means of two solder pads and two leads. The connection between the gauge and the object is carried out through an adhesive so that, when the object is deformed along the strain direction, so is the resistive grid, which therefore changes its resistance, according to (4.5).

The value of K for commercial resistive gauges is around 2 and the typical value of the initial resistance R of the grid is 120 Ω. This means that if the gauge measures a strain ε equal to 2000 μstrain, which may be close to the elastic limit for several basic aluminium alloys, the change in resistance ΔR is

$$\Delta R = KR\varepsilon = 2 \cdot 120 \cdot 2000 \cdot 10^{-6} = 0.48\,\Omega. \tag{4.7}$$

Such a value is difficult to measure through a normal ohmmeter, so an amplification of the output signal turns out to be necessary. The device able to do it is the Wheatstone bridge, shown in Figure 4.2, where R_i $(i = 1, 2, 3, 4)$ represent the resistances, V_{EX} the supply voltage and V_o the output voltage, which is the measured quantity.

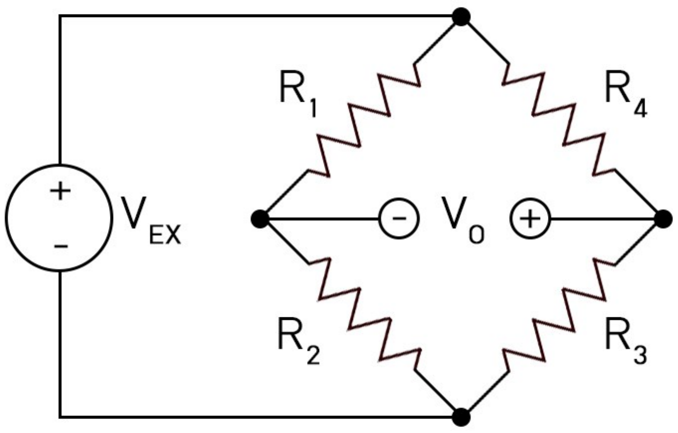


Figure 4.2 *Representation of the Wheatstone bridge.*

Through electrical considerations and simple mathematical steps, it is possible to obtain the following.

$$\frac{V_o}{V_{ex}} = \frac{R_1R_3 - R_2R_4}{(R_1 + R_4)(R_2 + R_3)} \tag{4.8}$$

The bridge is said to be “balanced” if the output voltage V_o is equal to zero. From (4.8), this occurs if the following condition is satisfied.

$$R_1R_3 = R_2R_4 \tag{4.9}$$

Assume now that resistor 1 is nothing but a resistive gauge and that is deformed, so that a change in resistance ΔR_1, and consequently a change in the output voltage ΔV_1, are observed. Thus, (4.8) becomes the following.

$$\frac{V_o + \Delta V_1}{V_{ex}} = \frac{(R_1 + \Delta R_1)R_3 - R_2R_4}{(R_1 + \Delta R_1 + R_4)(R_2 + R_3)} \tag{4.10}$$

If all the resistances have the same value R, the bridge turns out to be balanced from (4.9), so (4.10) becomes

$$\frac{\Delta V_1}{V_{ex}} = \frac{\Delta R_1}{2(2R + \Delta R_1)}. \tag{4.11}$$

As reported in (4.7), the change in resistance ΔR_1 is neglectable with respect to R, so (4.11) can be rewritten as follows.

$$\frac{\Delta V_1}{V_{ex}} = \frac{\Delta R_1}{4R} \tag{4.12}$$

If the same passages are done for the three other resistances separately and the superposition principle is applied, it turns out that ΔV, sum of the single effects ΔV_i $(i = 1, 2, 3, 4)$ can be expressed as

$$\frac{\Delta V}{V_{ex}} = \frac{\Delta R_1 - \Delta R_2 + \Delta R_3 - \Delta R_4}{4R} \tag{4.13}$$

which, considering (4.6), becomes

$$\frac{\Delta V}{V_{ex}} = \frac{K}{4}(\varepsilon_1 - \varepsilon_2 + \varepsilon_3 - \varepsilon_4) \tag{4.14}$$

Thus, it is clear how it is possible, from the measurement of ΔV, to obtain the generic strain ε_i at the generic i^th^ leg ($i = 1, 2, 3, 4$).

In general, the gauges can be arranged in a Wheatstone bridge in three different ways:

1. Full-bridge configuration, when each of the four legs is occupied by a gauge.
2. Half-bridge configuration, when the gauges occupy two legs only, whereas the two others are occupied by passive resistors, not mounted on the object.
3. Quarter-bridge configuration, when only one leg is occupied by a gauge, while in each of the three others a passive resistor is present.

4.1.2 Semiconductor strain gauges

The semiconductor gauges began to be produced in the 1950s as a result of the studies carried out by Smith [18]. The design of such sensors is similar to the one of the resistive gauge, shown in Figure 4.1. In the semiconductor gauges, however, instead of the resistive grid there is a thin strip of semiconductor material that has been cut from a single crystal of silicon or germanium, with the former typically doped with an element improving the gauge characteristics. These two materials have a high piezoresistive effect, meant as the ability to generate voltage when subjected to mechanical stress. This means that when strain is applied to an object on which the gauge is mounted, the change in resistivity $\Delta\rho$ in (4.5) is not neglectable, so the dependence between $\Delta R/R$ and ε is not linear anymore. For this particular characteristic, semiconductor gauges exhibit gauge factors varying with strain and from fifty to hundred times greater than the ones of resistive gauges. Because of the latter characteristic, such gauges are typically used for measuring small levels of strain.

A comparison between the gauge factors of semiconductor and resistive gauges is shown in Figure 4.3, where the relative change in resistance $\Delta R/R$ is plotted as function of strain ε for typical P-type semiconductor and resistive gauges.

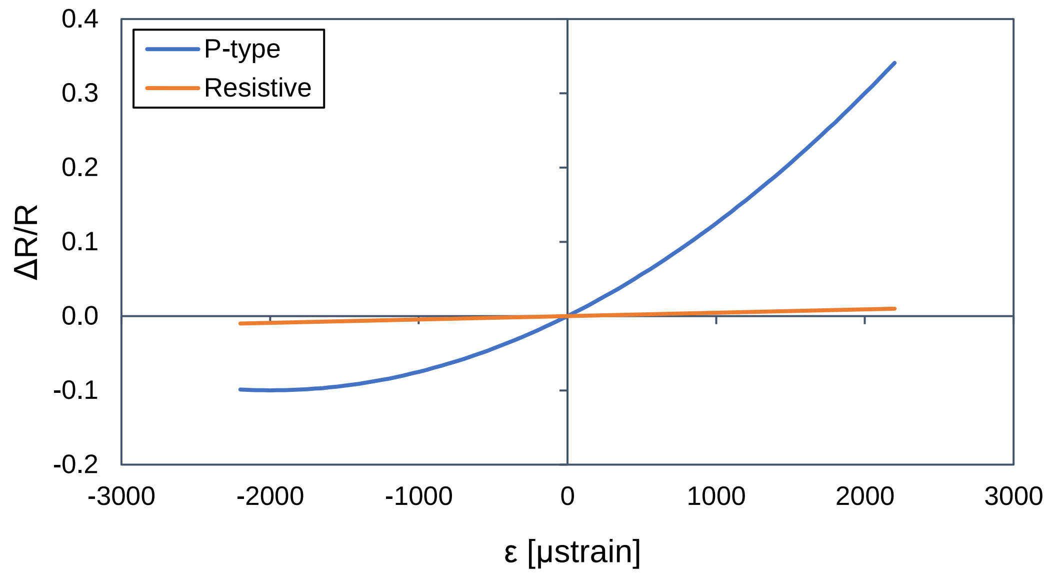


Figure 4.3 *Relative change in resistance as a function of applied strain for typical P-type semiconductor and resistive gauges, respectively.*

For resistive gauges, the slope of the curve, that is the gauge factor, is approximately constant over a wide range of strain, whereas for P-type semiconductor gauges it increases with strain. Moreover, the gauge factor of the latter is from 50 to 100 times greater than the one of the former.

4.1.3 *DyMaLab* strain gauges

In the experimental apparatus of *DyMaLab* there are in total eight strain gauges arranged in four pairs. Three of these pairs consist of resistive gauges, the fourth of semiconductor gauges and their main characteristics are reported in Table 4.1.

Gauge type	**Manufacturer**	**Model**	**R [Ω]**	**K**
Semiconductor	Kyowa	KSP-2-120-E4	125.8 ± 2 %	130 ± 3 %
Resistive	Tokyo Measuring Instruments Laboratory Co.	FLA-2-11	120 ± 0.3	2.14 ± 1 %

Table 4.1 *Main characteristics of resistive and semiconductor gauges of DyMaLab.*

With reference to Figure 4.4, all the eight gauges are mounted on the bars at diametrically opposed positions. On the input bar there are two pairs of resistive gauges, whereas on the output bar one pair of resistive gauges and another one of semiconductor gauges are present. The reason in using semiconductor gauges lies in the fact that this kind of sensors, thanks to the high value of gauge factor, enables to obtain reliable strain measurements up to nanostrain, which is about three orders of magnitude lesser than the microstrain provided by common resistive gauges. This feature is particularly important when the forces in the bars are low, as it happens when samples made of composite material are dynamically tested.

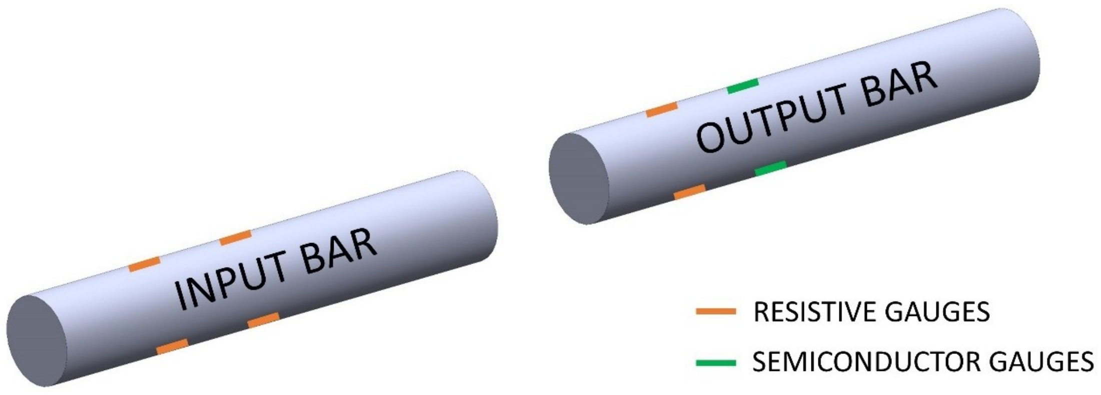


Figure 4.4 *Strain gauges arrangement on the bars.*

Each pair of gauges is connected, according to a half-bridge configuration, to a different Wheatstone bridge, supplied by a constant voltage of 5 V. With reference to Figure 4.2, the pairs of resistive gauges are placed in legs 1 and 3, in order to compensate bending effect. Since the value of the fixed resistances R_2 and R_4, 120 Ω, is the equal to the gauges one, the condition (4.9) is satisfied and the bridge is balanced. As far as the semiconductor gauges are concerned, they are arranged as shown in Figure 4.5.

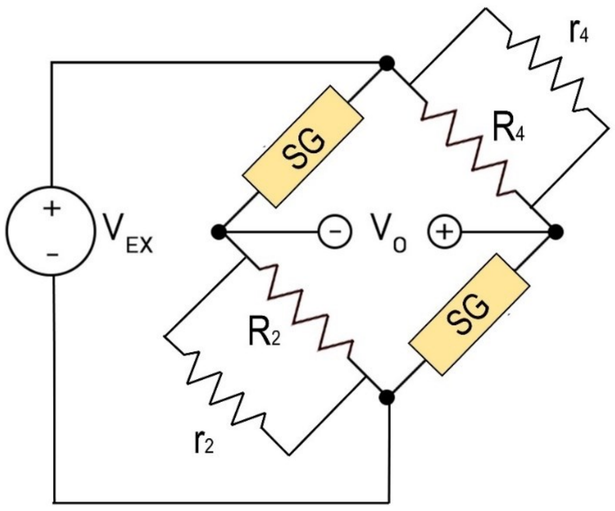


Figure 4.5 *Wheatstone bridge configuration for the pair of semiconductor gauges.*

The two gauges are positioned in opposite legs of the bridge, in order to compensate bending effect, however, with respect to the traditional configuration, in each of legs 2 and 4 a parallel resistor is placed in parallel to the original one. The resistance values r_2 and r_4 of the added resistors, equal to 2600 Ω, are considerably high if compared to the values R_2 and R_4 of the original ones, equal to 120 Ω. In this way, the resultant resistance in each of legs 2 and 4, equal to 114.7 Ω, is lower than the original values R_2 and R_4, so the current through the gauges increases, with the same supply voltage V_{ex}. The global results is a stabilisation of the signal mean value which would otherwise exhibit significant changes, however, the bridge is no longer balanced because (4.9) is not satisfied. Therefore, before performing any dynamic experiments, the output voltage is manually balanced by injecting an equal and opposite voltage. Even though this procedure results in a null output, the bridge is still resistively unbalanced, so non-linearity is induced.

4.2 Motivation and aim

As discussed in §3.3, the strain gauges function is relevant because, through their measurement of strain waves amplitude, it is possible to obtain the loading state of the sample. In particular, stress and force can be evaluated through (3.16) and (3.17), respectively, which require the knowledge of the transmitted strain wave amplitude ε_t at the output bar-sample interface, recorded by the resistive and semiconductor pairs of gauges mounted on the output bar. Although the strain levels obtained from the two measurements should be approximately equal, it has been experimentally observed that the ratio between the signal of latter and the signal of the former is always lesser than one and the reason is unknown. Moreover, this ratio is not constant but varies, in an apparently random way, from 0.78 to 0.88, which corresponds to a relative error in the signal of semiconductor gauges between 22 and 12 %, with respect to the resistive gauges signal. Thus, when an experiment has to be carried out, the semiconductor pair is calibrated with respect to the resistive one, whose measurement is assumed to be correct.

Moreover, the aforementioned equations (3.16) and (3.17) require, in addition to the quantity ε_t, the Young's modulus E of the bars material. Its value is assumed equal to 70000 MPa, corresponding to the value given by the manufacturer, however, it may slightly differ from the actual one.

For what mentioned, it turns out that, because of the uncertainty in the values of ε_t and E, the estimation of force and stress through the sample may not be precise. This is the reason why a calibration of these sensors, through a load cell, is necessary. This process, presented in the following subchapters, is carried out for the resistive gauges mounted on the input bar, too, in order to investigate the quality in their measurements of incident and reflected strain wave amplitudes ε_i and ε_r, necessary for the evaluation of strain rate and strain through (3.21) and (3.22), respectively.

4.3 Methods

To calibrate a Wheatstone bridge, there are basically two different ways, direct and indirect, respectively.

In the indirect method, a voltage output, obtained by simulating the deformation of the gauge, is compared to the one observed when the gauge is effectively loaded. The simulation of gauge deformation is carried out by unbalancing the bridge through an increase or decrease of one leg resistance, which can be obtained through the positioning of a precisely known resistor in parallel to the original resistor of that leg.

In the direct method an accurately known load is applied to the bridge and its output is properly adjusted. The current work is based on this method and thought in a relatively simple way. In fact, by loading the input and the output bar separately and in static conditions, it is possible to compare the signals of the gauges to the ones given by a load cell, assumed to be the reference measuring instrument. The input bars are loaded in compression by means of a hydraulic cylinder and below the elastic limit. Moreover, the loading process is carried out for different force levels, by varying the pressure of the hydraulic oil in the cylinder.

The load cell is a C2 model built by *HBM* and is able to measure compressive and tensile forces. As shown in Figure 4.6, such a sensor consists of a cable connected to a measuring body, above which a button, where the load is applied, is. An additional component, called “thrust piece”, is placed on the button to avoid the effects of potential lateral forces.

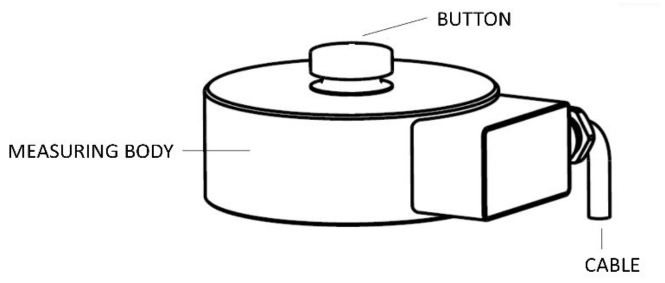


Figure 4.6 *The C2 load cell used in the calibration.*

The sensitivity of the load cell is equal to 2 mV/V ± 2 % at the nominal force, in turn equal to 50 kN, a value greater than the forces loading the input and output bars in a typical tensile or compressive experiment. This means that an investigation of the whole operating range of the gauges is possible. Precisely, during dynamic tests, the maximum force is always observed in the input bar and can reach 15 ÷ 20 kN. Finally, the supply voltage provided to the load cell is controlled through a closed loop system, thanks to the six-wire technique which compensates the electric losses in the cables caused by wire resistance.

Regarding the hydraulic cylinder, shown in Figure 4.7, it is a RC51 model built by *Enerpac* and it can apply a maximum force of 45 kN which, again, is greater than the maximum one observable in the input bar in a tensile or compressive experiment.

Figure 4.7 *The RC51 hydraulic cylinder used in the calibration.*

The high pressure oil is supplied to the cylinder by a manual and double acting hydraulic pump, which is a P77 model built by *Enerpac* as well and shown in Figure 4.8.

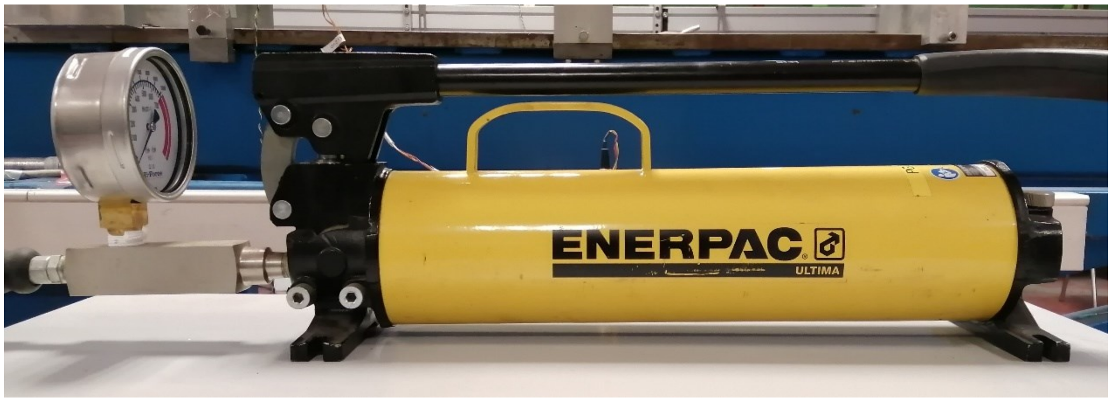


Figure 4.8 *The P77 hydraulic pump used in the calibration.*

4.3.1 Design of the calibration apparatus

In order to load the bars through the cylinder and record the force signal by means of the load cell, a mechanical support system must be designed. The schematic representation of such an apparatus is shown in Figure 4.9.

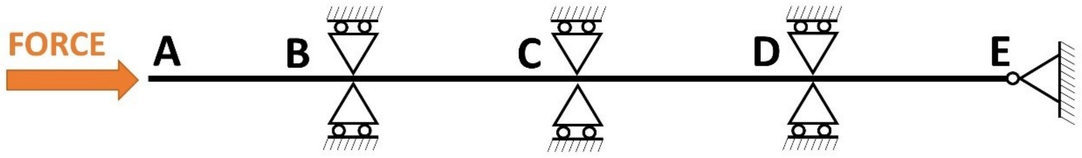


Figure 4.9 *Schematic representation of the mechanical support apparatus for the calibration.*

At point A, the hydraulic cylinder mounted on a support loads in compression the bar and the load cell, mounted on another support at point E, measures the longitudinal force that is applied. In order to avoid any buckling issues, a proper number of evenly spaced supports constraints the vertical displacement of the bar at points B, C and D. The number of these supports is determined according to the formula for Euler’s critical load.

However, if the supports avoiding bucking are already present in *DyMaLab*, so are not the ones of cylinder and load cell. Therefore, these mechanical components need to be designed according to conventional procedures. In the current subchapter, the whole design process is not presented because of the basic concepts behind it, however, the technical drawings of all the produced components are reported in the appendix.

With regard to assembly supporting the cylinder, it is shown in Figure 4.10.

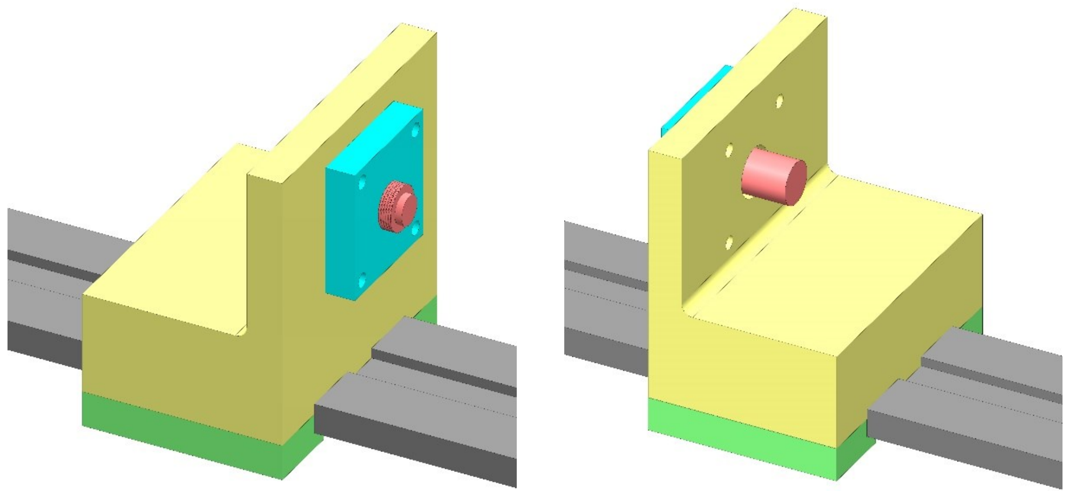


Figure 4.10 *Anterior (a) and posterior (b) view of the assembly supporting the cylinder.*

A big block (in yellow) made of AlMgSi1, an aluminium alloy, is placed on the aluminium guide (in grey), already present in laboratory. A hole in the big block houses the cylinder (in red), that is fixed, through a thread, to a flange made of steel S235 (in light blue). Then, the flange is in turn connected to the big block by means of four horizontal M10 bolts. Since the cylinder loads the bar in compression, the whole support assembly would tend to detach from the latter, moving towards left with reference to Figure 4.10 (a), so it is necessary to fix it to the guide. This connection would be easily carried out by means of some vertical bolts connecting the two parts but this would imply drilling holes in them. For this reason, another approach, based on friction between the contact surfaces of the big block and the guide, is followed in order to keep the guide unchanged. Hence, two small blocks of AlMgSi1 (in green) are connected to the big block by means of three vertical M14 bolts each. By means of torque tightening, these bolts press the big block against the guide and a horizontal frictional force is developed at the guide-big block interface, loading the latter. The design process makes sure that this force is always greater than the one applied by the cylinder to the bar, so that the whole assembly can be kept motionless.

In Figure 4.11, the assembly supporting the load cell is represented. The load cell (in red) is connected to the big block (in yellow) by means of four horizontal M10 bolts, positioned in holes passing through the whole thickness. The aforementioned thrust piece (in light blue) is placed on the load cell button in order to neutralise any lateral forces. The method followed to fix the big block to the guide (in grey) is the same used for the cylinder support, and so are the materials used for the big and small blocks, where the latter are represented in green in Figure 4.11.

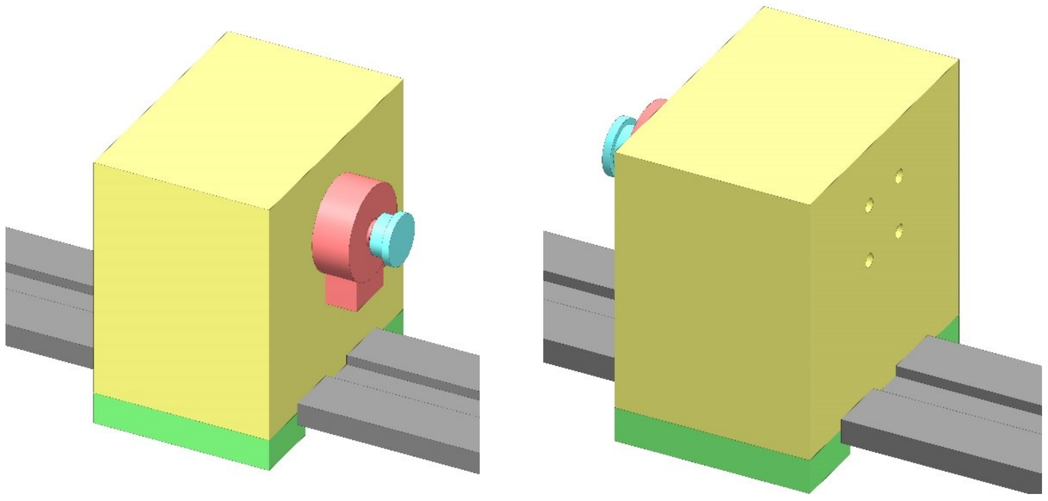

(a) (b)

Figure 4.11 *Anterior (a) and posterior (b) view of the assembly supporting the load cell.*

4.3.2 Procedure

After the support assemblies of Figure 4.10 and Figure 4.11 are mounted and placed on the guide, the bar is placed in between them and on the supports avoiding buckling. Then, the cylinder is connected to the hydraulic pump and the load cell to the oscilloscope. After properly set *Perception*, the software connected to the oscilloscope and able to control load cell and gauges, the whole apparatus, shown in Figure 4.12, is ready for the calibration.

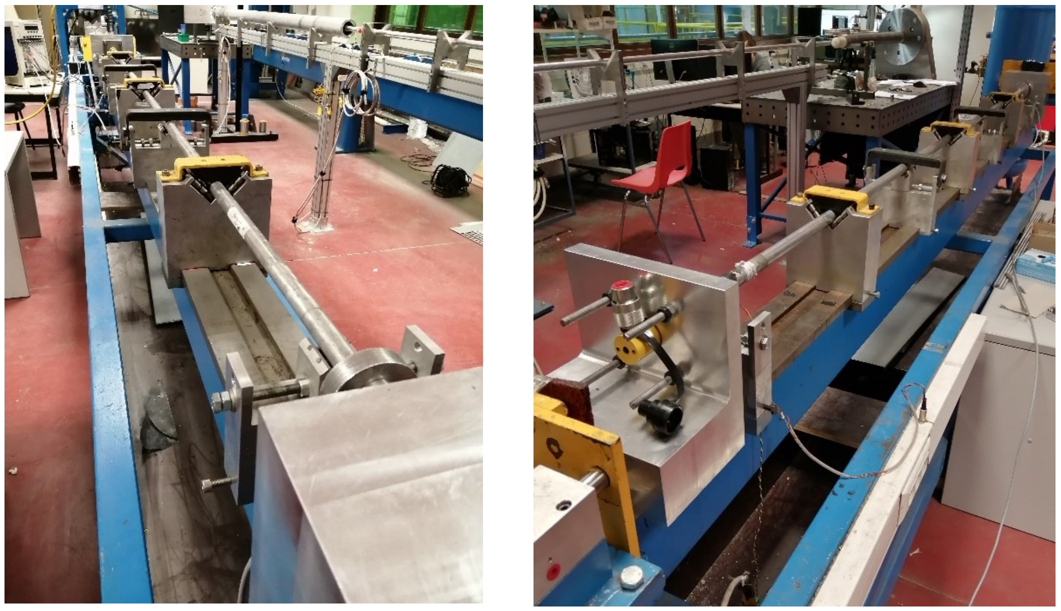


Figure 4.12 *Calibration apparatus for the output bar seen from the load cell (a) and from the cylinder (b).*

When the bar is still unloaded, the potential residual voltage is compensated for each of the three measuring instruments through *Perception*¸ the software connected to the oscilloscope. Then, the three signals are simultaneously acquired, in order to evaluate again potential residuals. If the signals coming from the gauges are strains, recorded in microstrain, the one coming from the load cell is a force, recorded in newton.

Subsequently, the loading process can begin. As mentioned §4.3, the capacities of load cell and cylinder enable to explore the behaviour of the bars over the whole range of forces usually loading them during dynamic experiments. For this reason, it is chosen to load the bars from 0 to 20 kN, which corresponds to the oil pressure varying in the range of 0 ÷ 300 bar. The step increment in the loading process is set equal to 20 bar because well controllable through a manometer attached to the pump, as shown in Figure 4.13.

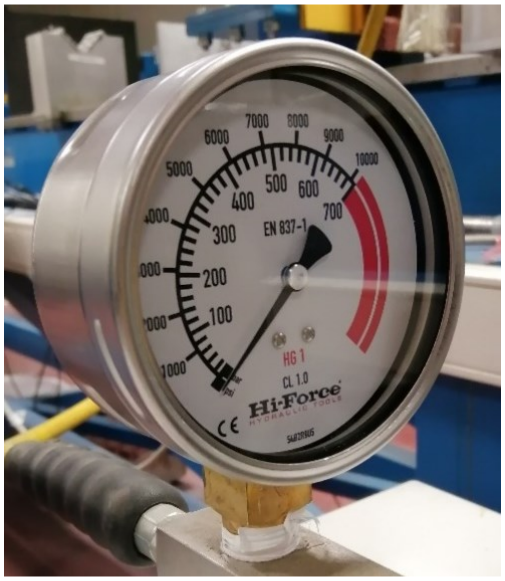


Figure 4.13 *The manometer attached to the pump. The inner and outer tick marks refer to bar and psi, respectively.*

For each loading condition, so for each pressure level, the signals coming from the gauges and the load cell are recorded three different times, in order to avoid potential bias.

For what regards the signals acquisition, the sampling rate must be properly set. In a typical experiment with the Hopkinson Bar, its value is 1 MHz in order to clearly record all the strain waves, however, since the calibration process is static, a lower value can be chosen. Thus, the sampling rate is set equal to 1 kHz and the duration of each acquisition equal to 1 second, so that 1000 data points are recorded for each instrument and 3000 in total.

Finally, a critical issue regards the unavoidable hydraulic losses in the circuit composed of pump, pipe and cylinder. When the oil pressure in the pump is manually set to a certain value, these losses progressively lower it, so the three different acquisitions refer to pressure levels that are slightly different. In order to make the acquisitions statistically comparable, these levels must be kept as close as possible to each other, thus previous acquisitions are carried out in order to evaluate the time interval in which the change in the pressure value is minimal. Figure 4.14 shows the typical trend observed for the load cell signal right after a pressure increase of 20 bar.

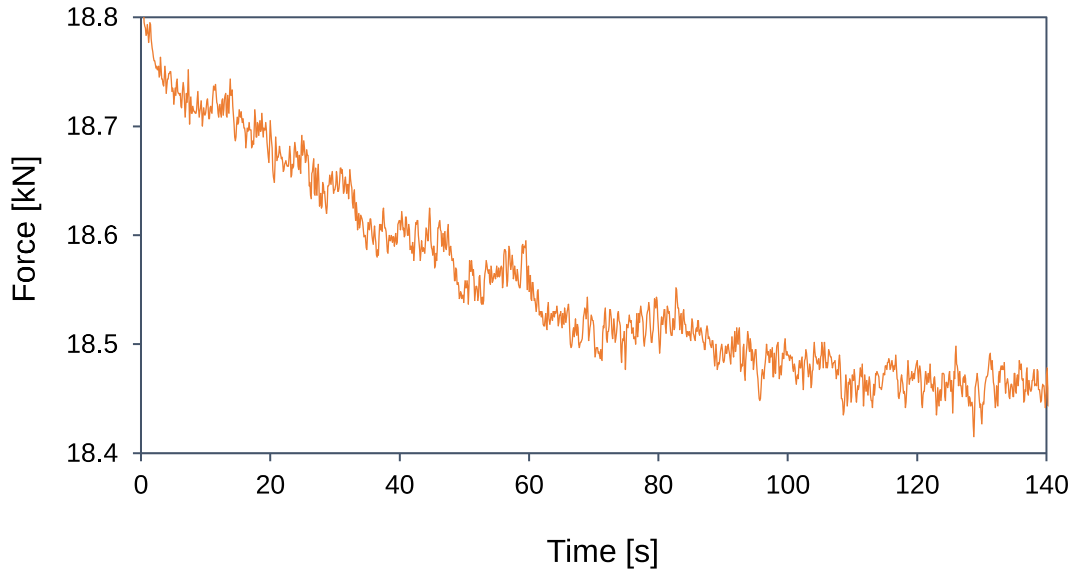


Figure 4.14 *Typical trend observed for the load cell signal over time, right after a pressure increase of 20 bar.*

It is possible to observe that after a steep decrease, the signal, so the oil pressure, are approximately constant from 60 to 80 seconds. Therefore, for each pressure level, the first acquisition starts 60 seconds after increasing the pressure and the two others subsequently, making sure that the whole process does not last more than 20 seconds. It is also true that a horizontal plateau, where the oil pressure is approximately constant, occurs also after 80 seconds, however, it is preferable to consider only the first horizontal plateau (60 ÷ 80 s), so that the pressure value is closer to the one initially set in the pump.

4.3.3 Data processing

As mentioned in §4.3.2, each acquisition consists of 3000 data points and, for each of the several pressure levels investigated, three acquisitions are carried out. This means that at the end of the calibration a huge number of data is obtained, so a processing is needed before discussing the results. In the following points the steps of this procedure are explained, where points 1, 2, 4, 6 refer to a single measuring instrument.

1. With reference to the acquisition carried out when the bar is unloaded, the residual q is evaluated as the mean value of the 1000 data points, as shown for the load cell in Figure 4.15.

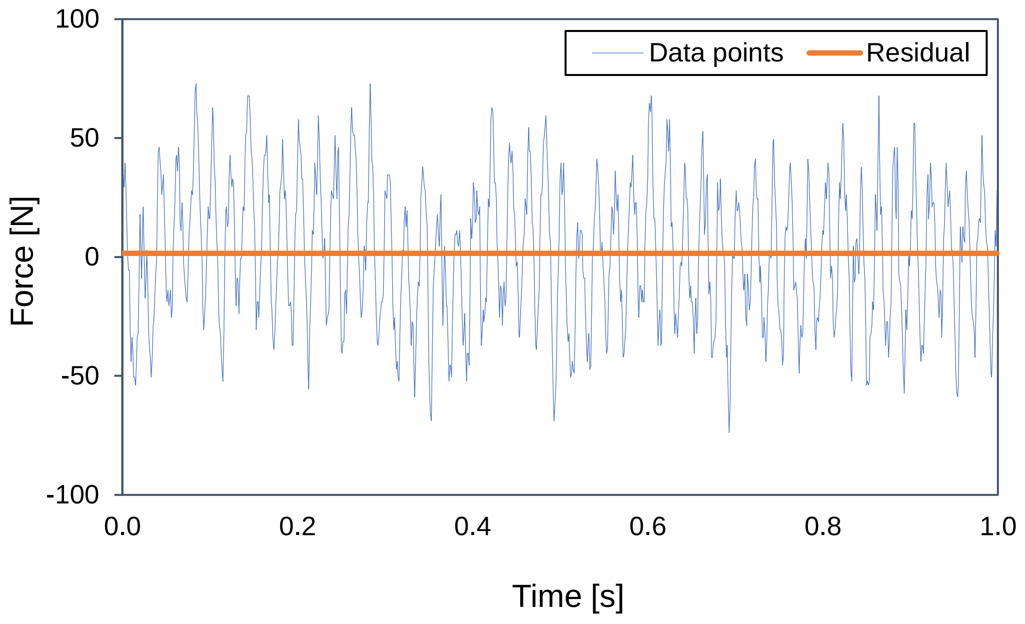


Figure 4.15 *Data points and residual of load cell signal at 0 bar.*

   Because of the compensation manually carried out through *Perception*, the value of q is always close to zero.

2. For the generic kth pressure level $(k = 0, 20, 40, \dots, 300)$, the jth acquisition $(j = 1, 2, 3)$ is corrected by means of the residual q. Because of the small value of q, this correction does not produce considerable differences in the signal.

3. In order to compare the gauges signals to the load cell one, the former are converted from microstrain to newton, according to the following equation. A_b is the cross sectional area of the bar and E its Young's modulus, assumed equal to 70000 MPa. Moreover, $F_{i,j,k}$ is the force obtained from $\varepsilon_{i,j,k}$, which is are the ith data point $(i = 1, 2, \dots, 1000)$ of the jth acquisition of gauge signal at the kth pressure level.

$$F_{i,j,k} = \varepsilon_{i,j,k} \cdot E \cdot A_b \tag{4.15}$$

4. The quantity $x_{j,k}$ $(j = 1, 2, 3)$ is calculated as the mean value of the 1000 data points of the jth acquisition and assumed to well represent the signal. The comparison between $x_{j,k}$ and the original data points is shown in Figure 4.16, corresponding to the first acquisition of the load cell signal $(j = 1)$ for a pressure level of 100 bar $(k = 100)$.

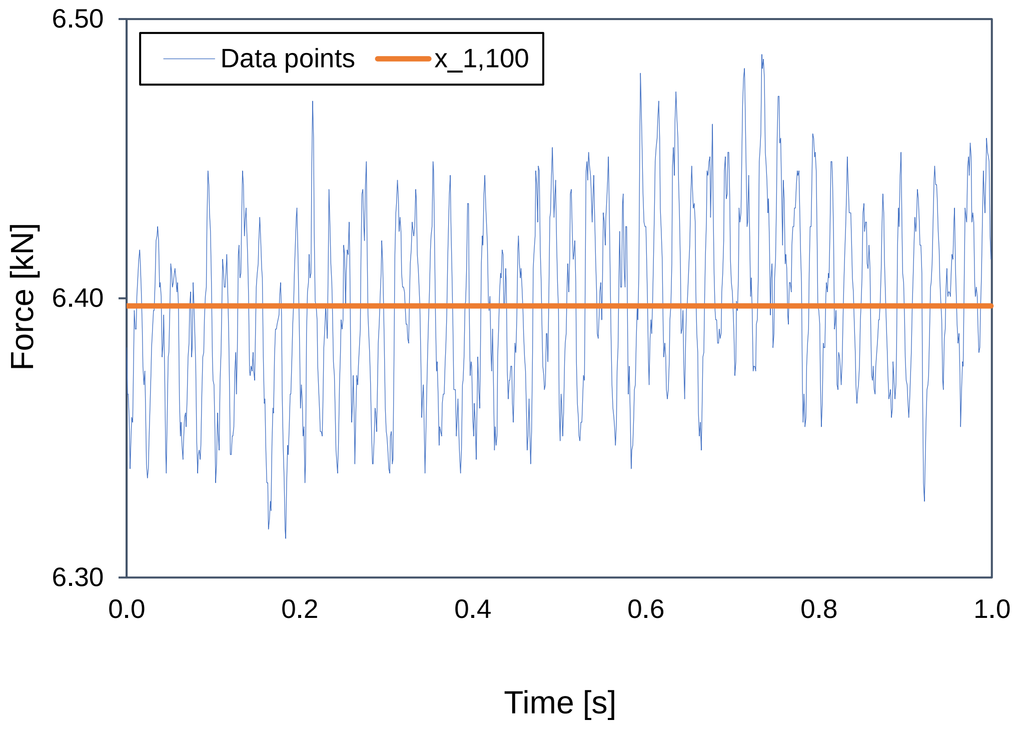


Figure 4.16 *Data points and mean value $x_{1,100}$ of load cell first signal at 100 bar.*

5. The quantity X_k, representative value of the kth pressure level, is calculated as the arithmetic mean of $x_{j,k}$ $(j = 1, 2, 3)$, as reported in Figure 4.17, in which all these four quantities are plotted over time for the load cell signal at 100 bar. The differences in the three values $x_{j,k}$ is caused by the aforementioned hydraulic losses which lower the pressure level, in fact $x_{1,100} > x_{2,100} > x_{3,100}$.

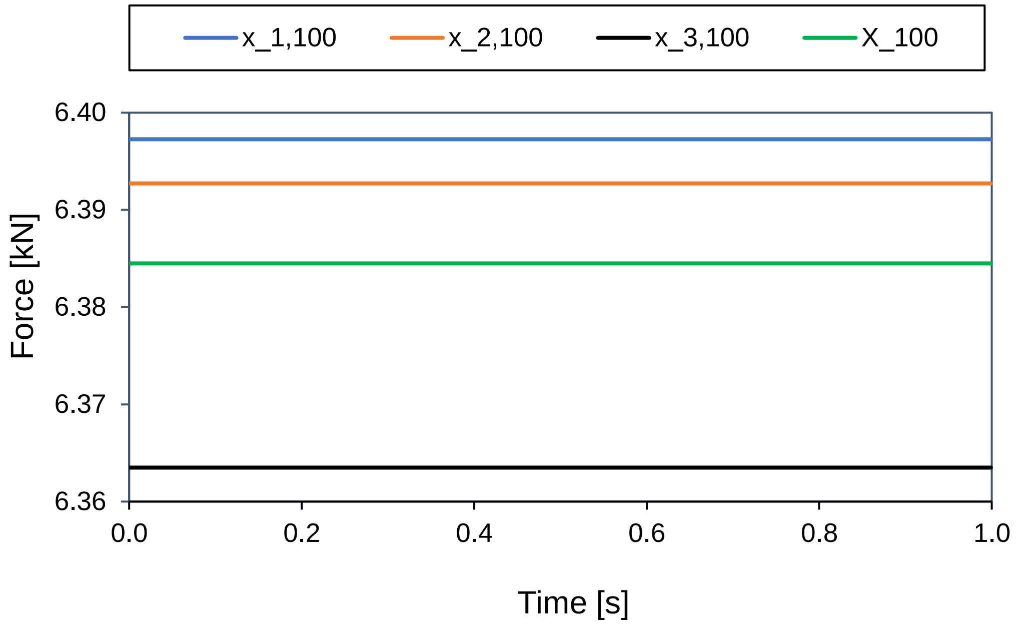


Figure 4.17 *Mean values $x_{1,100}$ (i=1, 2, 3) and representative mean value X_{100} of load cell signal at 100 bar.*

6. Thanks to the conversion of point 3, all the signals are now expressed in newton. Thus, for the kth pressure level, the ratio $r_{j,k}$ between the quantity $x_{j,k}$ of the two gauges and the same quantity of the load cell is calculated, in order to evaluate how close the gauges signals are to the load cell one. The same procedure is applied to the quantity X_k, in order to obtain the ratio R_k, representative of the kth pressure level.

4.4 Results and discussion

4.4.1 Output bar

In Figure 4.18 the signals ratio R_k is plotted for different levels of oil pressure, with reference to resistive (RG) and semiconductor (SG) gauges. The coloured dots, corresponding to the experimental data, are connected through a dashed grey line in order to quickly visualize the trend, whereas the vertical solid black lines refer to the maximum deviations or $r_{j,k}$ $(i = 1, 2, 3)$ from R_k. As discussed in §4.3.2, these deviations occur because of the unavoidable hydraulic losses which lower the pressure level and make the three different acquisitions less comparable from a statistical point of view. It is possible to observe that the deviations from R_k are higher for lower pressures and reach a more stable value for pressures greater than 120 bar. This may lie in the fact that the entity of such hydraulic losses is approximately the same, so for lower pressures the relative change in pressure is more relevant.

For what regards the signal ratios R_k of resistive and semiconductor gauges, even though they both do not reach a clear horizontal plateau, the stabilisation of the trend is observed to happen for pressure levels greater than 200 bar, corresponding to longitudinal strain and force in the bar around 400 μstrain and 13 kN, respectively. This is mainly caused by the late stabilisation of the load cell signal. In fact, if the gauges are thought to measure deformation at least up to microstrain, the load cell is not that accurate up to this level because of its high capacity, equal to 50 kN, which makes it more suitable for forces greater than the ones usually loading the bars.

Thus, the resistive gauges exhibit a relative error around 5 % with respect to the load cell signal. Of course, the potential slight bending caused by a not perfectly longitudinal load is not the reason of such a deviation, because the bridge configuration of these gauges ensures bending compensation. Instead, because of the gauges misalignment with respect to the longitudinal axis of the output bar, the measured strain would not correspond to the longitudinal one and would be, for this reason, lower.

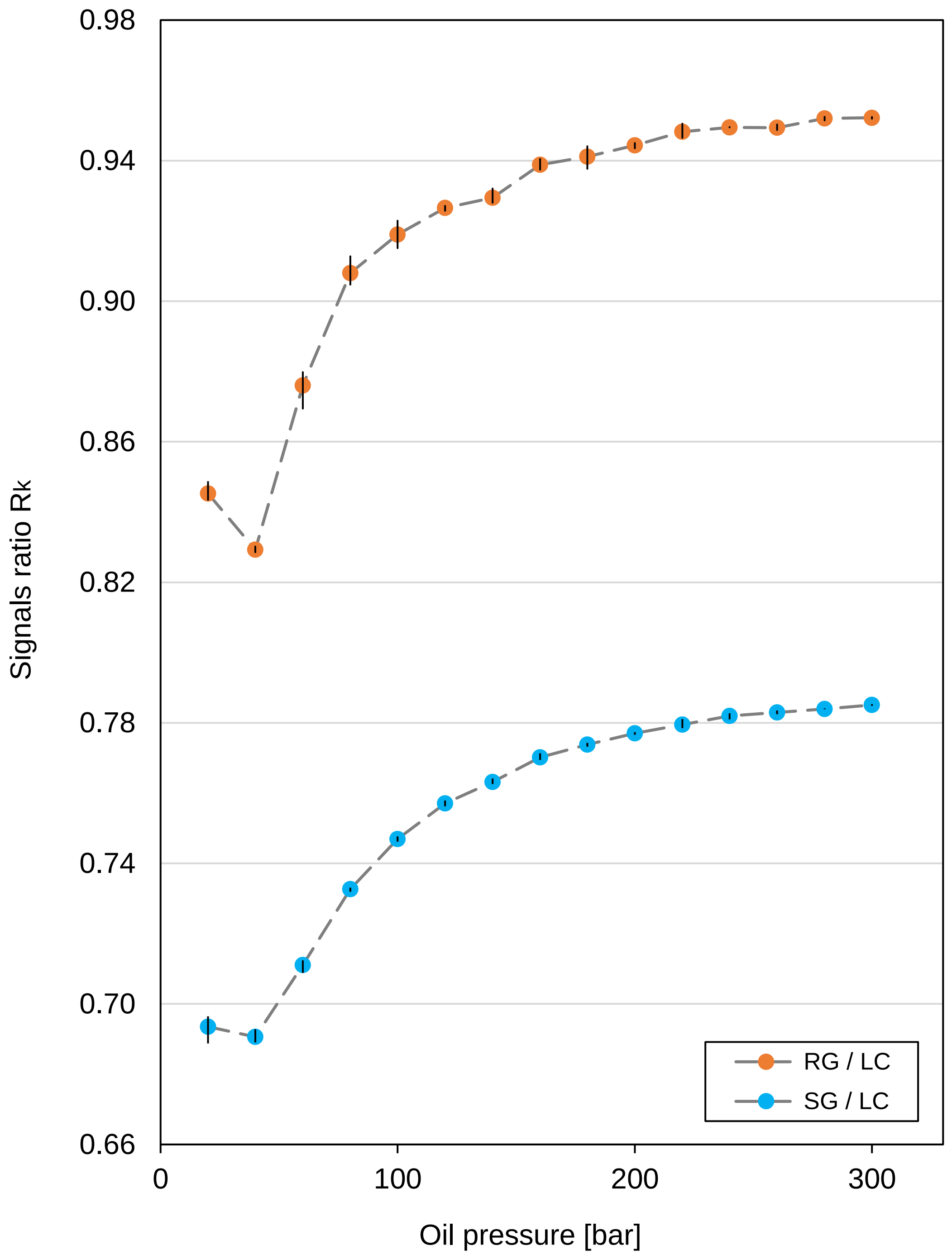


Figure 4.18 *The ratio R_k between the signals of gauges and load cell for the first experiment on the output bar. The vertical black lines refer to the maximum deviations of $r_{j,k}$ from R_k*

Another reason which may justify the value of ratio R_k could be the uncertainty in the value of Young's modulus E, involved in the conversion of the gauges signal from microstrain to newton. In particular, a value of E around 73400 MPa would make the ratio R_k approximately 1. If this value was correct, the value given by the manufacturer, equal to 70000 MPa, would be affected by an error of 4.6 %, which is unlikely because of the instrumentation used in the determination of such quantities during the production process. Moreover, a value of E around 73400 MPa would potentially explain the difference between the signals of resistive gauges and load cell, but not between resistive and semiconductor gauges. For these reasons, other possible reasons for such signal differences are further discussed.

As already mentioned, the ratio R_k for the semiconductor gauges is far from one and corresponds to a relative error around 22 % with respect to the load cell. This results in a relative error around 18 % with respect to the resistive gauges signal, which is within the range of 12 ÷ 22 % observed during dynamic tests. One possible reason of such a great deviation can lie in the fact that, because of the piezoresistive effect, the gauge factor of a semiconductor gauge, that is the slope of the curve of Figure 4.3, changes with strain, which means that the gauge factor value set in *Perception* and equal to 130 can differ from the actual one. However, for the level of strain investigated, this change is neglectable, in fact the trend of the ratio R_k for the semiconductor and resistive gauges is very similar. The real issue lies in the fact that when the semiconductor gauge is cemented onto the backing, the adhesive dries and compresses the gauge which, because of the piezoresistive properties, changes its resistance. With reference to Figure 4.3, it means that the unstrained condition goes from the origin to some point of the third quadrant, where the gauge factor is lesser. Thus, it turns out that in a real-case scenario the nominal gauge factor, equal to 130, differs from the actual one and this may be one of the most relevant reasons why such values or the ratio R_k are observed. In order to further investigate this aspect of the problem, the resistance of the gauges mounted on the bar should be measured in unstrained conditions and then compared to the nominal one indicated by the manufacturer.

Moreover, as discussed in §4.1.3, the Wheatstone bridge configuration for semiconductor gauges does not provide resistive balance so, even though a compensation of the output voltage is carried out before any experiments, non-linearity remains. Since this is probably the second main cause of a ratio R_k as low, it is suggested to increase the values of resistances R_2 and R_4 in order to balance the bridge as much as possible.

Finally, another possible improvement could be reached by lowering the supply voltage, currently equal to 5 V. High voltages would in fact heat up the gauge, causing an unwanted change in its gauge factor. In particular, the value of 5 V is the maximum suggested by the manufacturer, so by lowering it the operating conditions would be brought farther from the limit.

One day after the experiment whose results are shown in Figure 4.18, another identical one has been carried out and its results are shown in Figure 4.19.

Considering again only the pressure range of 200 ÷ 300 bar, where the load cell signal is stable and the hydraulic losses neglectable, it is possible to observe that the ratio R_k for the resistive gauges exhibits the same values of the first experiment, around 0.95.

Instead, for what regards the semiconductor gauges, even though the trend is similar to the one of the first experiment, the stabilised error, approximately 24 %, is different from the one observed in the first experiment, around 22 %. Since the conditions of the two experiments are completely the same, the only possible difference regards the slight changes in temperature, humidity and light. Thus, the reason why the ratio between the signals of semiconductor and resistor gauges changes from time to time can be attributed to the influence of these three parameters on semiconductor gauges measurements. Temperature, humidity and light, however, are not controlled during the presented experiments so, even though the causes of such fluctuations are identified, they are not here quantifiable. Thus, an interesting further investigation would consist in repeating the experiments while accurately monitoring these three parameters, in order to investigate how the measurements are affected by them.

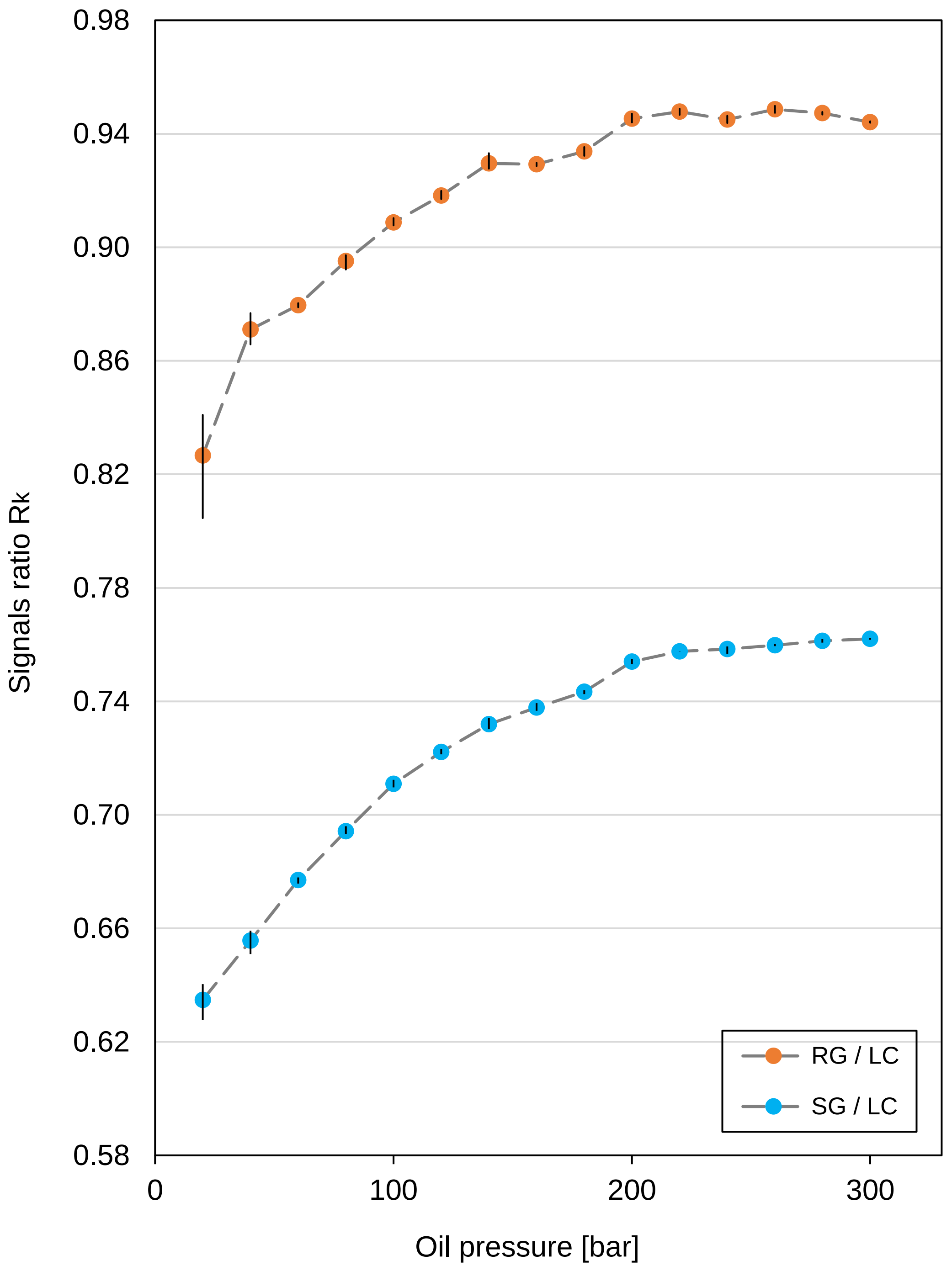


Figure 4.19 *The ratio R_k between the signals of gauges and load cell for the second experiment on the output bar. The vertical black lines refer to the maximum deviations of $r_{j,k}$ from R_k.*

4.4.2 Input bar

The same experiments carried out for the gauges of the output bar are performed for the input bar as well and the results are shown in Figure 4.20.

With regard to the signals ratio R_k for the first pair of resistive gauges (RG1), the hydraulic losses become neglectable and the load cell stabilises for pressures greater than 120 and 220 bar, respectively. For this reason, the region of interest is in the pressure range of 220 ÷ 300 bar, where the value of ratio R_k is approximately 0.94. Again, this means that the measurements of gauges RG1 are affected by a relative error around 6 %, similar to the one observed for the resistive gauges RG on the output bar. As discussed for the resistive gauges RG on the output bar, the reasons is not likely to be found neither in the value of Young's modulus nor in the potential bending effect, but in the mounting conditions. In fact, slight misalignments of the gauges with respect to the longitudinal axis of the input bar could worsen the measurements.

Regarding instead the second pair of resistive gauges (RG2), because of the aforementioned reasons, the pressure range of interest is 260 ÷ 300 bar. Here, the ratio R_k is approximately 0.92 and corresponds to a relative error around 8 %: again, one possible reason could be again the misalignment of the gauges with respect to the longitudinal axis of the input bar. However, the ratio R_k is slightly lower than the one observed for the gauges RG1, so possible electrical errors may occur in the measurement chain. For all the aforementioned reasons, it is suggested to mount these gauges on the bars again, making sure, through proper instrumentation, that the measuring direction effectively corresponds to the longitudinal axis of the bar.

Finally, as done for the output bar, one day after the discussed experiment another one has been performed under the same conditions, except for temperature, humidity and light, which have not been controlled. The obtained results are consistent with to the ones of Figure 4.20 and for this reason not reported. This proves how the resistive gauges are much less sensitive to slight changes in temperature, humidity and light with respect to semiconductor ones.

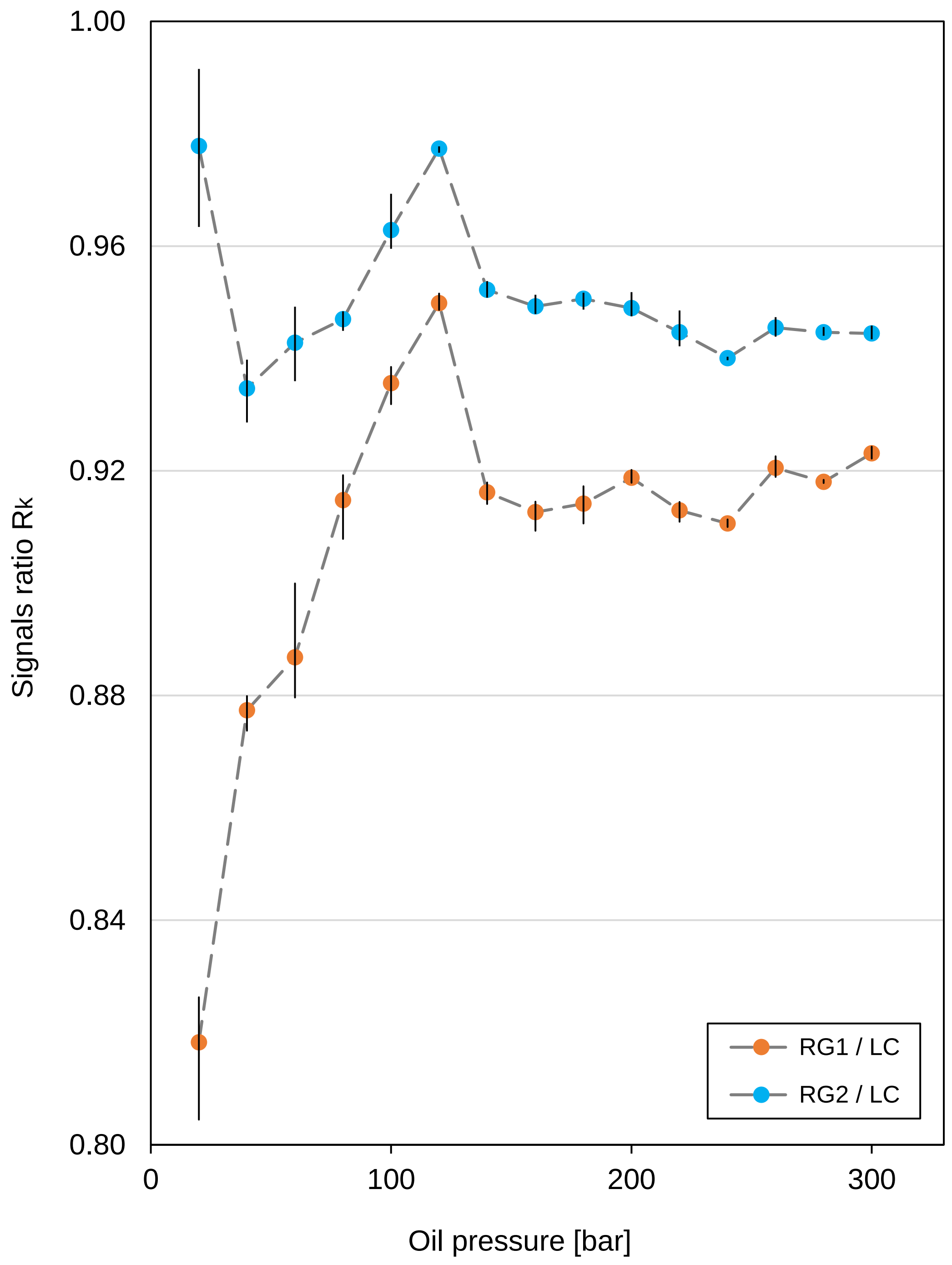


Figure 4.20 *The ratio R_k between the signals of gauges and load cell for the experiment on the input bar. The vertical black lines refer to the maximum deviations of $r_{j,k}$ from R_k.*

5. Optimization of sample geometry

5.1 Motivation and aim

As already discussed in §2.3, one of the factors affecting the stability of retained austenite is stress state. Though this dependence has been widely investigated for static cases [11] [12], few dynamic counterparts exist, that is one of the reasons why the *Dynaustab* project was born.

A quantity of relevant importance for stress state definition is a dimensionless parameter called "stress triaxiality" and defined as follows.

$$T = \frac{\sigma_m}{\sigma_{VM}} \tag{5.1}$$

In (5.1), σ_m e σ_{VM} represent hydrostatic stress and von Mises equivalent stress, respectively, therefore T expresses the relative degree of hydrostatic stress in a given stress state. The expressions of σ_m e σ_{VM} are reported in (5.2) and (5.3), where the term σ_{ij} $(i, j = 1, 2, 3)$ is the generic component of Cauchy stress tensor.

$$\sigma_m = \frac{1}{3}(\sigma_{11} + \sigma_{22} + \sigma_{33}) \tag{5.2}$$

$$\sigma_{VM} = \sqrt{\frac{(\sigma_{11} - \sigma_{22})^2 + (\sigma_{22} - \sigma_{33})^2 + (\sigma_{33} - \sigma_{11})^2 + 6({\sigma_{12}}^2 + {\sigma_{23}}^2 + {\sigma_{31}}^2)}{2}} \tag{5.3}$$

Moreover, T can be an indicator of the type of fracture, ductile or brittle, of the material. In fact, the lower its value, the closer the stress state to shear and therefore the more possible the slip motion between adjacent crystal planes. For this reason, lower values of T promote ductile fracture. On the other hand, the higher T, the more hampered the slip motion and a brittle fracture is more likely to occur.

During a generic car collision, stress triaxiality can take many different values and this justifies that one of the aims of the *Dynaustab* project is to investigate the dependence of austenite stability on this stress parameter under dynamic conditions, according to the following workflow.

1. Optimization of already developed sample geometries, each one characterised by a different global value of stress triaxiality T. The aim of the optimization is to reach, for each geometry, values of T as constant as possible both in a specific space region and over time, in order to associate a single value of T to each experiment. Moreover, the other aim of the optimization is to have values of T as high as possible.
2. Development of techniques to interrupt dynamic tests when certain strain levels are achieved.
3. Dynamic tensile tests on the optimized samples carried out through the Hopkinson Bar.
4. Microstructural analysis of the austenite and martensite content through X-Ray Diffraction (XRD) techniques.

The work presented in this chapter regards the point 1 of the previous list. In the recent years, more and more optimization studies like this have been found in the literature [19] [20], especially because their bottom-up approach is much less time-consuming than a traditional top-down method.

The current work aims to optimize a particular sample geometry, shown in Figure 5.1. The function of the two clamp regions, shown in light blue, is to connect the sample to the input and output bar, by means of the pin holes. The gauge region, in yellow, represents the region of interest for the measurements.

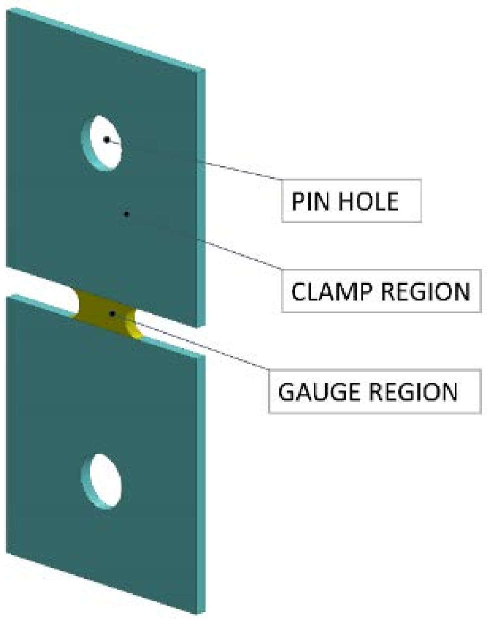


Figure 5.1 *The sample geometry that is considered in the optimization work presented in this chapter.*

As it is possible to see in Figure 5.2, two pins (in yellow) are inserted the corresponding pin holes and make the connection between the sample (in light blue) and a small clamping component (in red) possible. After that, the assembly composed of the two clamping components and the sample is glued to the bars (in grey).

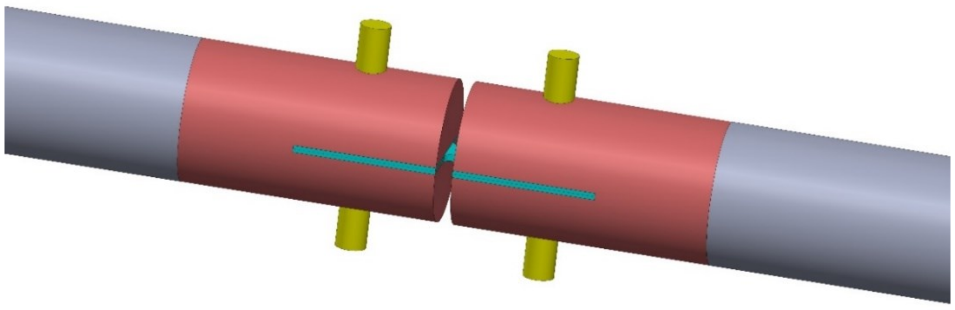


Figure 5.2 *The assembly composed of the sample (in light blue), the pins (in yellow), the clamping components (in red) and the bars (in grey).*

When the tensile experiment is performed, the transmitted wave generating at the input bar-sample interface is propagated through the sample and travels through the gauge region, loading it. In general, the shape of the gauge region affects the global value of stress triaxiality that can be obtained. The particular geometry shown in Figure 5.1 makes it possible to reach high values, especially at the centre of the sample.

As already mentioned, the goal of defining a single correlation between the stability of retained austenite and stress triaxiality makes necessary that the latter remains as constant as possible both during the duration of the experiment and in a specific space region, to be later scanned through XRD techniques and from now on called "XRD volume". Because of the characteristics of the instrumentation available for this kind of analysis, an accurate investigation of the microstructure requires at least 0.4 mm of material to be considered, therefore the XRD volume is chosen to be a cube of material of side 1 mm located at the centre of the sample, as shown in Figure 5.3, because there the highest values of stress triaxiality occur.

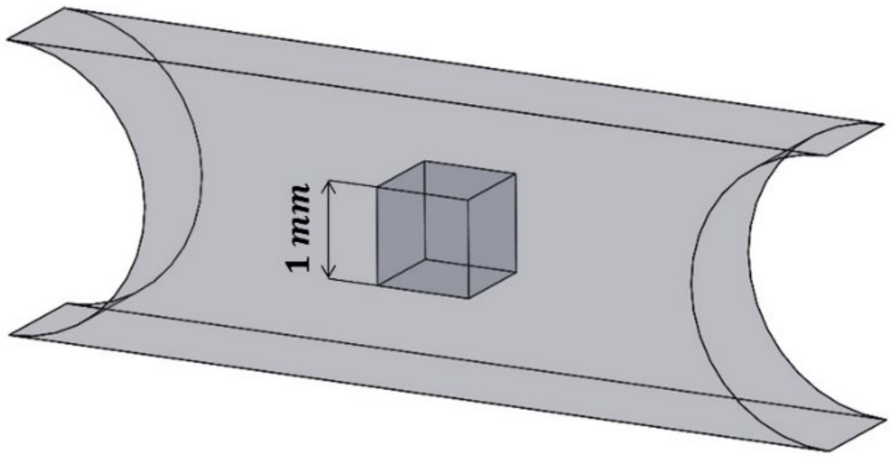


Figure 5.3 *The gauge region with the cube of material on which the optimization process focuses.*

The optimal solution, that is the one characterised by values of stress triaxiality satisfying the aforementioned requirements, is searched among different combinations of two geometric parameters: the width W of the gauge region and the radius R of the fillet, both shown in Figure 5.4.

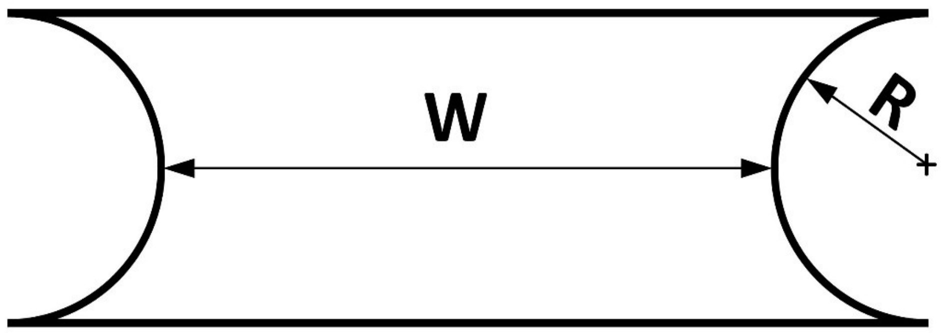


Figure 5.4 *Gauge region with its width W and the fillet radius R.*

Finally, since stress triaxiality is a function of the equivalent plastic strain ε^p, the comparison between different combinations of W and R has to be made with reference to the same level of ε^p, that is again chosen according to physical considerations. In fact, because it has been experimentally observed that most of the austenite to martensite transformation occurs in the 0 ÷ 5% range of strain, the results coming from the different combinations of W and R are compared as long as the mean value of equivalent plastic strain in the XRD volume is lower than 10%.

5.2 Methods

The developed optimization process is based on the synergy between MATLAB R2019a and Abaqus FEA 6.14 and the corresponding scripts are reported in the appendix. The procedure followed in the search for the optimal combination of width W and radius R is shown in Figure 5.5.

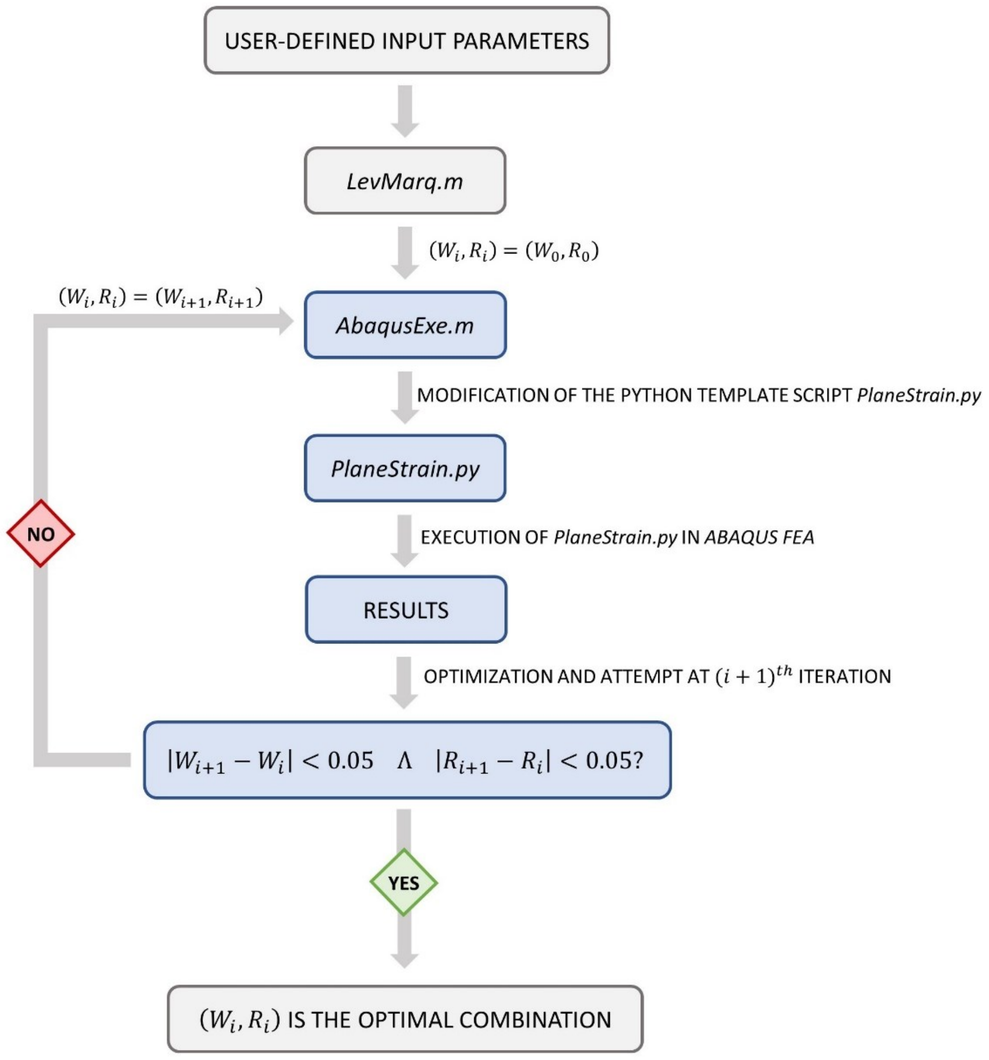

USER-DEFINED INPUT PARAMETERS
LevMarq.m
$(W_i, R_i) = (W_0, R_0)$
$(W_i, R_i) = (W_{i+1}, R_{i+1})$
AbaqusExe.m
MODIFICATION OF THE PYTHON TEMPLATE SCRIPT PlaneStrain.py
PlaneStrain.py
EXECUTION OF PlaneStrain.py IN ABAQUS FEA
RESULTS
OPTIMIZATION AND ATTEMPT AT $(i+1)^{th}$ ITERATION
NO
$|W_{i+1} - W_i| < 0.05 \;\; \wedge \;\; |R_{i+1} - R_i| < 0.05?$
YES
(W_i, R_i) IS THE OPTIMAL COMBINATION

Figure 5.5 *Procedure followed in the optimization process. The blue frames refer to a generic ith iteration.*

Firstly, the user defines some input parameters, among which the most relevant ones are the starting point (W_0, R_0) and the upper and lower bounds (W_{max}, R_{max}) and (W_{min}, R_{min}), defining the range to be investigated in the optimization. Secondly, a MATLAB script called *LevMarq.m* receives these input parameters and starts the optimization process by setting (W_0, R_0) as the first combination to be investigated. Then, the first iteration, whose steps are represented in blue in Figure 5.5, is carried out. Another MATLAB script, called *AbaqusExe.m*, modifies a pre-existing Python template script called *PlaneStrain.py*, according to the user-defined input parameters and to the combination (W_0, R_0) to be investigated. The script *PlaneStrain.py* is later executed in Abaqus FEA and the results of the Finite Element Method (FEM) analysis are obtained. Based on them, *LevMarq.m* carries out the optimization by means of Levenberg-Marquadt algorithm (LMA), which is presented in §5.2.3 and aims to minimize an objective function. Therefore, the jump to the next iteration by changing the values of (W_0, R_0) to (W_1, R_1) is attempted: if both step differences $|W_1 - W_0|$ and $|R_1 - R_0|$ are lesser than 0.05, the process ends because (W_0, R_0) is found to be the optimal combination, otherwise the next iteration takes place with the new combination (W_1, R_1).

Since the MATLAB function in *LevMarq.m*, performing the optimization and presented in §5.2.3, does not allow to control the precision of W and R, the optimal combination coming from the process has five decimal digits. This is not physically meaningful because the accuracy of the machine tool that cuts the steel sheet to get the sample is approximately 0.1 mm. For this reason, at the end of the process shown in Figure 5.5, a loop cycle investigates, in the neighbourhood of the optimal solution, the combinations of W and R with one decimal digit only.

5.2.1 Preprocessing in Abaqus FEA

As discussed in the previous section §5.2, for each algorithm iteration a FEM analysis is carried out through the execution in Abaqus FEA of the Python script *PlaneStrain.py*. In the current subchapter, all the relevant features of this analysis are reported, following the order of the different modules in the FEM software.

Firstly, only the gauge region, the area of interest, is modelled, in order to speed up the FEM analysis. In this way, the whole clamp region is not taken into account but it is possible to prove that the effect on the results is neglectable if compared to the case in which the whole sample is modelled. The only possible issue in not considering the clamp region is that in this part of the sample stress concentration can occur in two different areas, shown in Figure 5.6.

1. The roots of the pin hole, represented in black.
2. The fillet roots at the transition zone from the clamp to the gauge region, represented in red.

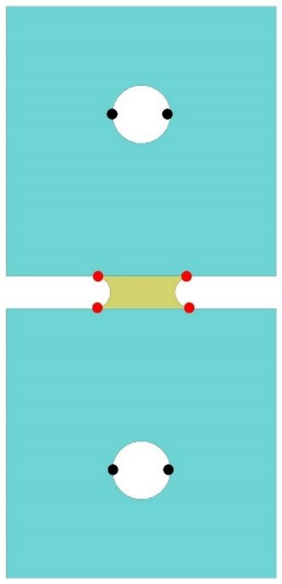


Figure 5.6 *The whole sample with the areas of the clamp region in which stress concentration can occur.*

However, with some FEM analyses considering the whole sample, it is possible to observe that that highest stress always occurs in the gauge region and not in these two areas, so the clamp region loses its relevance and can be neglected.

Regarding now the gauge region, the three symmetries make possible to model just one-eighth of the sample, so that the FEM analysis is further speeded up. The modelled part is shown in Figure 5.7, where the three symmetries are with respect to the XY, YZ and XZ-plane, respectively.

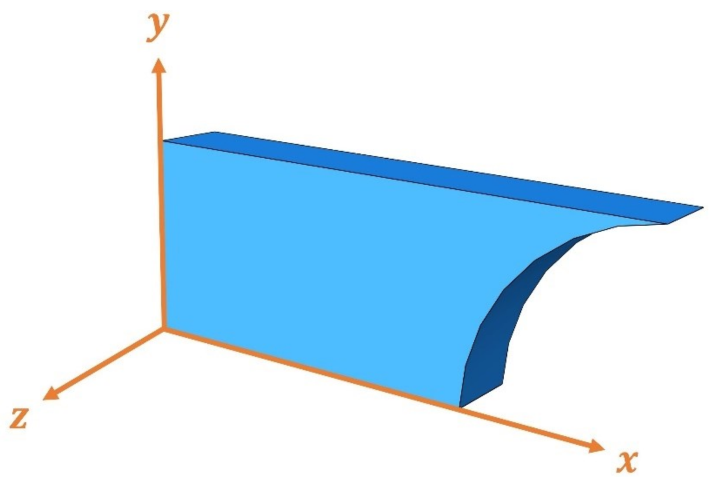


Figure 5.7 *The modelled part, corresponding to one-eighth of the whole gauge region.*

Subsequently, the material has to be defined. Among all the different steels to be investigated in the *Dynaustab* project, the one with the highest work-hardening coefficient is considered, assuming that in such a condition a maximum strain-induced transformation from austenite to martensite occurs. Thus, the choice falls on a medium-Mn steel that has been subjected to a double annealing heat treatment process at the temperatures of 900 °C and 720 °C, respectively.

The experimental data obtained for the material through a static tensile test consist of an engineering stress-strain curve. However, since the conversion to the true curve is only valid up to necking, the experimental data are processed following three different steps which aim to build the curve also after necking.

1. The Young's modulus is determined through a least squares linear regression in a limited part of the experimental curve, where the behaviour of the material is assumed to be perfectly linear.
2. The offset yield point is taken as the one at which the plastic strain reaches 0.2 %, so that the plastic part of the curve can be isolated.
3. This plastic part up to necking is modelled by means of the Swift hardening model, whose fundamental equation is

$$\sigma_p = A(\varepsilon_0 + \varepsilon_p)^n, \tag{5.4}$$

where σ_p is the true stress, ε_p the true plastic strain and A, ε_0, n constants for a particular material, usually determined through uniaxial tensile tests.

Since A, ε_0 and n are not available from the tensile test, they are first arbitrarily set and then determined by making use of a least squares regression between the experimental curve up to necking and the curve given by the Swift hardening model. Hence, (5.4) becomes the following.

$$\sigma_p = 2545.680 \cdot (0.019 + \varepsilon_p)^{0.426} \tag{5.5}$$

Finally, the curve is built from (5.5) for values of true plastic strain ε_p up to 1.5, assuming the material to fail at a true plastic strain beyond 100 %. The benefits of such a process is shown in Figure 5.8, reporting the experimental curve and the curve obtained from (5.5), where both are represented until a level of ε_p equal to 18.34 %, corresponding to necking.

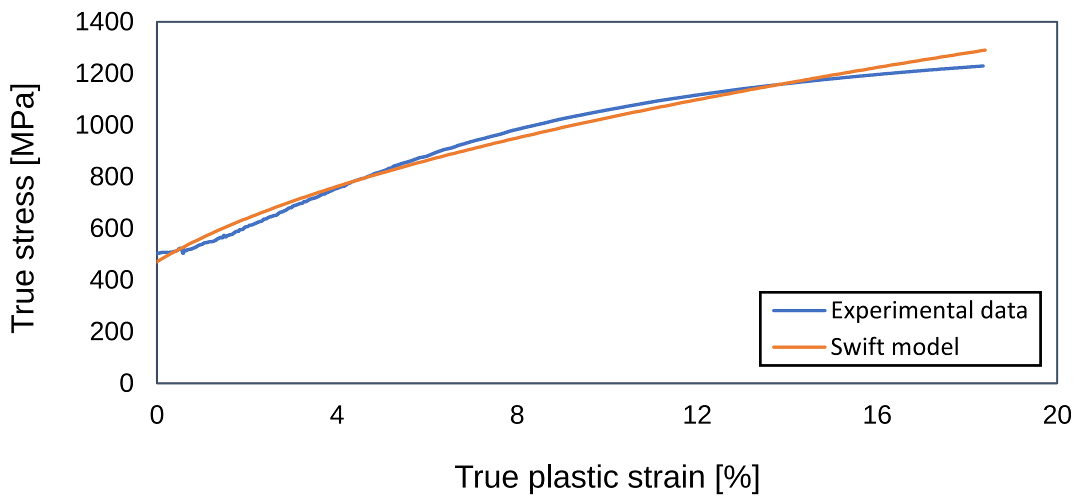


Figure 5.8 *True stress as a function of true plastic strain up to necking. The curves are obtained from the Swift model of (5.5) and from the experimental data, respectively.*

It is possible to observe how the curve obtained from the Swift model exhibits continuity over the whole range of true plastic strain ε_p, whereas for the curve coming from experimental data this condition is not always satisfied, especially for lower values of ε_p. If the latter were used in Abaqus FEA, it would badly affect the FEM analysis.

Then, a static standard analysis is defined and the boundary conditions are set on the model. With reference to Figure 5.7, in addition to the three aforementioned symmetries, displacement is applied to the superior face along the positive y-direction, and this face is also bounded not to move along the x-direction. This particular boundary condition derives from the fact that in the real-case scenario, the x-displacement of the clamp region is partly constrained because of the way the sample is clamped to the bars, as shown in Figure 5.2. As this region is not modelled in the FEM analysis, it is assumed that this constraint affects the superior face of the gauge region, too.

Finally, as shown in Figure 5.9, the mesh is refined at the centre of the sample, because there the highest values of stress triaxiality can be observed.

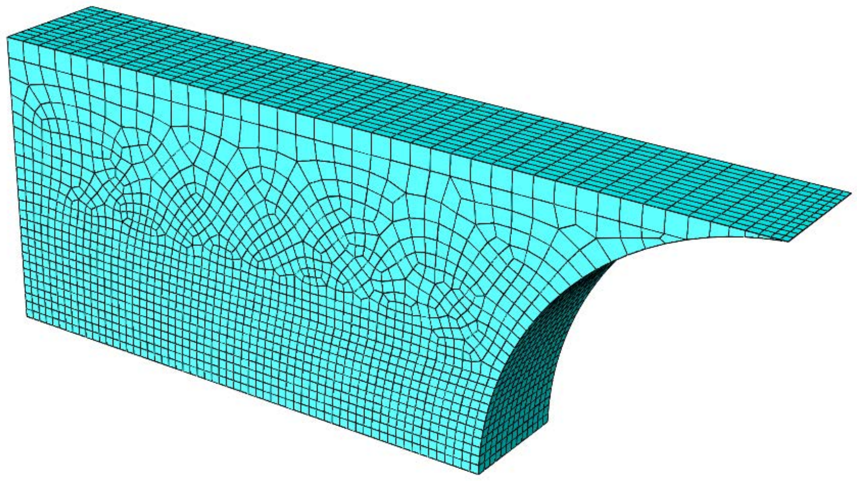


Figure 5.9 *FEM model with mesh refinement at the centre of the sample.*

5.2.2 Post-processing in Abaqus FEA

After the analysis is completed, results are extracted and processed.

First, stress triaxiality is obtained at the J evenly spaced points of the XRD volume and at each of the K time instants considered. Then, in order to link the particular combination of W and R to a single value of stress triaxiality, the quantity $\bar{T}$ is defined as the mean value of $T_{j,k}$ among all the K time instants and all the J points of the XRD volume.

$$\bar{T} = \frac{1}{K \cdot J} \sum_{k=1}^{K} \sum_{j=1}^{J} T_{j,k} \tag{5.6}$$

- $T_{j,k}$ is the value of stress triaxiality at a generic jth point of the XRD volume and at the kth time instant.
- K is set equal to 25.
- J is set equal to 1331.

Subsequently, in order to plot the stress triaxiality as a function of equivalent plastic strain, the quantities $\bar{T}_k$ and $\overline{\varepsilon_k^p}$ are evaluated. They are defined as the mean value of $T_{j,k}$ and $\varepsilon_{j,k}^p$, respectively, among all the J points of the XRD volume and at the k^{th} time instant.

$$\bar{T}_k = \frac{1}{J} \sum_{j=1}^{J} T_{j,k} \tag{5.7}$$

$$\overline{\varepsilon_k^p} = \frac{1}{J} \sum_{j=1}^{J} \varepsilon_{j,k}^p \tag{5.8}$$

Another important quantity is $\bar{T}_j$, mean value of stress triaxiality of the j^{th} point of the XRD volume among all the K time instants.

$$\bar{T}_j = \frac{1}{K}\sum_{k=1}^{K} T_{j,k} \tag{5.9}$$

Finally, the objective function OF to be minimized can be defined as

$$OF = \sum_{k=1}^{K}\sum_{j=1}^{J}(T_{target} - T_{j,k})^2, \tag{5.10}$$

where T_{target} is the target value of stress triaxiality. Since one of the optimization aims is to obtain $T_{j,k}$ as high as possible, T_{target} is set equal to 0.8, a value that, however, is not achievable by any combinations of W and R.

Because of the definition of the objective function in (5.10), its minimization decreases the dispersion of $T_{j,k}$ from T_{target}, both over time and in the XRD volume. However, it is not guaranteed that this optimization process simultaneously minimizes the dispersion of $T_{j,k}$ from $\bar{T}_k$ and $\bar{T}_j$, respectively, which is the reason why two indicators, expressing these two dispersions phenomena, are evaluated by means of the definition of standard deviation, in order to monitor this other aspect of the problem.

$$ind_{space} = \sum_{k=1}^{K}\sqrt{\frac{1}{J-1}\sum_{j=1}^{J}(\bar{T}_k - T_{j,k})^2} \tag{5.11}$$

$$ind_{time} = \sum_{j=1}^{J}\sqrt{\frac{1}{K-1}\sum_{k=1}^{K}(\bar{T}_j - T_{j,k})^2} \tag{5.12}$$

5.2.3 Algorithm

In Figure 5.5 the optimization procedure has been discussed, with a particular focus on how the different MATLAB and Python scripts interact with each other. However, few things have been mentioned regarding the algorithm itself and the way it works in jumping from the i^{th} to the $(i+1)^{th}$ iteration, so a deeper view is provided in the current subchapter.

The optimization algorithm is developed in the MATLAB script *LevMarq.m*, which makes use of *lsqnonlin* function, already integrated in MATLAB Optimization Toolbox. This function is based on Levenberg-Marquardt algorithm (LMA), typically used in minimization of non-linear functions expressed as residual sum of squares and dependent on more parameters [21]. LMA combines two numerical minimization algorithms: the gradient descent method and the Gauss-Newton method, behaving more like the former when the parameters are far from optimal and more like the latter when the parameters are close to optimal. In problems with multiple minima, LMA is more likely to find the global minimum if the starting point is close to the solution.

In the current case, the objective function to be minimized is (5.10) and the parameters on which it depends are the width W and the radius R. With reference to a generic i^{th} iteration, in Figure 5.10 the steps used by the solver to jump to the $(i+1)^{th}$ iteration are shown.

Firstly, the objective function OF corresponding to the i^{th} iteration is evaluated through a FEM analysis. Then, four other FEM analyses are carried out in the neighbourhood of (W_i, R_i) and the correspondent objective functions are evaluated. In particular, the parameters combinations that are investigated are $(W_i \pm 0.1\ mm, R_i)$ and $(W_i, R_i \pm 0.1\ mm)$. Based on the values of the four objective function, the Jacobian matrix, needed for LMA, is approximated through the finite-difference method and LMA is able to jump to the $(i+1)^{th}$ iteration through the definition of new parameters (W_{i+1}, R_{i+1}). Therefore, the whole iteration consists of five FEM analyses.

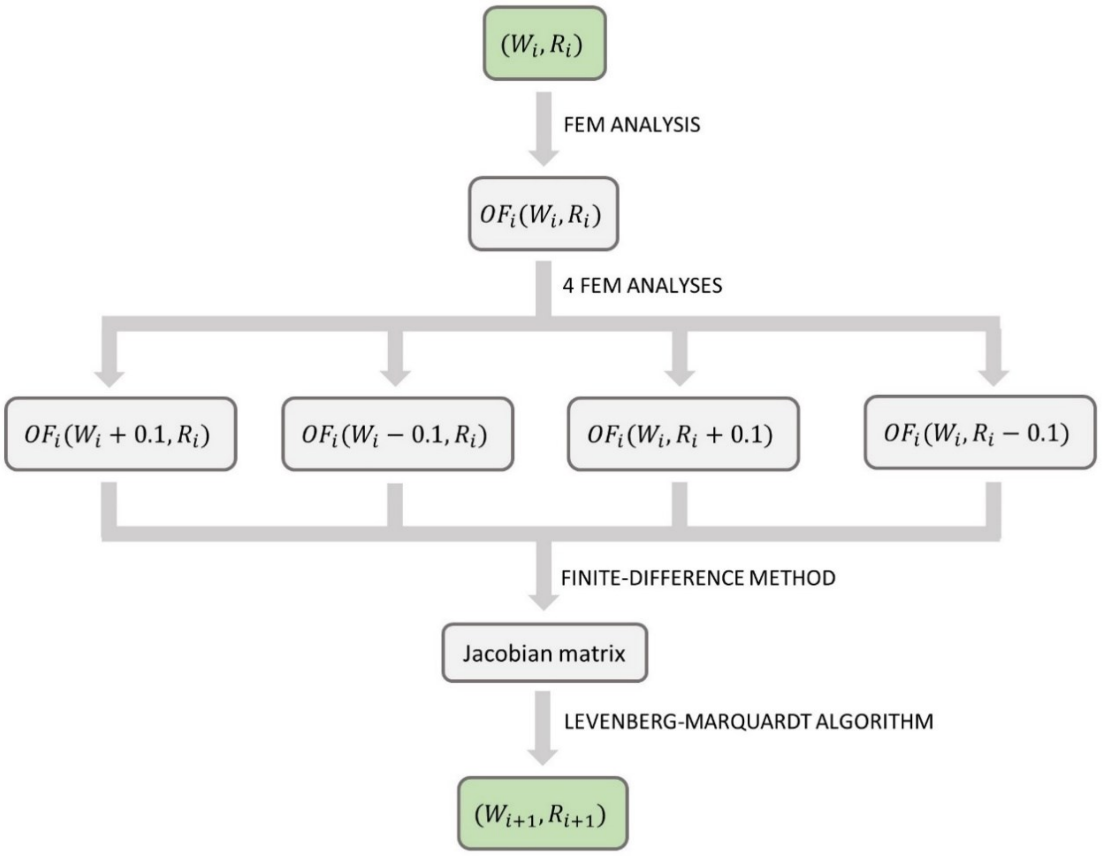


Figure 5.10 *Steps of LMA to jump from the ith to the (i+1)th iteration.*

The optimal combination of W and R is searched in between an upper bound (W_{max}, R_{max}) and a lower bound (W_{min}, R_{min}). The determination of such bounds is carried out by detecting, through some FEM analysis previous to the optimization process, the range of W and R guaranteeing high values of stress triaxiality. However, some particular conditions must be satisfied in this choice.

— From experimental evidence, the maximum longitudinal force that loads the sample tested with the Hopkinson Bar is in the range of 8 ÷ 10 kN. Since the greater its width W, the higher the longitudinal force, the maximum width W_{max} is chosen according to this requirement.

— It is better to avoid dimensions that might lead to heat affected zones, which can locally change the microstructure. For this reason, the minimum radius R_{min} cannot be lesser than a threshold value.

From all these considerations, it is set $(W_{min}, R_{min}) = (2.0, 0.8)$ and $(W_{max}, R_{max}) = (12.0, 5.0)$.

5.3 Results and discussion

As discussed in the previous subchapter §5.2.3, LMA is more likely to find the global minimum if the starting point is close to the optimal. For this reason, three optimizations are carried out with the same input parameters except for the starting point (W_0, R_0), in order to check if the solver returns the same optimal solution in all the cases. The three starting points are chosen as follows.

$$(W_0^1, R_0^1) = (W_{min}, R_{min}) \tag{5.13}$$

$$(W_0^2, R_0^2) = (W_{max}, R_{max}) \tag{5.14}$$

$$(W_0^3, R_0^3) = \left(W_{min} + \frac{W_{max} - W_{min}}{2}, R_{min} + \frac{R_{max} - R_{min}}{2}\right) \tag{5.15}$$

The three curves reporting the objective function OF as a function of the iteration number are shown in Figure 5.11.

It is possible to observe that, despite the different starting points, all the optimizations converge to the same value of OF and after the same number of iterations. Moreover, this value corresponds to the same particular combination of width and radius (W_{opt}, R_{opt}).

$$\left(W_{opt}, R_{opt}\right) = (12.0, 0.8)\ [mm] \tag{5.16}$$

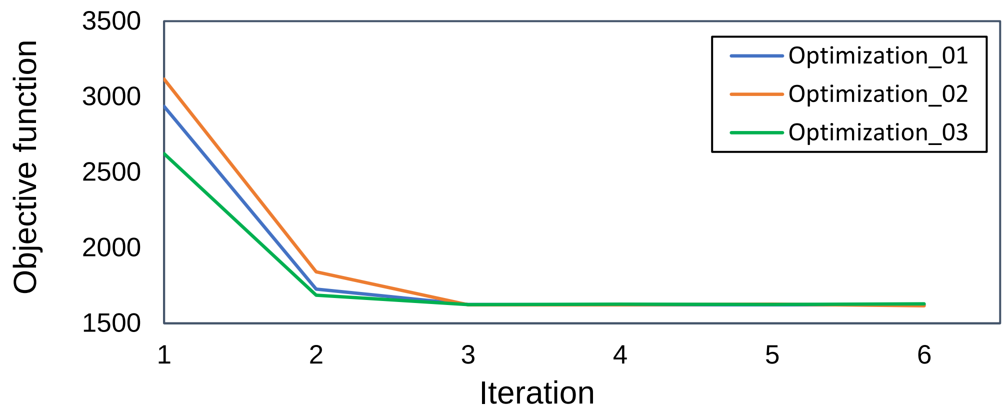


Figure 5.11 *Objective function as a function of the iteration number for the three optimizations carried out.*

In Figure 5.12 the same curves of Figure 5.11 are shown but, instead of the objective function OF, the global value of stress triaxiality $\bar{T}$ of (5.6) is plotted on the y-axis. The three optimizations converge to the same value of $\bar{T}$, equal to 0.573.

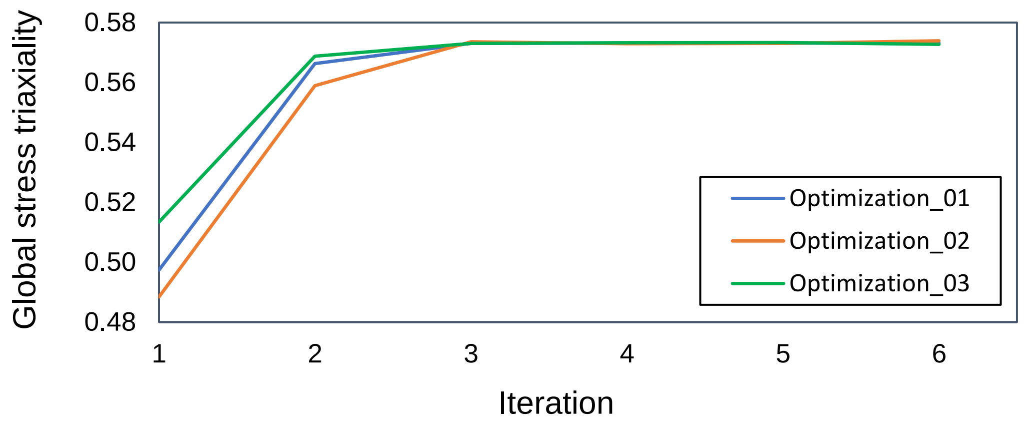
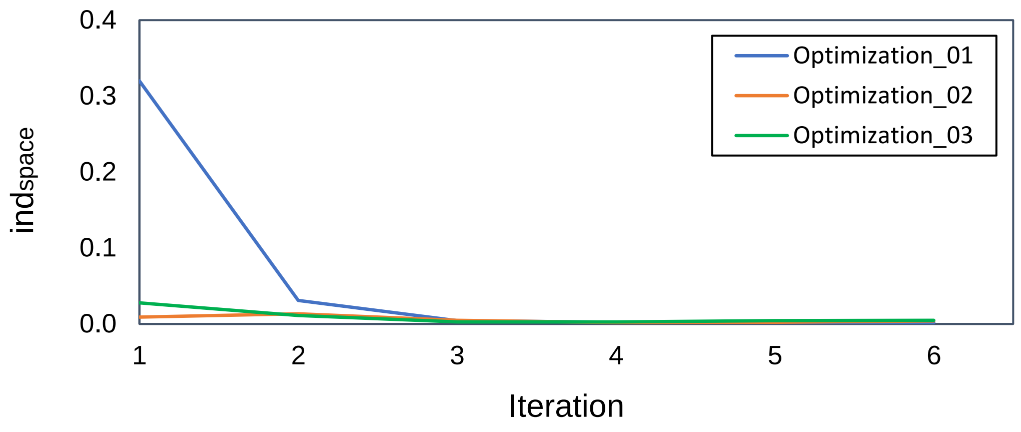


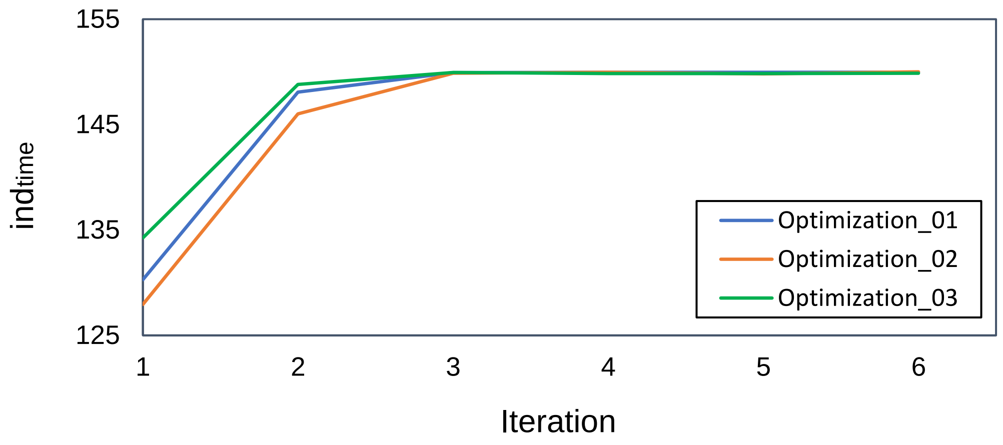
Figure 5.12 *Global value of stress triaxiality as a function of the iteration number for the three optimizations carried out.*

Since the trends of $\bar{T}$ and OF are opposite, that is when the latter decreases the former increases, the minimization of the objective function OF effectively corresponds to the maximization of the global value of stress triaxiality $\bar{\bar{T}}$, as previously assumed.

As already discussed §5.2.2, the primary goal in the minimization of the objective function is to decrease the dispersion of $T_{j,k}$ from T_{target} as less as possible, both over time and in the XRD volume. However, this does not necessarily imply the minimization of the dispersions of $T_{j,k}$ from $\bar{T}_k$ and $\bar{T}_j$, respectively. This is observable in Figure 5.13, where the indicators of (5.11) and (5.12), needed to quantify these two dispersions, are plotted as functions of the iteration number.



(a)



(b)

Figure 5.13 *The space (a) and time (b) indicators as functions of the iteration number for the three optimizations carried out.*

It is possible to observe that if ind_{space} is effectively minimized, ind_{time} increases as the optimization process goes on. Therefore, the optimal geometry exhibits the highest global stress triaxiality achievable, a low dispersion in the XRD volume, but a pronounced dispersion over time. This lies in the fact that the definition of the objective function favours the maximization of stress triaxiality values rather than the minimization of the dispersion of those values both in the XRD volume and over time. Thus, further investigation is needed to simultaneously satisfy all these three requirements.

To give a clearer view of the optimization process, a three-dimensional graph, reporting the objective function OF as a function of the pairs (W, R), is shown in Figure 5.14 for the first optimization. The black circles refer to the iterations, whereas the yellow ones correspond to the function evaluations needed to determine the Jacobian matrix. As mentioned before, the observed trend is the same for the two other optimizations.

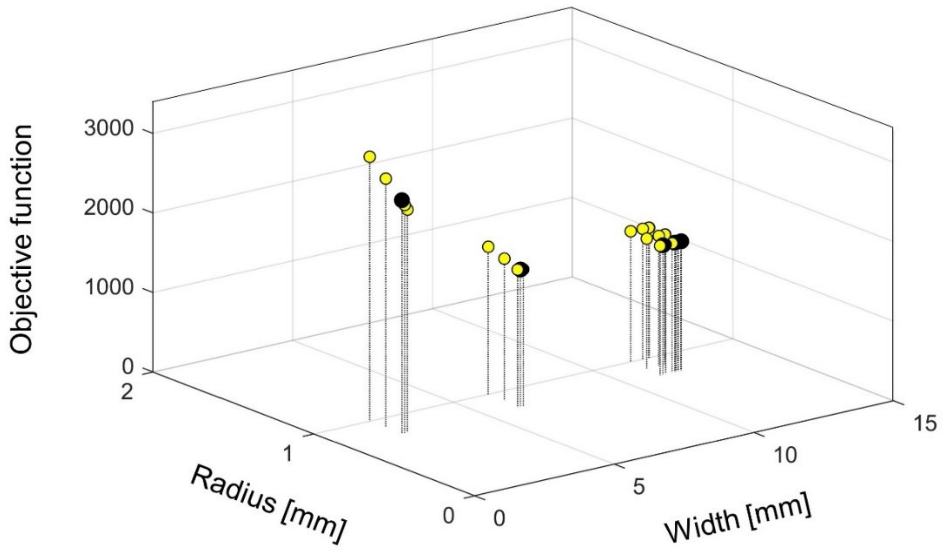
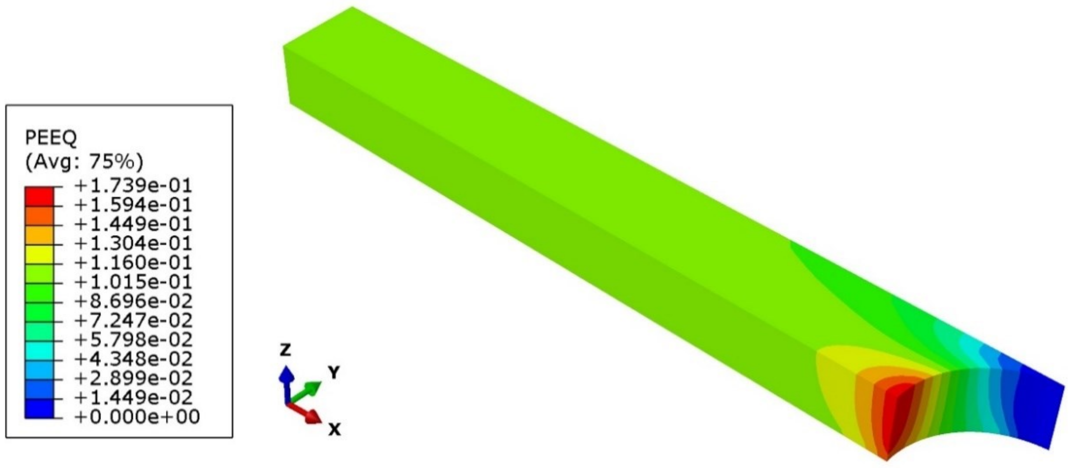
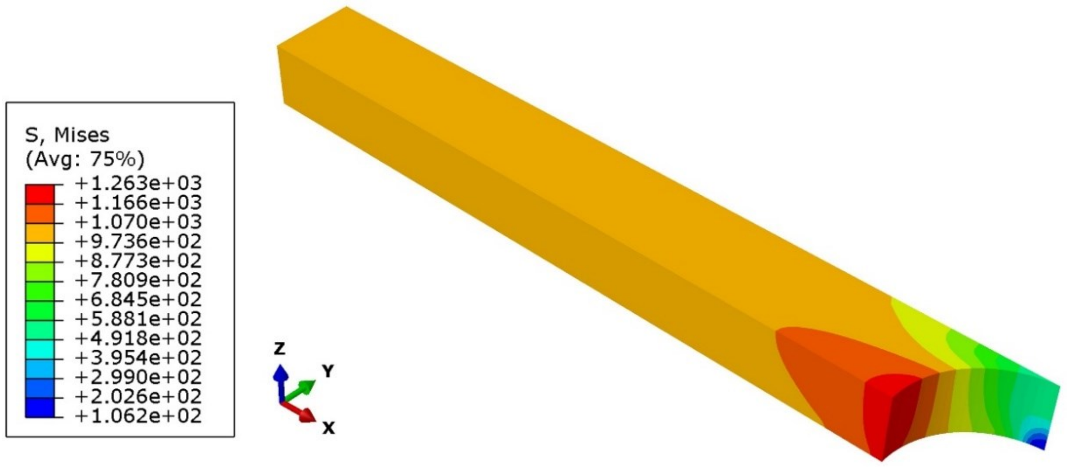


Figure 5.14 *Objective function as a function of width and radius. The black circles refer to the iterations, the yellow ones to the function evaluations needed to determine the Jacobian matrix.*

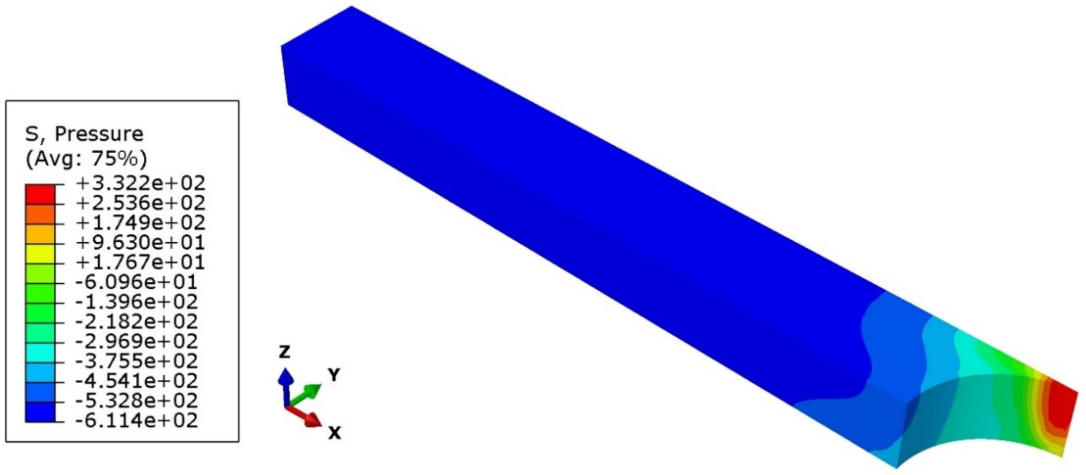
As reported in (5.16), the optimal values W_{opt} and R_{opt} are nothing but W_{max} and R_{min}, respectively. Therefore, a particular trend linking the geometry of the sample to the global value of stress triaxiality can be defined, that is the greater W and the lesser R, the greater $\bar{T}$. In Figure 5.15 and Figure 5.16 the contour plots of plastic equivalent strain and von Mises equivalent stress are shown for two different combinations of width and radius, the former corresponding to the optimal solution, the latter to the combination $(W, R) = (3, 4)$ $[mm]$.



(a)

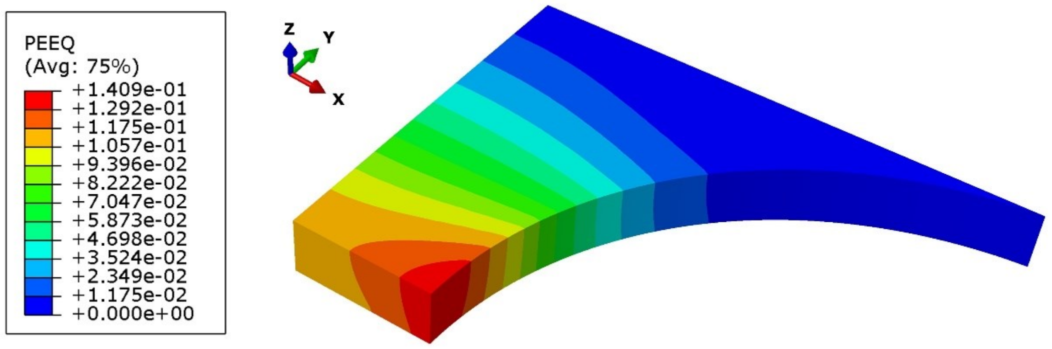


(b)

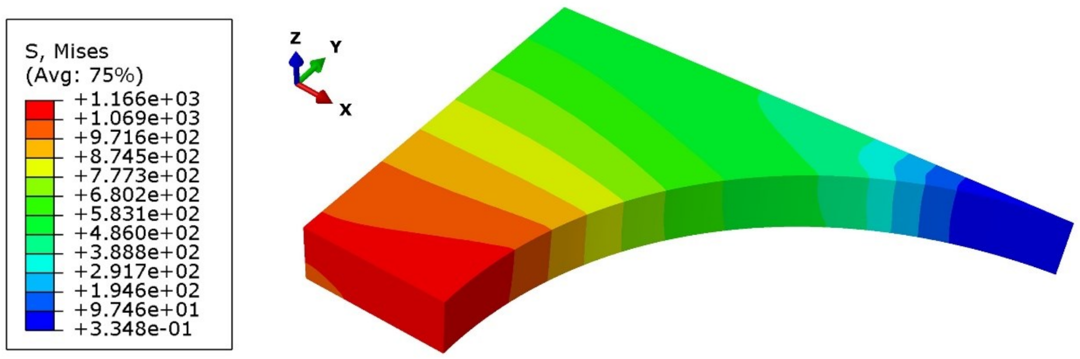


(c)

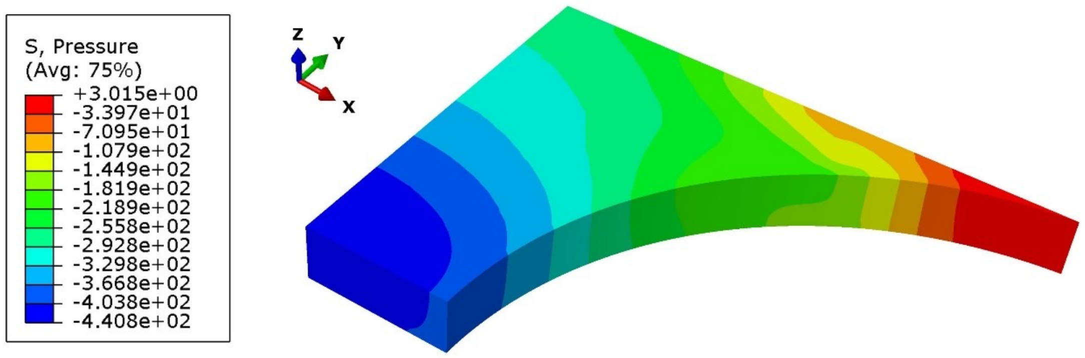
Figure 5.15 *Contour plot of equivalent plastic strain (a), von Mises equivalent stress (b) and the opposite hydrostatic stress (c) for the FEM analysis with W=10 mm and R=0.8 mm. The results correspond to a mean equivalent plastic strain in the XRD volume equal to 10 %.*



(a)



(b)



(c)

Figure 5.16 *Contour plot of equivalent plastic strain (a), von Mises equivalent stress (b) and the opposite of hydrostatic stress (c) for the FEM analysis with W=3 mm and R=4 mm. The results correspond to a mean equivalent plastic strain in the XRD volume equal to 10 %.*

In general, for a given width, if the radius decreases, the stress concentration at the notch becomes more relevant and the stress and strain gradients along the thickness in this region prove it**,** as shown in Figure 5.15. For this reason, the stress and strain gradients along the three directions concentrate at the notch, which promotes stress and strain heterogeneity and decreases von Mises equivalent stress at the centre. Moreover, because the displacement of upper face is constrained along the x-direction, a x-component of stress develops at the centre, increasing hydrostatic stress. The global result is an increase in stress triaxiality. The same effects are obtained also through an increase in width, for a given radius.

Finally, during the deformation of the sample it is possible to observe that $\bar{T}_k$, mean value of stress triaxiality in the XRD volume, is approximately constant for $\overline{\varepsilon_k^p}$ greater than 2 %, as shown in Figure 5.17.

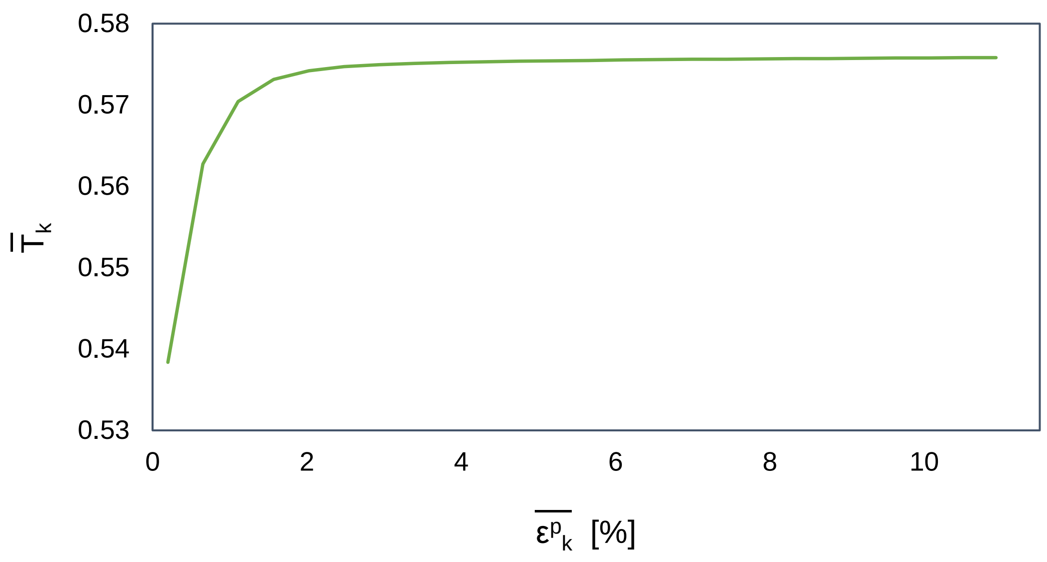


Figure 5.17 *Mean stress triaxiality in the XRD volume as a function of mean equivalent plastic strain in the XRD volume.*

6. Conclusions

In the work presented in this thesis, two investigations, one experimental and one numerical, are carried out in order to improve the quality of dynamic experiments with the Split-Hopkinson Tensile Bar (SHTB).

The experimental study consists in calibrating the strain gauges mounted on the input and the output bar. A direct calibration method is preferred to an indirect one because of its higher precision. Each of the two bars is statically loaded in compression by a hydraulic cylinder and the strain gauge signals are compared with the values of a load cell attached to the bar and considered as reference. The longitudinal forces on the bar that are considered, in the range of 0 ÷ 20 kN, correspond to those observed during typical dynamic tests.

The results of the first experiment on the output bar show that the resistive and semiconductor strain gauges are affected by a negative error with respect to the load cell around 5 % and 22 %, respectively, thus underestimate the measurement. As far as the former are concerned, the reason of this deviation is to be found in the mounting process, when small misalignments between the strain gauge measuring direction and the longitudinal axis of the bar can lead to considerable errors. On the other hand, with regard to the latter, the Wheatstone bridge to which they are connected is not resistively balanced, so non-linearity develops and is amplified by the high value of the gauge factor, typical of semiconductor gauges. Furthermore, the supply voltage of the bridge, equal to 5 V, can excessively overheat the strain gauges, whose temperature sensitivity is considerable. The results obtained for the input bar show that the two pairs of resistive strain are still affected by a negative error with respect to the load cell, around 6% and 8%, respectively. As already discussed for the resistive strain gauges on the output bar, the reason for this deviation is to be found in possible misalignments during the mounting process, however, this does not justify the difference, although small, between the signals of the two pairs of strain gauges.

In order to investigate repeatability, a new test is performed on the bars keeping all the conditions unchanged, except for temperature, humidity and light, which are not controlled. Regarding the output bar, although the results obtained for the resistive strain gauges are consistent with those of the first experiment, the relative error of the semiconductor strain gauges, equal to 24 %, deviates from the one observed previously, equal to 22 %. The reason for this deviation lies in the fact that this instrument is highly sensitive to slight changes in the three aforementioned parameters. On the other hand, as far as the input bar is concerned, the resistive strain gauges exhibit neglectable sensitivity, since the results obtained are consistent with the previous ones.

Therefore, further investigation is necessary to better analyse the reason for the discrepancies observed in the measurements. With regard to resistive strain gauges, it is advisable to mount them again on their corresponding bar, making sure that its longitudinal axis coincides with the measurement direction of the instrument. Instead, for semiconductor strain gauges, the resistive balance of Wheatstone bridge to which they are connected and a lower supply voltage may reduce the observed errors.

In the numerical study, geometric optimization of the sample is carried out. The aim of this work is to obtain high values of stress triaxiality and keep them as constant as possible during the dynamic test and in a specific region at the centre of the sample. These requirements can be satisfied by the minimization of a properly defined objective function. The optimization, performed using Abaqus FEA and automated by the development of MATLAB and Python scripts, consists in analysing the values of this objective function, dependent on stress triaxiality, as width and fillet radius of the sample change. The results obtained show how the optimization process converges to the same geometry even if starting from different combinations of width and radius. Precisely, the optimal geometry is characterised by a large value of width and a small value of radius. This is due to the fact that a decrease in the fillet radius leads to stress concentration at the notch, decreasing the equivalent von Mises stress at the centre of the sample. In addition, an increase in width makes the stress component along that dimension more relevant at the centre of the sample, which increases the hydrostatic component of stress. This dual effect results in an increase in stress triaxiality. However, in the optimal solution, the dispersion of this quantity over time is not effectively minimized, which is caused by the definition of the objective function, favouring values of stress triaxiality which are high rather than constant over time and in the region of interest. Thus, further investigation is needed in order to simultaneously satisfy all three requirements.

References

[1] European Parliament and Council of the European Union, *Regulation (EU) 2019/631*, Official Journal of the European Union, 2019, pp. 20-21.

[2] European Parliament and Council of the European Union, *Regulation (EC) No 443/2009*, Official Journal of the European Union, 2009, pp. 4-5.

[3] Zhang S. et al., *Well-to-wheels energy consumption and emissions of electric vehicles: Mid-term implications from real-world features and air pollution control progress*, Elsevier, 2016, pp. 371-373.

[4] Lutsey N., *Review of technical literature and trends related to automobile mass-reduction technology*, 2010, p. 38.

[5] Davies G., *Materials for Automobile Bodies*, Elsevier, 2003, p. 1.

[6] Schmitt J.H., Iung T., *New developments of advanced high-strength steels for automotive applications*, Comptes Rendus Physique, 2018

[7] Bittence J.C., *Dual-phase steels promise higher strength plus formability*, Mater. Eng. 87, 1978, pp. 39-42.

[8] G. Wassermann, *Untersuchungen an einer Eisen-Nickel-Legierung über die Verformbarkeit während der γ-α-Umwandlung*, Archiv Eisenhüttenwesen, vol.10, no.7, 1939, pp. 321-325.

[9] Speer J.G. et al., *Carbon partitioning into austenite after martensite transformation*, 2003, pp. 2611-2622.

[10] Bleck W. et al., *The TRIP effect and its application in cold formable sheet steels*, Steel Research International 88 (10), 2017, no. 1700218.

[11] Jacques P. et al., *On the role of martensitic transformation on damage and cracking resistance in TRIP-assisted multiphase steels*, Acta Materialia 49, 2000, pp. 139-152.

[12] Jacques P. et al., *Multiscale mechanisms of the TRIP- assisted multiphase steels: I. Characterization and mechanical testing*. Acta Materialia 55, 2007, pp. 3681-3693.

[13] Blondé R. et al., *High-energy X-ray diffraction study on the temperature-dependent mechanical stability of retained austenite in low-alloyed TRIP steels*, Acta Materialia 60, 2012, pp. 565-577.

[14] Zou D.Q. et al, *Temperature and strain rate dependent deformation induced martensitic transformation and flow behavior of quenching and partitioning steels*, Materials Science & Engineering A 680, 2017, pp. 54-63.

[15] Gronostajski Z. et al., *The effect of the strain rate on the stress-strain curve and microstructure of AHSS*, Journal of Materials Processing Technology 242, 2016, pp. 246-259.

[16] Chen W., Song B., *Split Hopkinson (Kolsky) Bar: Design, Testing and Applications*, Springer, 2011, pp. 1-17.

[17] Graff K.F., *Wave Motion in Elastic Solids*, Dover Publications, 1991, pp. 75-79.

[18] Smith C.S., *Piezoresistive effect in silicon and germanium*, Physical Review 94, 1954, pp. 42-49.

[19] Oliveira I. et al., *Inverse characterization of material constitutive parameters for dynamic applications*, Procedia Engineering, vol. 114, pp. 784-791.

[20] Wang Q. et al., *Geometry optimization of sheet specimen for the measurement accuracy improvement in the Hopkinson Bar based on intelligent algorithm*, IEEE Access, vol. 8, 2020, pp. 99655-99664.

[21] Levenberg K., *A Method for the Solution of Certain Non-Linear Problems in Least Squares*, Quarterly of Applies Mathematics, vol. 2, no. 2, 1944, pp. 164-168.

Appendix

LevMarq.m

```
%{
This script is able to perform the geometry optimization by means of
the 'lsqnonlin' function, already implemented in MATLAB. To make
this function work, the script calls another MATLAB user-defined
function, called 'AbaqusExe.m', able to run the analysis in Abaqus.

It is mandatory to set the necessary input parameters ('NECESSARY
INPUT PARAMETERS' section). If desired, it is also possible, but not
mandatory, to modify the pre-existing optional ones ('OPTIONAL INPUT
PARAMETERS' section).
%}

clear;
close all;
clc;
```

NECESSARY INPUT PARAMETERS

```
%--------------------------------------------------------------------
%Set the input parameters with consistent units of measurement (MPa,
mm, N, ...)
%--------------------------------------------------------------------

%Starting value (W0), upper (W_max) and lower (W_min) bounds for the
width 'W'
W0=2;

W_max=12;

W_min=2;
```

```
%Starting value (RAD0), upper (RAD_max) and lower (RAD_min) bounds
for the fillet radius 'RAD'
RAD0=0.8;

RAD_max=5;

RAD_min=0.8;

%Target value of stress triaxiality
trx_obj=0.8;

%Number of processors
nProc=3;

%Final displacement
uY=1;

%Path for the 'GeometryOptimization' folder
main_path= 'C:\\Users\\Claudio
Lonardi\\Desktop\\GeometryOptimization';
```

OPTIONAL INPUT PARAMETERS

```
%-------------------------------------------------------------------
%Set the input parameters with consistent units of measurement (MPa,
mm, N, ...)
%-------------------------------------------------------------------

%Thickness
Tck=1.2;

%Rounding angle [degrees]
alphaG=85;

%Young's modulus
E=204102.28;

%Poisson's ratio
ni=0.33;
```

```
%Side dimension of volume of material cube  to be analyzed with XRD
XRD=1;

%Distance between two adjacent points of XRD-volume where the
quantities of interest are evaluated
dist=0.05;

%Experiment duration
dT=0.001;

%PEEQ limit
PEEQ_lim=0.1;

%Number of elements along the semi-width at the center of the sample
nE_w =60;

%Number of elements along the semi-thickness at the center of the
sample
nE_t =12;

%Number of elements along the semi-width at the shoulder of the
sample
nE_wSh =60;

%Number of time intervals, at which the outputs quantities must be
evaluated
nInt=25;
```

LEVENBERG-MARQUARDT ALGORITHM

```
%Builds the array of the input parameters
inp=[Tck, alphaG, E, ni, XRD, dist, dT, uY, PEEQ_lim, nE_w, nE_t,
nE_wSh, nInt, nProc];

%Sets the 'GeometryOptimization' folder as the work directory
cd (main_path);

%Defines the path for the 'AbaqusData' folder, where the several
data of the FEM analysis will be saved
abaqusData_path=append(main_path, '\\AbaqusData');
```

```
%Deletes a possible pre-existing folder 'AbaqusData', containing all
the Abaqus folders that will be generated during the analyses
if isfolder('AbaqusData')==1

    rmdir AbaqusData s;
end

%Creates a new folder 'AbaqusData'
mkdir('AbaqusData');

%Deletes a possible pre-existing text file 'AlgorithmResults.txt',
containing the algorithm results data
if isfile('AlgorithmResults.txt')==1

    delete('AlgorithmResults.txt');
end

%Initializes the text file 'AlgorithmResults.txt', that will contain
the algorithm results data
fileID = fopen('AlgorithmResults.txt', 'at');

fprintf(fileID, '%s \n \n', 'ALGORITHM');

header= ["W [mm]    ", "RAD [mm]  ", "peeqMax   ", "FY_max [N]",
"indTime   ", "indSpace  ",  "objFunc   ",  "trx_avav  "];

fprintf(fileID, '%s \t %s \t %s \t %s \t %s \t %s \t %s \t %s \t \n
\n', header);

separator="-";

while strlength(separator) <= 120

    separator=append(separator, "-");

end

fprintf(fileID, '%s \n', separator);

delimiter="_";
```

```
while strlength(delimiter) <= 120

    delimiter=append(delimiter, "_");

end

fclose(fileID);

%Sets the starting point array 'par0' (2x1), input parameter for the
'lsqnonlin' function
par0=[W0, RAD0];

%Sets the lower bound array 'par_min' (2x1), input parameter for the
'lsqnonlin' function
par_min=[W_min; RAD_min];

%Sets the upper bound array 'par_max' (2x1), input parameter for the
'lsqnonlin' function
par_max=[W_max, RAD_max];

%Defines the objective function to be minimized
objFunc_mat = @(par) AbaqusExe(par, inp, trx_obj, main_path,
abaqusData_path, separator);

%Sets the options for the optimization
options=optimoptions(@lsqnonlin, 'Algorithm', 'levenberg-marquardt',
'DiffMinChange', 1e-01, 'FiniteDifferenceType', 'central',
'FunctionTolerance', 1e-04, 'StepTolerance', 5e-
02,'MaxFunctionEvaluations', 500, 'MaxIterations', 500, 'Display',
'iter');

%Runs the optimization
[par,resnorm,residual,exitflag,output] = lsqnonlin(objFunc_mat,
par0, par_min, par_max, options);
```

DETECTION OF THE OPTIMAL SOLUTION

```
%Appends to the text file 'AlgorithmResults.txt' the new section
header
```

```
fileID = fopen('AlgorithmResults.txt', 'at');

fprintf(fileID, '%s \n', delimiter);

fprintf(fileID, '%s \n \n', 'OPTIMUM DETECTION');

fprintf(fileID, '%s \t %s \t %s \t %s \t %s \t %s \t %s \t %s \t \n
\n', header);

fprintf(fileID, '%s \n', separator);

fclose(fileID);

%Sets the 'AbaqusData' folder as the work directory
cd (abaqusData_path);

%Extracts all the files in the 'AbaqusData' folder
allFiles = dir;

allFiles=allFiles(~ismember({allFiles.name},{'.','..'}));

%Extracts all the folders from 'allFiles'
allDir=allFiles([allFiles.isdir]);

%Finds the number of folders in 'AbaqusData'
nDir=length(allDir);

%Sorts the folders by date
[~,idx] = sort([allDir.datenum]);

allDir = allDir(idx);

%Extracts the folder name 'fName' correspondent to the final
iteration
fName=string({allDir(nDir-4).name});

%Extracts the values of width 'Wf' and radius 'RADf' correspondent
to the final iteration
Wf = str2double(extractBetween(fName, '_W', '_R'));

RADf = str2double(extractAfter(fName, '_R'));
```

```
%Sets the 'GeometryOptimization' folder as the work directory
cd (main_path);

%Defines the array W_det (3x1), containing the values of width in
the neighborhood of the final algorithm iteration
if round(Wf,1)==W_max

    W_det=[round(Wf,1)-0.2, round(Wf,1)-0.1, round(Wf,1)];

elseif round(Wf,1)==W_min

    W_det=[round(Wf,1), round(Wf,1)+0.1, round(Wf,1)+0.2];

else

    W_det=[round(Wf,1)-0.1, round(Wf,1), round(Wf,1)+0.1];

end

%Defines the array RAD_det (3x1), containing the values of radius in
the neighborhood of the final algorithm iteration
if round(RADf,1)==RAD_max

    RAD_det=[round(RADf,1)-0.2, round(RADf,1)-0.1, round(RADf,1)];

elseif round(RADf,1)==RAD_min

    RAD_det=[round(RADf,1), round(RADf,1)+0.1, round(RADf,1)+0.2];

else

    RAD_det=[round(RADf,1)-0.1, round(RADf,1), round(RADf,1)+0.1];

end

%Runs Abaqus to find the best combination of width and radius values
contained in W_det and RAD_det, respectively
for i=1:length(W_det)

    for j=1:length(RAD_det)
```

```
        par=[W_det(i), RAD_det(j)];

        AbaqusExe(par, inp, trx_obj, main_path, abaqusData_path,
separator);

    end
end
```

AbaqusExe.m

FUNCTION DEFINITION

```
function objFunc_mat=AbaqusExe(par, inp, trx_obj, main_path,
abaqusData_path, separator)
```

```
%{
This function receives as inputs the width 'par(0)', the fillet
radius
'par(1)', the other input parameters 'inp', the target value of
stress
 triaxiality 'trx_obj', the path for the GeometryOptimization folder
'main_path', the path for AbaqusData folder 'abaqusData_path' and
the
string 'separator', useful for the exportation of the results in a
.txt
file. Based on these inputs, the script modifies a pre-existing
basic Python script 'PlaneStrain.py', and runs it in Abaqus.

The outputs is the matrix 'objFunc_mat', representing the difference
between the target value of stress triaxiality 'trx_obj' and the
stress
triaxiality of the points in the XRD-volume over the time.
%}
```

MODIFCATION OF THE PYTHON TEMPLATE SCRIPT

```
%Defines the variables of width 'W' and radius 'RAD' for the sake of
simplicity
W=par(1);

RAD=par(2);

%Sets 'scriptName', the name of the script that will be run in
Abaqus
scriptName=append('PlaneStrain_W', num2str(W), '_R', num2str(RAD));

scriptName = replace(scriptName, '.', '-');

scriptName = append(scriptName,'.py');

%Defines the new strings of the necessary input parameters in the
'PlaneStrain.py' script
W_new=append('W=', num2str(W));

RAD_new=append('RAD=', num2str(RAD));

mainPath_new=main_path;

%Defines the array 'inpStr', containing the strings of the optional
input parameters symbols
inpStr=["Tck"; "alphaG"; "E"; "ni"; "XRD"; "dist"; "dT"; "uY";
"PEEQ_lim"; "nE_w"; "nE_t"; "nE_wSh"; "nInt"; "nProc"];

%Finds the number 'nInp' of optional input parameters
nInp=length(inpStr);

%Reads the template script 'PlaneStrain.py'
py = fileread('PlaneStrain.py');

%Initializes the 'pyNew' character array, necessary for the next
loop cycle
pyNew=py;

%Builds the 'pyNew' character array, containing the Python script
with the new other input parameters
```

```
for i=1:nInp

    old=append(inpStr(i), '=000');

    new=append(inpStr(i), '=', num2str(inp(i)));

    pyNew=strrep(pyNew, old, new);

end

%Fills in 'pyNew' with the new necessary parameters
pyNew=strrep((strrep(strrep(pyNew, 'W=000', W_new), 'RAD=000',
RAD_new)), 'mainPath=000', append("mainPath='", mainPath_new, "'"));

%Sets the 'AbaqusData' folder as the work directory
cd (abaqusData_path);

%Opens the 'PlaneStrain_W!_R!.py' script, that will be run in Abaqus
fid = fopen(scriptName, 'w');

%Writes 'pyNew' in 'PlaneStrain_W!_R!.py'
fwrite(fid, pyNew);

%Closes 'PlaneStrain_W!_R!.py'
fclose(fid);
```

SCRIPT EXECUTION

```
%Runs the 'PlaneStrain_.py' script in Abaqus
system(append('abaqus cae noGUI=', scriptName));

%Extracts 'folder_path', the path name of the 'PlaneStrain_W!_R!'
folder, where the Abaqus data for the actual combination of width
and radius have been stored
fileID=fopen('folderPath.txt');

folder_path=string(textscan(fileID, '%q', 'delimiter', '\n'));

fclose(fileID);
```

```
%Sets the 'PlaneStrain_W!_R!' folder as the work directory
cd (folder_path);

%Extracts 'XRD', the side dimension of material cube to be analyzed
with XRD
XRD=inp(inpStr=='XRD');

%Extracts 'dist', the distance between two consecutive measures of
the quantities with XRD
dist=inp(inpStr=='dist');

%Evaluates 'nP' the number of evenly spaced points in the XRD-volume
where to measure the triaxiality
nP=(XRD/2/dist+1)^3;

%Extracts 'nSteps', the number of time instants, at which the
outputs quantities must be evaluated
nSteps=inp(inpStr=='nInt')+1;
```

RESULTS IMPORTATION

```
%Imports the matrix 'trx' (nSteps x nP), containing the stress
triaxiality of all the points in the XRD-volume and for each time
instant
fileID = fopen('TRIAX.txt', 'r');

trx = fscanf(fileID, '%f');

trx = reshape(trx, [nP,nSteps])';

fclose(fileID);

%Imports the scalar 'trx_avav', containing the global mean value of
stress triaxiality
fileID = fopen('TRIAX_avav.txt', 'r');

trx_avav = fscanf(fileID, '%f');

fclose(fileID);
```

```
%Imports the scalar 'peeqMax', containing the maximum value of PEEQ
in the sample
fileID = fopen('PEEQ_max.txt', 'r');

peeqMax= max(fscanf(fileID, '%f'));

fclose(fileID);

%Imports the scalar 'FY_max', containing the maximum value of the Y-
component of the force loading the superior face of the sample at
each time instant
fileID = fopen('FORCE_Y.txt', 'r');

FY_max=max(fscanf(fileID, '%f'));

fclose(fileID);
```

OUTPUTS DEFINITION

```
%Defines the residuals matrix 'objFunc_mat' ((nSteps-2) x nP) of the
objective function 'objFunc' to be minimized
objFunc_mat= trx_obj*ones(nSteps-2, nP) - trx(3:nSteps, :);

%Evaluates the objective function 'objFunc', representing the
dispersion of stress triaxiality in the XRD-volume and over the time
objFunc= sum(sum(objFunc_mat.^2));

%Evaluates the time dispresion indicator 'indTime', representing the
standard deviation of triaxiality over the time
indTime= sum(std(trx));

%Evaluates the space dispresion indicator 'indSpace', representing
the standard deviation of stress triaxiality in the XRD-volume
indSpace= sum(std(trx, 0, 2));
```

OUTPUTS EXPORTATION

```
%Exports the objective function 'objFunc' in a .txt file
fileID = fopen('objFunc.txt','w');
```

```
fprintf(fileID,'%f', objFunc);

fclose(fileID);

%Exports the time indicator 'indTime' in a .txt file
fileID = fopen('indTime.txt','w');

fprintf(fileID,'%f', indTime);

fclose(fileID);

%Exports the space indicator 'indSpace' in a .txt file
fileID = fopen('indSpace.txt','w');

fprintf(fileID,'%f', indSpace);

fclose(fileID);
```

MOVING OF FILES

```
%Sets the 'AbaqusData' folder as the work directory
cd (abaqusData_path);

%Extracts 'folderName', the name of the 'PlaneStrain_W!_R!' folder
folderName = extractAfter(folder_path, 'AbaqusData\');

%Moves the 'PlaneStrain_W!_R!.py' script in the 'PlaneStrain_W!_R!'
folder
movefile(scriptName, folderName)

%Moves 'folderPath.txt' in the 'PlaneStrain_W!_R!' folder
movefile('folderPath.txt', folderName)

%Moves 'abaqus.rpy', the Abaqus replay file, in the
'PlaneStrain_W!_R!' folder
movefile('abaqus.rpy', folderName)

%Builds the row array 'resCurrent', containing the current results
data
```

```
resCurrent = [W, RAD, peeqMax, FY_max, indTime, indSpace, objFunc,
trx_avav];

%Appends to the text file 'AlgorithmResults.txt' the current results
data array 'resCurrent'
allFiles = dir;

nDir=sum([allFiles(~ismember({allFiles.name},{'.','..'})).isdir]);

fileID = fopen(append(main_path, '\\', 'AlgorithmResults.txt'),
'r');

fileStr=fscanf(fileID, '%s');

fileID = fopen(append(main_path, '\\', 'AlgorithmResults.txt'),
'at');

if isempty(strfind(fileStr,'OPTIMUMDETECTION'))==1

    if mod((nDir-1),5)==0

        fprintf(fileID, '%f \t %f \t %f \t %f \t %f \t %f \t %f \t
%f \n \n', resCurrent);

    elseif mod(nDir,5)==0

        fprintf(fileID, '%f \t %f \t %f \t %f \t %f \t %f \t %f \t
%f \n ', resCurrent);

        fprintf(fileID, '%s \n', separator);

    else

        fprintf(fileID, '%f \t %f \t %f \t %f \t %f \t %f \t %f \t
%f \n ', resCurrent);

    end

else

    fprintf(fileID, '%f \t %f \t %f \t %f \t %f \t %f \t %f \t %f
\n', resCurrent);

end

fclose(fileID);

%Sets the 'GeometryOptimization' path as the work directory
cd (main_path)
```

PlaneStrain.py

MODULES IMPORT

```
# coding=utf-8


from abaqus import *
from abaqusConstants import *
from driverUtils import executeOnCaeStartup
from __main__ import *
from section import *
from regionToolset import *
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *
from xyPlot import *
from sys import *
```

```
import os.path
import displayGroupMdbToolset as dgm
import displayGroupOdbToolset as dgo
import numpy as np
```

INPUT PARAMETERS

```
#The geometric considerations refer to a frame of reference centered
at the center of the sample, with the X-axis along the width and
pointing to the right, the Z-axis along the thickness and going out
the sheet and the Y-axis along the height and pointing upwards.

#The input parameters must be set with consistent units of measurement
(MPa, mm, N,...)

#Width
W=000

#Fillet radius
RAD=000

#Thickness
Tck=000

#Rounding angle of the fillet [degrees]
alphaG=000

#Young's modulus
E=000

#Poisson's ratio
ni=000

#Side dimension of the material cube to be analyzed with XRD
XRD=000

#Distance between two consecutive quantities evaluations with XRD
dist=000

#Experiment duration
dT=000
```

```
#PEEQ limit
PEEQ_lim=000

#Final Y-displacement
uY=000

#Number of elements along the semi-width at the center of the sample
(at y=0)
nE_w=000

#Number of elements along the semi-thickness at the center of the
sample (at y=0)
nE_t=000

#Number of elements along the semi-width at the top of the sample (at
y=H/2)
nE_wSh=000

#Number of time intervals, each of which the outputs quantities must
be evaluated at
nInt=000

#Number of processors
nProc=000

#String of the 'GeometryOptimization' folder path
mainPath=000
```

INPUT PARAMETERS AS FLOATS

```
#Makes all the input parameters floats in order to be able to use them
in division without unwanted roundings
W=float(W)

RAD=float(RAD)

Tck=float(Tck)

alphaG=float(alphaG)
```

```
E=float(E)

ni=float(ni)

XRD=float(XRD)

dist=float(dist)

dT=float(dT)

PEEQ_lim=float(PEEQ_lim)

uY=float(uY)

nE_w=float(nE_w)

nE_t=float(nE_t)

nE_wSh=float(nE_wSh)

nInt=float(nInt)
```

INITIALIZATION

```
#Sets the option 'COORDINATE' for the replay file (.rpy)
session.journalOptions.setValues(replayGeometry=COORDINATE,
recoverGeometry=COORDINATE)

#Sets the 'GeometryOptimization' folder as the work directory
os.chdir(mainPath)

#Imports the plastic data 'plData' from the text file 'PlasticData.txt'
plData=np.array([0, 0])

fid=open('PlasticData.txt', 'r')

for line in fid:

    sigmaPl=line.split('\t')[0]
    epsPl=line.split('\t')[1]

    plData_line=np.array([float(sigmaPl), float(epsPl)])

    plData=np.vstack((plData, plData_line))

fid.close()

plData=plData[1:len(plData)]

#Defines the path name where all the folders of the Abaqus analyses
will be saved
abaqusDataPath=mainPath + '\\AbaqusData'

#Defines the folder specification for the current work
specFolder='W'+str(W)+'_R'+str(RAD)

#Defines the job specification for the current work
specJob=specFolder.replace('.', ',')

#Defines the analysis name
WorkName='PlaneStrain_' + specFolder

#Checks if any folders with the same actual 'WorkName' exist, in order
not to try to create duplicates
listDir=os.listdir(abaqusDataPath)

count=0

for s in range(len(listDir)):

    listAct=listDir[s]

    listAct.count(specFolder)

    count=count + listAct.count(specFolder)

if count==0:

    WorkName=WorkName
```

```
else:

    WorkName=WorkName + '_(' + str(count) + ')'

#Creates the work directory
folderPath= abaqusDataPath +'\\' + WorkName

os.mkdir(folderPath)

#Changes the work directory
os.chdir(folderPath)
```

EVALUATION OF OTHER PARAMETERS

```
#Semi-width
w=W/2

#Fillet radius
R=RAD

#Semi-thickness
t=Tck/2

#Semi-side dimension of the material cube to be analyzed with XRD
xrd= XRD/2

#Rounding angle of the fillet [rad]
alpha = alphaG/180*pi

#Semi-width of the shoulder (at y=H/2)
w_sh=w+R*(1-cos(alpha))

#Semi-height  at the shoulder
h_sh=R*sin(alpha)

#Size of the smallest element
s_min=t/nE_t

#Size of the biggest element
if w_sh/nE_wSh > s_min:
    s_max=w_sh/nE_wSh

else:

    s_max=s_min

#Size of the smallest element at the fillet shoulder
s_fil=s_max/3

#Semi-height at the inferior partition to be created
#    -because the whole cell will be partitioned to make the mesh
transition possible
h_inf=xrd

#Semi-height at the superior partition to be created
#    -because the whole cell will be partitioned to make the mesh
transition possible
h_sup=h_sh-s_max

#Fillet angle at the inferior partition
alpha_inf=asin(h_inf/R)

#Fillet angle at the superior partition
alpha_sup=asin(h_sup/R)

#Semi-width at the inferior partition to be created
w_inf=w+R*(1-cos(alpha_inf))

#Semi-width at the superior partition to be created
w_sup=w+R*(1-cos(alpha_sup))

#Number of time steps when to get the quantities of interest
nSteps=nInt+1
```

PART

```
#Creates the sheet
mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
sheetSize=200.0)
```

```
#Creates the lines
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0.0, 0.0),
point2=(
    w, 0.0))

mdb.models['Model-1'].sketches['__profile__'].Line(point1=(w_sh,
    h_sh), point2=(0, h_sh))

mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0,  h_sh),
point2=(0.0, 0.0))

#Creates the fillet
mdb.models['Model-
1'].sketches['__profile__'].ArcByCenterEnds(center=(w+R, 0.0),
    point1=(w, 0), point2=(w_sh, h_sh), direction=CLOCKWISE)

#Creates the part 'Gage region'
mdb.models['Model-1'].Part(dimensionality=THREE_D,         name='Gage
region', type=
    DEFORMABLE_BODY)

#Extrudes the part
mdb.models['Model-1'].parts['Gage  region'].BaseSolidExtrude(depth=t,
sketch=
    mdb.models['Model-1'].sketches['__profile__'])

#Creates a point lying on each frontal face (i.e. perpendicular to Z-
axis, at z=T/2)
infFace_Z = (w/3, h_inf/3, t)

medFace_Z = (w/3, h_inf+(h_sup-h_inf)/3, t)

supFaceLeft_Z = (w/3, h_sup+(h_sh-h_sup)/3, t)

supFaceRight_Z = (w_sup+(w_sh-w_sup)/100, h_sh-(h_sh-h_sup)/100, t)

#Creates a point lying on each lateral face (i.e. perpendicular to X-
axis, at x=0)
infFace_X = (0, h_inf/3, t/3)
```

```
medFace_X = (0, h_inf+(h_sup-h_inf)/3, t/3)

supFace_X = (0, h_sup+(h_sh-h_sup)/3, t/3)

#Creates a point lying on the inferior face (i.e. perpendicular to Y-
axis, at y=0)
infFace_Y = (w/3, 0, t/3)

#Creates a point lying on each superior face (i.e. perpendicular to Y-
axis, at y=H/2)
supFaceLeft_Y = (w/3, h_sh, t/3)

supFaceRight_Y = (w_sup+(w_sh-w_sup)/3, h_sh, t/3)

#Cell inferior partition
mdb.models['Model-1'].ConstrainedSketch(gridSpacing=0.08,
name='__profile__',
    sheetSize=3.3, transform=
    mdb.models['Model-1'].parts['Gage region'].MakeSketchTransform(
    sketchPlane=mdb.models['Model-1'].parts['Gage
region'].faces.findAt(infFace_Z, ),
    sketchPlaneSide=SIDE1,            sketchUpEdge=mdb.models['Model-
1'].parts['Gage region'].
    edges.findAt((0, h_sh/3, t), ), sketchOrientation=LEFT, origin=(0,
0, t)))

mdb.models['Model-1'].parts['Gage
region'].projectReferencesOntoSketch(filter=
    COPLANAR_EDGES,                              sketch=mdb.models['Model-
1'].sketches['__profile__'])

mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(0,
0),
    point2=(w_sh, h_inf))

mdb.models['Model-1'].parts['Gage
region'].PartitionFaceBySketch(faces=
    mdb.models['Model-1'].parts['Gage
region'].faces.findAt((infFace_Z, )),
    sketch=mdb.models['Model-1'].sketches['__profile__'],
    sketchUpEdge=mdb.models['Model-1'].parts['Gage
region'].edges.findAt((0, h_sh/3, t), ))

del mdb.models['Model-1'].sketches['__profile__']

mdb.models['Model-1'].parts['Gage
region'].PartitionCellByExtrudeEdge(cells=
    mdb.models['Model-1'].parts['Gage
region'].cells.findAt((infFace_Z,
    )),                      edges=(mdb.models['Model-1'].parts['Gage
region'].edges.findAt((w/3,
    h_inf, t), ), ), line=
    mdb.models['Model-1'].parts['Gage   region'].edges.findAt((0,   0,
t/3), ),
    sense=REVERSE)

#Cell superior left partition
mdb.models['Model-1'].ConstrainedSketch(gridSpacing=0.08,
name='__profile__',
    sheetSize=3.3,        transform=mdb.models['Model-1'].parts['Gage
region'].
    MakeSketchTransform(sketchPlane=mdb.models['Model-1'].parts['Gage
region']
    .faces.findAt(medFace_Z, ), sketchPlaneSide=SIDE1,
    sketchUpEdge=mdb.models['Model-1'].parts['Gage
region'].edges.findAt((0,
    h_sh/3, t), ), sketchOrientation=LEFT, origin=(0, 0, t)))

mdb.models['Model-1'].parts['Gage
region'].projectReferencesOntoSketch(filter=
    COPLANAR_EDGES,                        sketch=mdb.models['Model-
1'].sketches['__profile__'])

mdb.models['Model-1'].sketches['__profile__'].rectangle(point1=(0,
0),
    point2=(w_sh, h_sup))

mdb.models['Model-1'].parts['Gage
region'].PartitionFaceBySketch(faces=
    mdb.models['Model-1'].parts['Gage
region'].faces.findAt((medFace_Z, )),
    sketch=mdb.models['Model-1'].sketches['__profile__'],
    sketchUpEdge=mdb.models['Model-1'].parts['Gage
region'].edges.findAt((0, h_sh/3, t), ))

del mdb.models['Model-1'].sketches['__profile__']

mdb.models['Model-1'].parts['Gage
region'].PartitionCellByExtrudeEdge(cells=
    mdb.models['Model-1'].parts['Gage
region'].cells.findAt((medFace_Z,
    )),                         edges=(mdb.models['Model-1'].parts['Gage
region'].edges.findAt((w/3,
    h_sup, t), ), ), line=mdb.models['Model-1'].parts['Gage region'].
    edges.findAt((0, 0, t/3), ), sense=REVERSE)

#Cell superior right partition
mdb.models['Model-1'].ConstrainedSketch(gridSpacing=0.08,
name='__profile__',
    sheetSize=3.3,        transform=mdb.models['Model-1'].parts['Gage
region'].
    MakeSketchTransform(sketchPlane=mdb.models['Model-1'].parts['Gage
region']
    .faces.findAt(supFaceLeft_Z, ), sketchPlaneSide=SIDE1,
    sketchUpEdge=mdb.models['Model-1'].parts['Gage
region'].edges.findAt((0,
    h_sh/3, t), ), sketchOrientation=LEFT, origin=(0, 0, t)))

mdb.models['Model-1'].parts['Gage
region'].projectReferencesOntoSketch(filter=
    COPLANAR_EDGES,                          sketch=mdb.models['Model-
1'].sketches['__profile__'])

mdb.models['Model-
1'].sketches['__profile__'].rectangle(point1=(w_sup, h_sup),
    point2=(w_sh, h_sh))

mdb.models['Model-1'].parts['Gage
region'].PartitionFaceBySketch(faces=
    mdb.models['Model-1'].parts['Gage
region'].faces.findAt((supFaceLeft_Z, )),
    sketch=mdb.models['Model-1'].sketches['__profile__'],
    sketchUpEdge=mdb.models['Model-1'].parts['Gage
region'].edges.findAt((0, h_sh/3, t), ))

del mdb.models['Model-1'].sketches['__profile__']

mdb.models['Model-1'].parts['Gage
region'].PartitionCellByExtrudeEdge(cells=
    mdb.models['Model-1'].parts['Gage
region'].cells.findAt((supFaceLeft_Z,
    )),                         edges=(mdb.models['Model-1'].parts['Gage
region'].edges.findAt
    ((w_sup, h_sup+(h_sh-h_sup)/3, t), ), ), line=mdb.models['Model-
1'].parts['Gage    region'].edges.findAt((w_sh,    h_sh,    t/3),    ),
sense=REVERSE)
```

PROPERTIES

```
#Defines the material
mdb.models['Model-1'].Material(name='Steel')

#Elastic properties
mdb.models['Model-1'].materials['Steel'].Elastic(table=((E, ni), ))

#Plastic properties
mdb.models['Model-1'].materials['Steel'].Plastic(table=plData)

#Creates the section
mdb.models['Model-1'].HomogeneousSolidSection(material='Steel', name=
    'Section-1', thickness=None)

#Assigns the section to each cell separately
mdb.models['Model-1'].parts['Gage
region'].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
    cells=mdb.models['Model-1'].parts['Gage     region'].cells[0:4]),
sectionName=
    'Section-1', thicknessAssignment=FROM_SECTION)
```

ASSEMBLY

```
   #Creates the assembly
```

```
mdb.models['Model-1'].rootAssembly.Instance(dependent=ON,  name='Gage
region-1',
    part=mdb.models['Model-1'].parts['Gage region'])
```

STEP

```
#Defines the static step
mdb.models['Model-1'].StaticStep(description='Imposes vertical
displacement uY',
    initialInc=dT, maxInc=0.001, minInc=1e-08, name='Pulling',
previous='Initial',
    timePeriod=dT, nlgeom=ON)

#Sets the field outputs to be exported
mdb.models['Model-1'].FieldOutputRequest(createStepName='Pulling',
    exteriorOnly=OFF, name='F-Output-1', region=MODEL,
timeInterval=dT/nInt,
    variables=('S', 'PEEQ', 'U', 'NFORC'))

#Sets the history outputs to be exported
mdb.models['Model-1'].HistoryOutputRequest(createStepName='Pulling',
    timeInterval=dT/nInt, name='H-Output-1', variables=('ETOTAL', ))
```

LOAD

```
#Sets the X-symmetry boundary condition
mdb.models['Model-1'].rootAssembly.Set(faces=
    mdb.models['Model-1'].rootAssembly.instances['Gage region-1'].
    faces.findAt((infFace_X,  ),  (medFace_X,  ),  (supFace_X,  ),  ),
name='Faces_Xsym')

mdb.models['Model-1'].XsymmBC(createStepName='Initial',
localCsys=None, name=
    'X-symmetry              ',              region=mdb.models['Model-
1'].rootAssembly.sets['Faces_Xsym'])

#Sets the Y-symmetry boundary condition
mdb.models['Model-1'].rootAssembly.Set(faces=
    mdb.models['Model-1'].rootAssembly.instances['Gage region-1'].
    faces.findAt((infFace_Y, ), ), name='Face_Ysym')
```

```
mdb.models['Model-1'].YsymmBC(createStepName='Initial',
localCsys=None, name=
    'Y-symmetry              ',              region=mdb.models['Model-
1'].rootAssembly.sets['Face_Ysym'])

#Sets the Z-symmetry boundary condition
mdb.models['Model-1'].rootAssembly.Set(faces=mdb.models['Model-1'].
    rootAssembly.instances['Gage  region-1'].faces.findAt((infFace_Z,
),
    (medFace_Z,   ),   (supFaceLeft_Z,   ),   (supFaceRight_Z,   )),
name='Faces_Zsym')

mdb.models['Model-1'].ZsymmBC(createStepName='Initial',
localCsys=None, name=
    'Z-symmetry              ',              region=mdb.models['Model-
1'].rootAssembly.sets['Faces_Zsym'])

#Sets the final displacement
mdb.models['Model-1'].rootAssembly.Set(faces=
    mdb.models['Model-1'].rootAssembly.instances['Gage region-1'].
    faces.findAt((supFaceLeft_Y,    ),    (supFaceRight_Y,    )    ),
name='Face_Displ')

mdb.models['Model-1'].DisplacementBC(amplitude=UNSET,
createStepName='Initial',
    distributionType=UNIFORM,      fieldName='',      localCsys=None,
name='Displacement',
    region=mdb.models['Model-1'].rootAssembly.sets['Face_Displ'],
u1=SET, u2=SET,
    u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=UNSET)

mdb.models['Model-
1'].boundaryConditions['Displacement'].setValuesInStep(stepName=
    'Pulling', u1=0, u2=uY)
```

MESH

```
#Creates a geometric entity corresponding to each cell
infCell              =              mdb.models['Model-1'].parts['Gage
region'].cells.findAt((infFace_Z, ), )
```

```
medCell              =              mdb.models['Model-1'].parts['Gage
region'].cells.findAt((medFace_Z, ), )

supCellLeft            =            mdb.models['Model-1'].parts['Gage
region'].cells.findAt((supFaceLeft_Z, ), )

supCellRight            =           mdb.models['Model-1'].parts['Gage
region'].cells.findAt((supFaceRight_Z, ), )

#Sets sweep mesh for the medial cell
mdb.models['Model-1'].parts['Gage
region'].setMeshControls(algorithm=ADVANCING_FRONT,
    regions=medCell, technique=SWEEP, elemShape=HEX)

#Sets sweep mesh for the superior right cell
mdb.models['Model-1'].parts['Gage
region'].setMeshControls(algorithm=ADVANCING_FRONT,
    regions=supCellRight, technique=SWEEP, elemShape=HEX_DOMINATED)

#Sets the elements size on the inferior cell lines
mdb.models['Model-1'].parts['Gage
region'].seedEdgeBySize(constraint=FINER,
    deviationFactor=0.1, edges=
    mdb.models['Model-1'].parts['Gage  region'].edges.findAt(((w/3,  0
, t), ),
    ((0, h_inf/3 , t), ), ((w, 0, t/3), ),
    ((w+R*(1-cos(alpha_inf/3)),   R*sin(alpha_inf/3),   t),   ),   ),
size=s_min)

#Sets the elements size on the medial cell lines
mdb.models['Model-1'].parts['Gage
region'].seedEdgeByBias(biasMethod=SINGLE,
    constraint=FINER, end1Edges=    mdb.models['Model-1'].parts['Gage
region'].
    edges.findAt(((0, h_inf+(h_sup-h_inf)/3, t), )), end2Edges=
    mdb.models['Model-1'].parts['Gage region'].edges.findAt
    (((w+R*(1-cos(alpha_inf+ (alpha_sup-alpha_inf)/3)),
    R*sin(alpha_inf+(alpha_sup-alpha_inf)/3),  t),  )), maxSize=s_max,
minSize=s_min)

#Sets the elements size on the superior left and right cell lines
```

```
mdb.models['Model-1'].parts['Gage
region'].seedEdgeBySize(constraint=FIXED,
    deviationFactor=0.1,      edges=mdb.models['Model-1'].parts['Gage
region'].
    edges.findAt(((w/3, h_sup, t), ), ((0, h_sup+(h_sh-h_sup)/3, t),
),
    ((w_sup, h_sup+(h_sh-h_sup)/3, t), ), ), size=s_max)

mdb.models['Model-1'].parts['Gage
region'].seedEdgeBySize(constraint=FINER,
    deviationFactor=0.1, edges=
    mdb.models['Model-1'].parts['Gage region'].edges.findAt
    (((0, h_sh, t/3), ), ((w_sh, h_sh, t/3 ), ), ), size=s_min)

mdb.models['Model-1'].parts['Gage
region'].seedEdgeByBias(biasMethod=SINGLE,
    constraint=FINER,     end1Edges=mdb.models['Model-1'].parts['Gage
region'].edges.findAt
    (((w+R*(1-cos(alpha_sup+ (alpha-alpha_sup)/3)),
    R*sin(alpha_sup+(alpha-alpha_sup)/3), t), )), end2Edges=
    mdb.models['Model-1'].parts['Gage region'].edges.findAt
    (((w_sup+(w_sh-w_sup)/3,   h_sh,   t   ),   )),   maxSize=s_max,
minSize=s_fil)

#Generates the mesh
mdb.models['Model-1'].parts['Gage region'].generateMesh(regions=
    infCell)

mdb.models['Model-1'].parts['Gage region'].generateMesh(regions=
    medCell)

mdb.models['Model-1'].parts['Gage region'].generateMesh(regions=
    supCellLeft)

mdb.models['Model-1'].parts['Gage region'].generateMesh(regions=
    supCellRight)
```

INITIAL JOB

```
#Defines the job name
JobName='PlaneStrain_'+specJob
```

```
#Creates the job
mdb.Job(name=JobName, model='Model-1', description='', type=ANALYSIS,
    atTime=None, waitMinutes=0, waitHours=0, queue=None, memory=90,
    memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,
    explicitPrecision=SINGLE,            nodalOutputPrecision=SINGLE,
echoPrint=OFF,
    modelPrint=OFF,        contactPrint=OFF,        historyPrint=OFF,
userSubroutine='',
    scratch='',    resultsFormat=ODB,    multiprocessingMode=DEFAULT,
numCpus=int(nProc),
    numDomains=int(nProc), numGPUs=0)

#Saves the work in a .cae file
mdb.saveAs(pathName=folderPath+'\\'+JobName)

#Submits the job
mdb.jobs[JobName].submit(consistencyChecking=OFF)

#Waits for the job to be completed
mdb.jobs[JobName].waitForCompletion()
```

THRESHOLD STEP DETECTION

```
#Opens and displays the results file (.odb)
session.Viewport(name='Viewport: 1', origin=(0.0, 0.0), width=100,
    height=150)

session.viewports['Viewport: 1'].makeCurrent()

session.viewports['Viewport: 1'].maximize()

session.viewports['Viewport:
1'].partDisplay.geometryOptions.setValues(
    referenceRepresentation=ON)

session.viewports['Viewport:
1'].viewportAnnotationOptions.setValues(title=OFF)

odb=session.openOdb(JobName+'.odb')
```

```
session.viewports['Viewport: 1'].setValues(displayedObject=odb)

#Evaluates the number of evenly spaced points along one single
direction (X, Y or Z) where to measure the quantities of interest
nP_XYZ=int(xrd/dist+1)

#Evaluates the number of evenly spaced points in the XRD volume to be
analyzed
nP=nP_XYZ**3

#Initializes the 'XRD_path_ar' matrix (nP x 3), that will contain the
coordinates of the points in the volume to be analyzed with XRD
XRD_path_ar=np.array([], dtype=np.int64).reshape(0,3)

#Fills out the 'XRD_path_ar' matrix
for itY in range(nP_XYZ):

    v=np.zeros((nP_XYZ,3))

    v[:,0]=np.linspace(0, xrd, nP_XYZ)

    v[:,1]=itY*dist*np.ones(nP_XYZ)

    for itZ in range(nP_XYZ):

        v[:,2]=(t-itZ*dist)*np.ones(nP_XYZ)

        XRD_path_ar=np.vstack((XRD_path_ar,v))

#Converts the 'XRD_path_ar' matrix to a list
XRD_path=XRD_path_ar.tolist()

#Creates the path for the points of the volume to be analyzed with XRD
session.Path(name='XRD_path', type=POINT_LIST, expression=XRD_path)

#Creates the path representing a single point on the superior face, in
order to extract later the displacement u2 along the Y-axis
session.Path(name='Displacement_path',               type=POINT_LIST,
expression=[[0, h_sh, t]])

#Initializes the 'K' index, necessary to let the next loop cycle run
```

```
K=0

#Initializes the 'peeq_av_XRD' scalar, representing the mean PEEQ in
the XRD-volume and necessary to let the next loop cycle run
peeq_av_XRD=0

#Runs  the  'while'  loop  cycle,  able  to  find  the  time  instant  when
'peeq_av_XRD' =~ 'PEEQ_lim'
while peeq_av_XRD <= PEEQ_lim:

    #Exports the PEEQ for the 'XRD_path' at the Kth time instant
    session.viewports['Viewport:      1'].odbDisplay.setFrame(step=0,
frame=K)

    session.viewports['Viewport: 1'].odbDisplay.setPrimaryVariable(
        variableLabel='PEEQ', outputPosition=INTEGRATION_POINT)

    session.XYDataFromPath(name='XRD_'+str(K),
path=session.paths['XRD_path'],
        includeIntersections=False,           projectOntoMesh=False,
pathStyle=PATH_POINTS,
        numIntervals=1,    projectionTolerance=0,    shape=UNDEFORMED,
labelType=TRUE_DISTANCE,
        variable=(('PEEQ', INTEGRATION_POINT), ))

    session.xyDataObjects.changeKey(fromName='XRD_'+str(K),
toName='XRD_'+str(K)+'-PEEQ')

    del session.xyDataObjects['XRD_'+str(K)]

    #Converts the PEEQ data object to the 'peeq' array (nP x 2)
    peeq = np.array(session.xyDataObjects['XRD_'+str(K)+'-PEEQ'])

    #Deletes the PEEQ data object
    del session.xyDataObjects['XRD_'+str(K)+'-PEEQ']

    #Removes the possible duplicated rows from the 'peeq' array
    if len(np.unique(peeq[:,0])) != len(peeq[:,0]):

        while len(np.unique(peeq[:,0])) < len(peeq[:,0]):
                aux=np.unique(peeq[:,0])-peeq[0:nP,0]

                aux=[ '%.5f' % elem for elem in aux.tolist()]

                aux=aux[::-1]

                ind=len(aux)- 1 - aux.index('0.00000')

                peeq[ind:nP]=peeq[ind+1:nP+1]

                peeq=np.delete(peeq, (nP-1), axis=0)

    #Removes from the 'peeq' array the column representing the true
distance along the 'XRD_path' , making 'peeq' a column array (nP x 1)
    peeq=peeq[:,1]

    #Evaluates the scalar 'peeq_av_XRD', mean value of PEEQ in the
XRD-volume at the Kth time instant
    peeq_av_XRD=np.average(peeq)

    #Exports u2 for the 'Displacement_path' at the Kth time instant
    session.XYDataFromPath(name='Displacement_'+str(K),
path=session.paths['Displacement_path'],
        includeIntersections=False,          projectOntoMesh=False,
pathStyle=PATH_POINTS,
        numIntervals=1,    projectionTolerance=0,    shape=UNDEFORMED,
labelType=TRUE_DISTANCE,
        variable=(('U', NODAL, ((COMPONENT, 'U2'), )), ))

    session.xyDataObjects.changeKey(fromName='Displacement_'+str(K),
toName='Displacement_'+str(K)+'-U2')

    del session.xyDataObjects['Displacement_'+str(K)]

    #Converts the U2 data object to the 'u2' array (1 x 2)
    u2   =   np.array(session.xyDataObjects['Displacement_'+str(K)+'-
U2'])

    #Deletes the U2 data object
    del session.xyDataObjects['Displacement_'+str(K)+'-U2']
    #Removes from the 'u2' array the column representing the true
distance along the 'Displacement_path' , making 'u2' a scalar
    u2=u2[:,1]

    #Updates the 'K' index
    K=K+1

#Evaluates the threshold displacement, at which 'peeq_min_XRD' =~
'PEEQ_lim'
uY_thsd= u2[0]
```

UPDATED JOB

```
#Sets the updated final Y-displacement, so that when it is reached it
is 'peeq_av_XRD' =~ 'PEEQ_lim'
mdb.models['Model-
1'].boundaryConditions['Displacement'].setValuesInStep(stepName=
    'Pulling', u2=uY_thsd)

#Defines the job name
JobName='PlaneStrain_'+specJob

#Creates the job
mdb.Job(name=JobName, model='Model-1', description='', type=ANALYSIS,
    atTime=None, waitMinutes=0, waitHours=0, queue=None, memory=90,
    memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,
    explicitPrecision=SINGLE,             nodalOutputPrecision=SINGLE,
echoPrint=OFF,
    modelPrint=OFF,        contactPrint=OFF,        historyPrint=OFF,
userSubroutine='',
    scratch='',    resultsFormat=ODB,    multiprocessingMode=DEFAULT,
numCpus=int(nProc),
    numDomains=int(nProc), numGPUs=0)

#Saves the work in a .cae file
mdb.saveAs(pathName=folderPath+'\\'+JobName)

#Submits the job
mdb.jobs[JobName].submit(consistencyChecking=OFF)

#Waits for the job to be completed
```

```
mdb.jobs[JobName].waitForCompletion()
```

RESULTS

```
#Opens and displays the results file (.odb)
session.Viewport(name='Viewport: 1', origin=(0.0, 0.0), width=100,
    height=150)

session.viewports['Viewport: 1'].makeCurrent()

session.viewports['Viewport: 1'].maximize()

session.viewports['Viewport:
1'].partDisplay.geometryOptions.setValues(
    referenceRepresentation=ON)

session.viewports['Viewport:
1'].viewportAnnotationOptions.setValues(title=OFF)

odb=session.openOdb(JobName+'.odb')

session.viewports['Viewport: 1'].setValues(displayedObject=odb)

#Creates the 'TIME' array (nSteps x 1), containing all the time
instants (in [ms]) when the quantities of interest have been evaluated
TIME=1000*np.linspace(0, nInt*(dT/nInt), nSteps)

#Creates the empty matrix 'PEEQ' (nSteps x nP), to be filled out with
the plastic equivalent strain data of the XRD-volume points over time
PEEQ=np.zeros((nSteps, nP))

#Creates the empty matrix 'TRIAX' (nSteps x nP), to be filled out with
the stress triaxiality data of the XRD-volume points over time
TRIAX=np.zeros((nSteps, nP))

#Creates the empty array 'PEEQ_max' (nSteps x 1), to be filled out
with the minimum PEEQ evaluated over time
PEEQ_max=np.zeros(nSteps)

#Creates the empty array 'PEEQ_min_XRD' (nSteps x 1), to be filled out
with the minimum PEEQ evaluated in the XRD-volume over time
```

```
PEEQ_min_XRD=np.zeros(nSteps)

#Creates the empty array 'PEEQ_av_XRD' (nSteps x 1), to be filled out
with the mean value of PEEQ evaluated in the XRD-volume over time
PEEQ_av_XRD=np.zeros(nSteps)

#Creates the empty array 'PEEQ_max_XRD' (nSteps x 1), to be filled out
with the maximum PEEQ evaluated in the XRD-volume over time
PEEQ_max_XRD=np.zeros(nSteps)

#Creates the empty array 'TRIAX_av' (nSteps x 1), to be filled out
with the mean values of stress triaxiality evaluated in the XRD-volume
over time
TRIAX_av=np.zeros(nSteps)

#Creates the empty array 'U2' (nSteps x 1), to be filled out with the
Y-displacement of the superior face over time
U2=np.zeros(nSteps)

#Creates the empty array 'FORCE_Y' (nSteps x 1), to be filled out with
the Y-force acting on the whole superior face over time
FORCE_Y=np.zeros(nSteps)

#Initializes the 'for' loop cycle, able to get the data of interest at
each time instant
for J in range(int(nSteps)):

    #Exports the data objects for the 'XRD_path'
    session.viewports['Viewport:      1'].odbDisplay.setFrame(step=0,
frame=J)

    session.viewports['Viewport: 1'].odbDisplay.setPrimaryVariable(
        variableLabel='S',          outputPosition=INTEGRATION_POINT,
refinement=(INVARIANT,
        'Mises'))

    session.viewports['Viewport: 1'].odbDisplay.setPrimaryVariable(
        variableLabel='PEEQ', outputPosition=INTEGRATION_POINT)

    session.XYDataFromPath(name='XRD_'+str(J),
path=session.paths['XRD_path'],
        includeIntersections=False,            projectOntoMesh=False,
pathStyle=PATH_POINTS,
        numIntervals=1,    projectionTolerance=0,    shape=UNDEFORMED,
labelType=TRUE_DISTANCE,
        variable=(('S',  INTEGRATION_POINT,  ((INVARIANT,  'Mises'  ),
(INVARIANT, 'Pressure' ), ), ),
        ('PEEQ', INTEGRATION_POINT), ))

    session.xyDataObjects.changeKey(fromName='XRD_'+str(J),
toName='XRD_'+str(J)+'-PEEQ')

    del session.xyDataObjects['XRD_'+str(J)]

    #Exports u2 for the 'Displacement_path'
    session.XYDataFromPath(name='Displacement_'+str(J),
path=session.paths['Displacement_path'],
        includeIntersections=False,            projectOntoMesh=False,
pathStyle=PATH_POINTS,
        numIntervals=1,    projectionTolerance=0,    shape=UNDEFORMED,
labelType=TRUE_DISTANCE,
        variable=(('U', NODAL, ((COMPONENT, 'U2'), )), ))

    session.xyDataObjects.changeKey(fromName='Displacement_'+str(J),
toName='Displacement_'+str(J)+'-U2')

    del session.xyDataObjects['Displacement_'+str(K)]

    #Converts the data objects to arrays
    peeq = np.array(session.xyDataObjects['XRD_'+str(J)+'-PEEQ'])

    mises = np.array(session.xyDataObjects['XRD_'+str(J)+'-Mises'])

    pressure    =    np.array(session.xyDataObjects['XRD_'+str(J)+'-
Pressure'])

    u2   =   np.array(session.xyDataObjects['Displacement_'+str(J)+'-
U2'])

    #Deletes the data objects
    del session.xyDataObjects['XRD_'+str(J)+'-PEEQ']
    del session.xyDataObjects['XRD_'+str(J)+'-Mises']

    del session.xyDataObjects['XRD_'+str(J)+'-Pressure']

    del session.xyDataObjects['Displacement_'+str(J)+'-U2']

    #Removes the possible duplicated rows from the 'peeq' array
    if len(np.unique(peeq[:,0])) != len(peeq[:,0]):

        while len(np.unique(peeq[:,0])) < len(peeq[:,0]):

            aux=np.unique(peeq[:,0])-peeq[0:nP,0]

            aux=[ '%.5f' % elem for elem in aux.tolist()]

            aux=aux[::-1]

            ind=len(aux)- 1 - aux.index('0.00000')

            peeq[ind:nP]=peeq[ind+1:nP+1]

            peeq=np.delete(peeq, (nP-1), axis=0)

    #Removes the possible duplicated rows from the 'mises' array
    if len(np.unique(mises[:,0])) != len(mises[:,0]):

        while len(np.unique(mises[:,0])) < len(mises[:,0]):

            aux=np.unique(mises[:,0])-mises[0:nP,0]

            aux=[ '%.5f' % elem for elem in aux.tolist()]

            aux=aux[::-1]

            ind=len(aux)- 1 - aux.index('0.00000')

            mises[ind:nP]=mises[ind+1:nP+1]

            mises=np.delete(mises, (nP-1), axis=0)

    #Removes the possible duplicated rows from the 'pressure' array
    if len(np.unique(pressure[:,0])) != len(pressure[:,0]):

        while len(np.unique(pressure[:,0])) < len(pressure[:,0]):

            aux=np.unique(pressure[:,0])-pressure[0:nP,0]

            aux=[ '%.5f' % elem for elem in aux.tolist()]

            aux=aux[::-1]

            ind=len(aux)- 1 - aux.index('0.00000')

            pressure[ind:nP]=pressure[ind+1:nP+1]

            pressure=np.delete(pressure, (nP-1), axis=0)

    #Removes from the arrays the column representing the true distance
along the 'XRD_path', making them column arrays (nP x 1)
    peeq=peeq[:,1]

    mises=mises[:,1]

    pressure=pressure[:,1]

    #Removes from the 'u2' array the column representing the true
distance along the 'Displacement_path' , making 'u2' a scalar
    u2=u2[:,1]

    #Inserts the 'peeq' array in the Jth row of the 'PEEQ' matrix
    PEEQ[J,:]=peeq

    #Inserts the 'u2' scalar in the Jth row of the 'U2' array
    U2[J]=u2

    #Inserts the 'peeq_max' scalar in the Jth spot of the 'PEEQ_max'
array
    PEEQ_max[J]=peeq_max

    #Evaluates the 'triax' array (1 x nP), representing the stress
triaxiality of the points in the XRD-volume at the Jth time instant
    triax=-pressure/(mises + (mises==0))
    #Inserts the 'triax' array in the Jth row of the 'TRIAX' matrix
    TRIAX[J,:]=triax

    #Evaluates the scalar 'triax_av', mean value of stress triaxiality
in the XRD volume at the Jth time instant
    triax_av=np.average(triax)

    #Inserts the 'triax_av' scalar in the Jth spot of the 'TRIAX_av'
array
    TRIAX_av[J]=triax_av

    #Creates the free body cut 'FreeBody', in order to get the value
of the Y-force acting on the superior face
    eLeaf = dgo.LeafFromElementSets(elementSets=('FACE_DISPL', ))

    nLeaf = dgo.LeafFromNodeSets(nodeSets=('FACE_DISPL', ))

    session.FreeBodyFromNodesElements(name='FreeBody',
elements=eLeaf,
        nodes=nLeaf, summationLoc=NODAL_AVERAGE,
        componentResolution=NORMAL_TANGENTIAL)

    session.viewports['Viewport:
1'].odbDisplay.setValues(freeBodyNames=(
        'FreeBody', ), freeBody=ON)

    #Creates the 'FORCE_Y' data object
    odbName=session.viewports[session.currentViewportName].
        odbDisplay.name

    session.odbData[odbName].setValues(activeFrames=(('Pulling',  (J,
)), ))

    session.XYDataFromFreeBody(odb=odb,     force=ON,     moment=OFF,
resultant=OFF,
        comp1=ON, comp2=OFF, comp3=OFF)

    #Converts the 'FORCE_Y' data object to the 'force_Y' scalar
    force_Y=np.array(session.xyDataObjects['FreeBody force component
1'])[:,1]
```

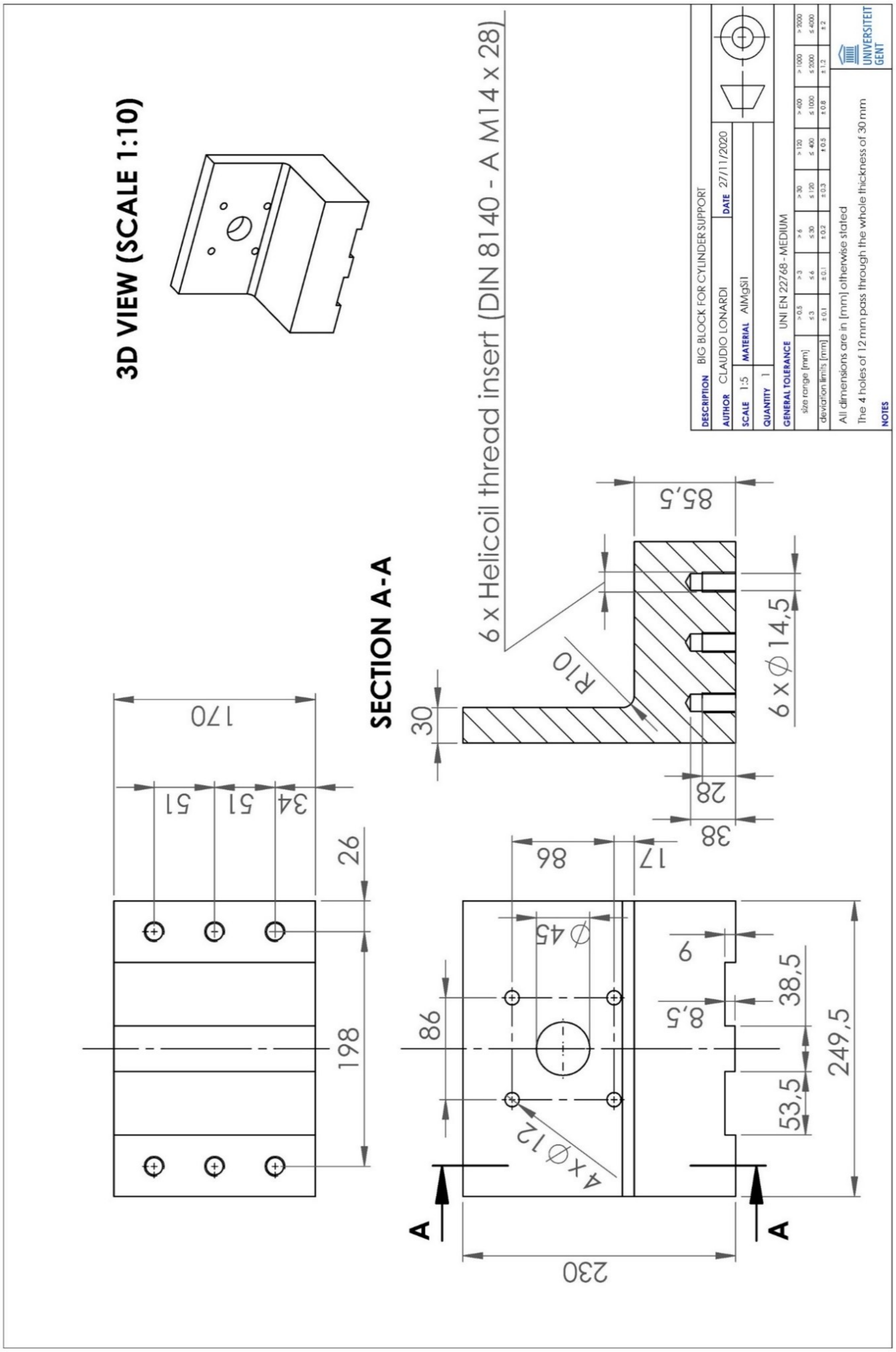
```
    #Deletes the FORCE_Y' data object
    del session.xyDataObjects['FreeBody force component 1']

    #Inserts the 'force_Y' scalar in the Jth spot of the 'FORCE_Y' array
    FORCE_Y[J]=force_Y

    J=J+1

#Turns from '-0' to '0' the first row of 'TRIAX' matrix, for a better visualization
TRIAX[0,:]=-TRIAX[0,:]

#Evaluates the scalar 'TRIAX_avav', mean value of the mean values stored in the 'TRIAX_av' array
TRIAX_avav=np.average(TRIAX_av[1:nSteps])
```

EXPORT TO FILE

```
#Sets the number of space characters between two columns in the .txt files that are going to be exported
Space = '        '

#Exports the 'TIME' array in a .txt file
np.savetxt('TIME.txt', TIME, fmt='%4.4f', delimiter=Space)

#Exports the 'PEEQ' matrix in a .txt file
np.savetxt('PEEQ.txt', PEEQ, fmt='%4.3f', delimiter=Space)

#Exports the 'TRIAX' matrix in a .txt file
np.savetxt('TRIAX.txt', TRIAX, fmt='%4.3f', delimiter=Space)

#Exports the 'PEEQ_min_XRD' array in a .txt file
np.savetxt('PEEQ_min_XRD.txt',       PEEQ_min_XRD,       fmt='%4.5f',
delimiter=Space)

#Exports the 'PEEQ_av_XRD' array in a .txt file
np.savetxt('PEEQ_av_XRD.txt',        PEEQ_av_XRD,        fmt='%4.5f',
delimiter=Space)
```

```
#Exports the 'PEEQ_max_XRD' array in a .txt file
np.savetxt('PEEQ_max_XRD.txt',       PEEQ_max_XRD,       fmt='%4.5f',
delimiter=Space)

#Exports the 'PEEQ_max' array in a .txt file
np.savetxt('PEEQ_max.txt', PEEQ_max, fmt='%4.4f', delimiter=Space)

#Exports the 'TRIAX_av' array in a .txt file
np.savetxt('TRIAX_av.txt', TRIAX_av, fmt='%4.5f', delimiter=Space)

#Exports the 'TRIAX_avav' array in a .txt file
np.savetxt('TRIAX_avav.txt',        np.array(TRIAX_avav).reshape(1,),
fmt='%4.5f', delimiter=Space)

#Exports the 'U2' array in a .txt file
np.savetxt('U2.txt', U2, fmt='%4.5f', delimiter=Space)

#Exports the 'FORCE' matrix in a .txt file
np.savetxt('FORCE_Y.txt', FORCE_Y, fmt='%4.3f', delimiter=Space)

#Changes the work directory
os.chdir(abaqusDataPath)

#Exports the string 'folderPath', the name of the work directory, in
a .txt file
fid = open('folderPath.txt', 'w')

fid.write(folderPath)

fid.close()
```

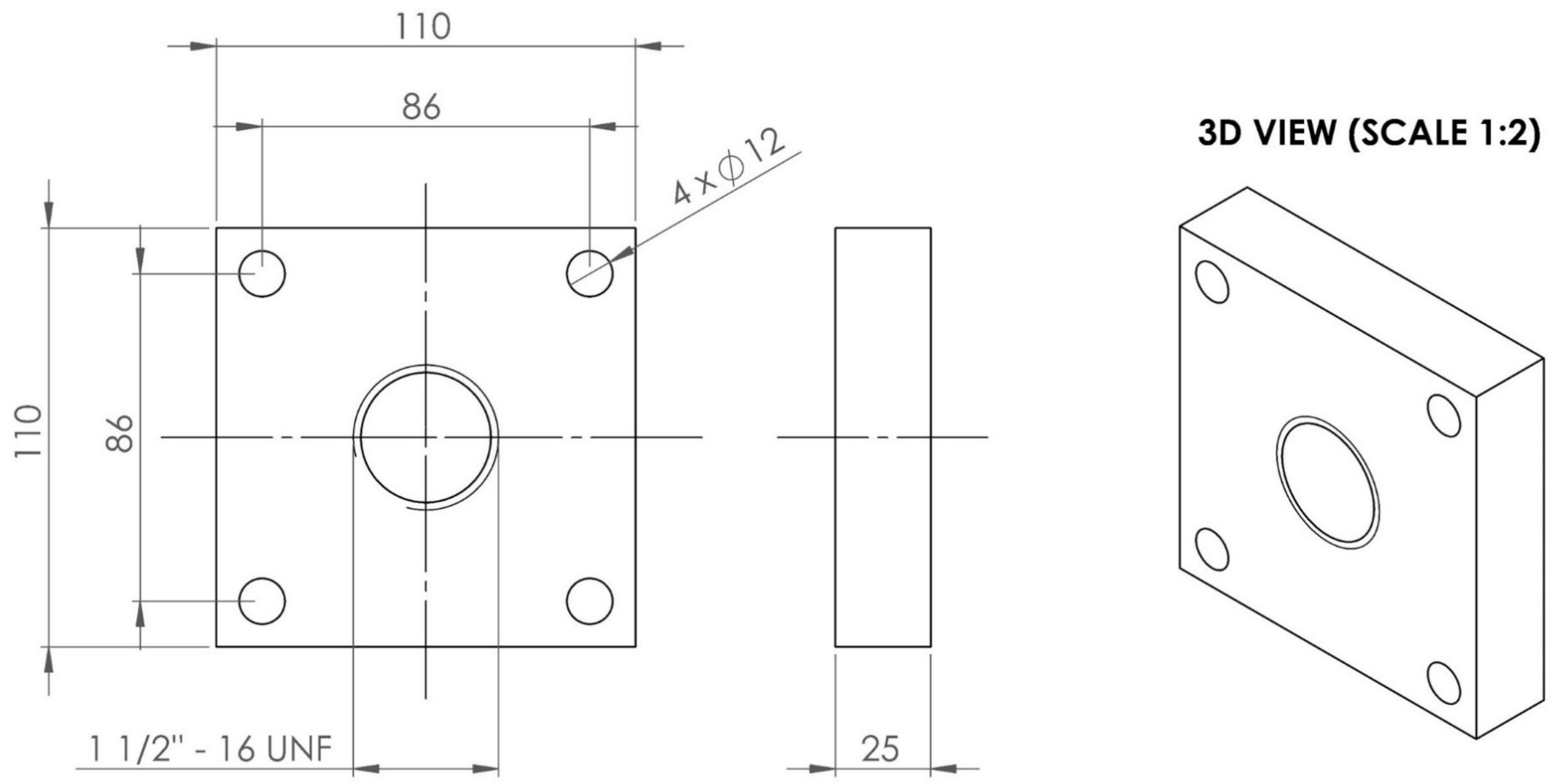
Technical drawings



DESCRIPTION BIG BLOCK FOR CYLINDER SUPPORT								
AUTHOR CLAUDIO LONARDI				**DATE** 27/11/2020				
SCALE 1:5	**MATERIAL** AlMgSi1							
QUANTITY 1								
GENERAL TOLERANCE UNI EN 22768 - MEDIUM								
size range [mm]	> 0.5 ≤ 3	> 3 ≤ 6	> 6 ≤ 30	> 30 ≤ 120	> 120 ≤ 400	> 400 ≤ 1000	> 1000 ≤ 2000	> 2000 ≤ 4000
deviation limits [mm]	± 0.1	± 0.1	± 0.2	± 0.3	± 0.5	± 0.8	± 1.2	± 2

NOTES
All dimensions are in [mm] otherwise stated
The 4 holes of 12 mm pass through the whole thickness of 30 mm

UNIVERSITEIT GENT



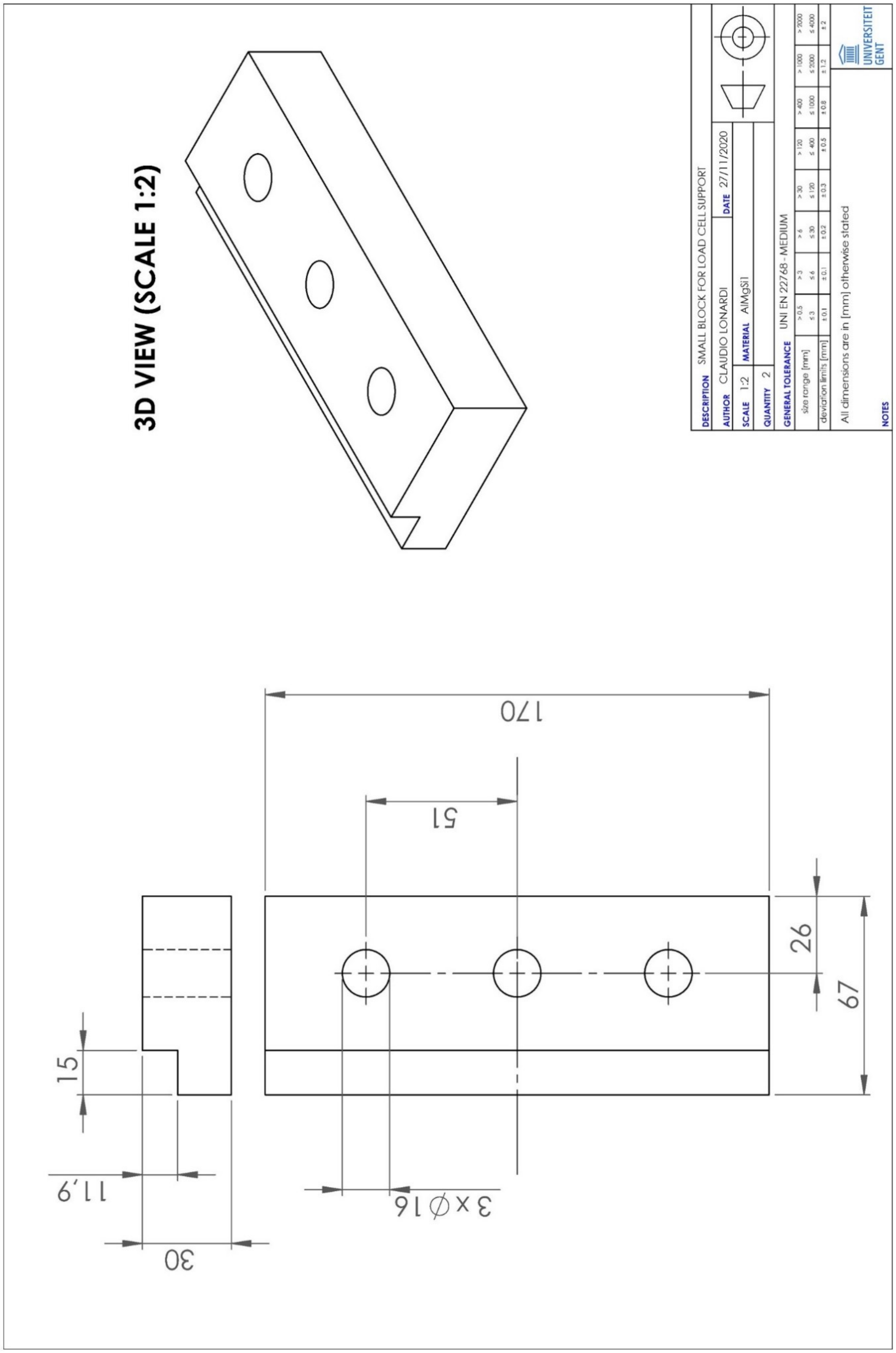
DESCRIPTION	FLANGE FOR CYLINDER SUPPORT
AUTHOR	CLAUDIO LONARDI
DATE	27/11/2020
SCALE	1:2
MATERIAL	S235
QUANTITY	1
GENERAL TOLERANCE	UNI EN 22768 - MEDIUM

size range [mm]	> 0.5 ≤ 3	> 3 ≤ 6	> 6 ≤ 30	> 30 ≤ 120	> 120 ≤ 400	> 400 ≤ 1000	> 1000 ≤ 2000	> 2000 ≤ 4000
deviation limits [mm]	± 0.1	± 0.1	± 0.2	± 0.3	± 0.5	± 0.8	± 1.2	± 2

All dimensions are in [mm] otherwise stated

NOTES

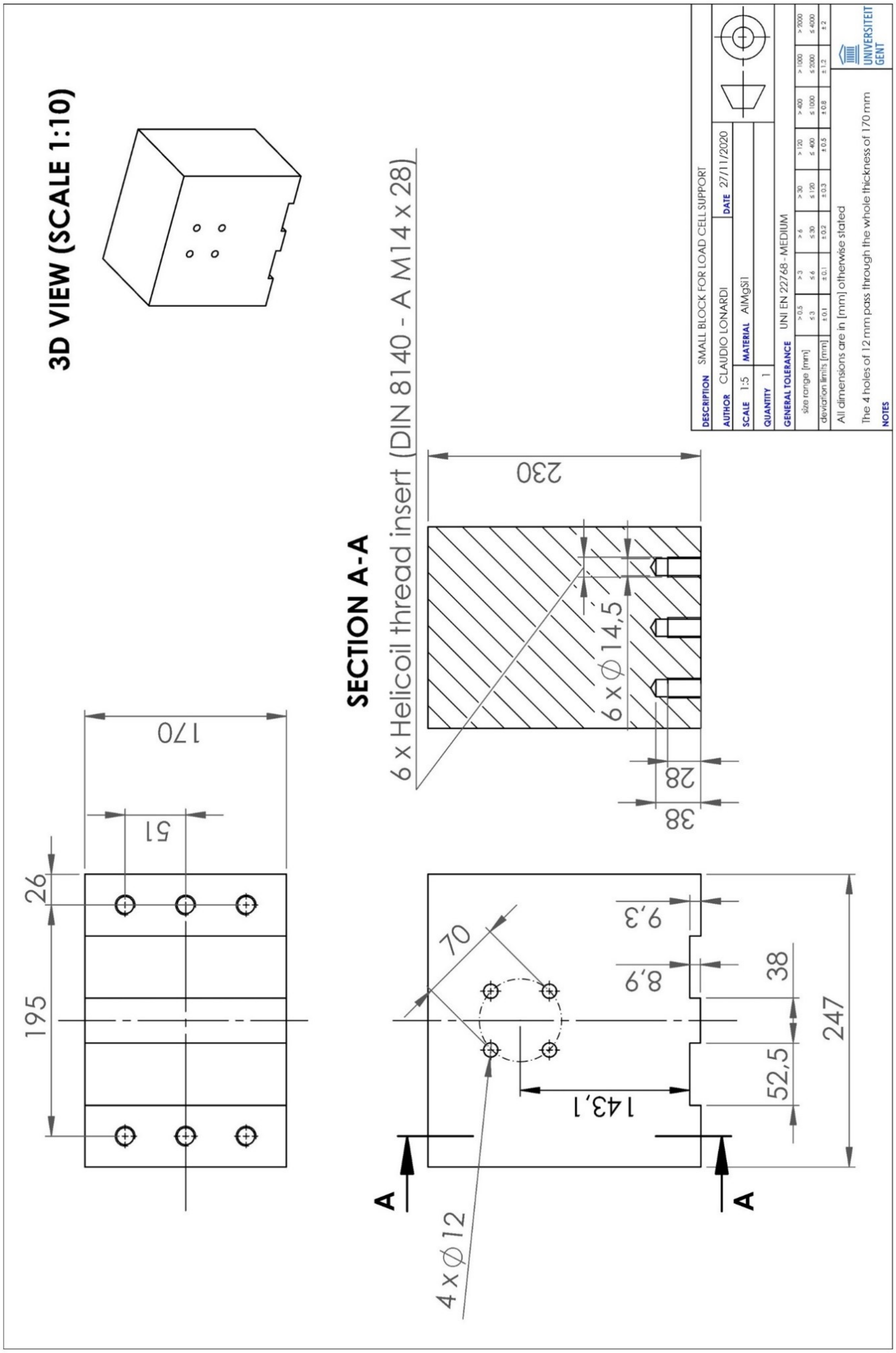
UNIVERSITEIT GENT

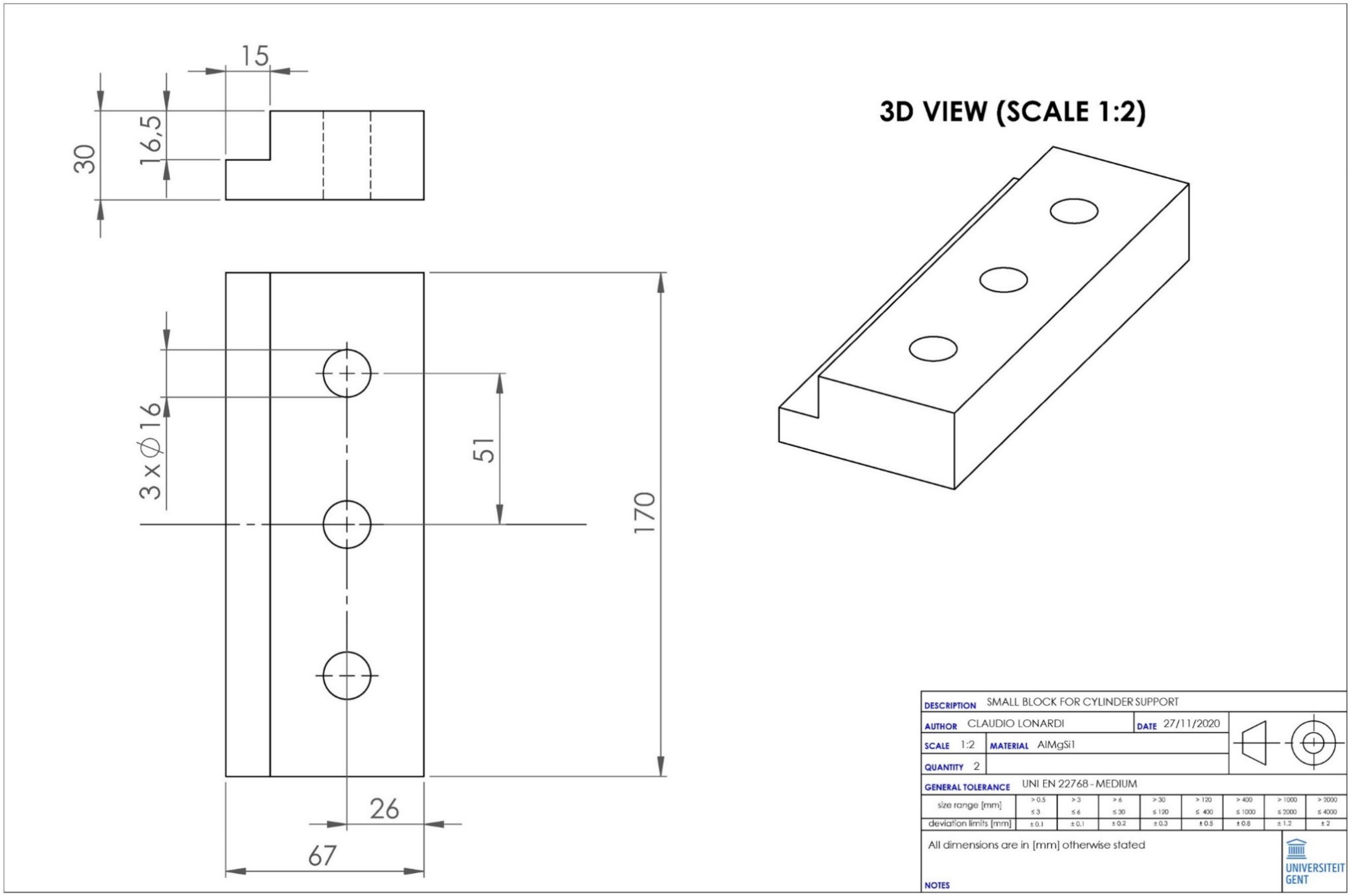


DESCRIPTION SMALL BLOCK FOR LOAD CELL SUPPORT
AUTHOR CLAUDIO LONARDI
DATE 27/11/2020
SCALE 1:2
MATERIAL AlMgSi1
QUANTITY 2
GENERAL TOLERANCE UNI EN 22768 - MEDIUM

size range [mm]	> 0.5 ≤ 3	> 3 ≤ 6	> 6 ≤ 30	> 30 ≤ 120	> 120 ≤ 400	> 400 ≤ 1000	> 1000 ≤ 2000	> 2000 ≤ 4000
deviation limits [mm]	± 0.1	± 0.1	± 0.2	± 0.3	± 0.5	± 0.8	± 1.2	± 2

All dimensions are in [mm] otherwise stated
NOTES


3D VIEW (SCALE 1:10)
SECTION A-A
6 x Helicoil thread insert (DIN 8140 - A M14 x 28)
6 x ∅14,5
230
170
51
28
38
26
195
70
9,3
8,9
38
52,5
247
143,1
4 x ∅12
A
A
DESCRIPTION SMALL BLOCK FOR LOAD CELL SUPPORT
AUTHOR CLAUDIO LONARDI
DATE 27/11/2020
SCALE 1:5
MATERIAL AlMgSi1
QUANTITY 1
GENERAL TOLERANCE UNI EN 22768 - MEDIUM
size range [mm] > 0.5 ≤ 3 | > 3 ≤ 6 | > 6 ≤ 30 | > 30 ≤ 120 | > 120 ≤ 400 | > 400 ≤ 1000 | > 1000 ≤ 2000 | > 2000 ≤ 4000
deviation limits [mm] ± 0.1 | ± 0.1 | ± 0.2 | ± 0.3 | ± 0.5 | ± 0.8 | ± 1.2 | ± 2
All dimensions are in [mm] otherwise stated
The 4 holes of 12 mm pass through the whole thickness of 170 mm
NOTES
UNIVERSITEIT GENT



DESCRIPTION SMALL BLOCK FOR CYLINDER SUPPORT

AUTHOR CLAUDIO LONARDI **DATE** 27/11/2020

SCALE 1:2 **MATERIAL** AlMgSi1

QUANTITY 2

GENERAL TOLERANCE UNI EN 22768 - MEDIUM

size range [mm]	> 0.5 ≤ 3	> 3 ≤ 6	> 6 ≤ 30	> 30 ≤ 120	> 120 ≤ 400	> 400 ≤ 1000	> 1000 ≤ 2000	> 2000 ≤ 4000
deviation limits [mm]	± 0.1	± 0.1	± 0.2	± 0.3	± 0.5	± 0.8	± 1.2	± 2

All dimensions are in [mm] otherwise stated

NOTES

UNIVERSITEIT GENT