UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE INGENIERÍA Y DE ARQUITECTURA

MÁSTER UNIVERSITARIO EN INGENIERÍA AERONÁUTICA

TRABAJO FIN DE MÁSTER

FASTENER CLEARANCE EFFECT EVALUATION ON SINGLE LAP SHEAR STRUCTURES

Author: Jesús MUÑOZ HERRERA
Tutor: Alan DOMÍNGUEZ MONTERO

January 2025

to obtain the degree of Master of Science in Aerospace Engineering

at the Universidad Europea de Madrid

to be defended publicly on Saturday January 25, 2025 at 10:00 AM.

Master Thesis committee:	Prof. A. Dominguez Montero,	UE, supervisor
	Ir. A. ,	UE professor
	Ir. A. ,	UE professor
	Ir. A. ,	UE professor

Contents

ABSTRACT

Manufacturing processes are very important in order to standardize and control the production of the different products. Focusing on the aerospace sector, riveting is one of the most used processes along the entire production cycle of an aircraft. It allows a permanent joint between metallic components and because of its high relevance, the project will be developed around this mechanical operation.

The thesis itself will consist in evaluating some of the parameters that take part inside riveting and study their effects in the model behavior in terms load transfer and stress distribution. Among all these variables, a special focus will be put in the hole clearance effect. Due to its complexity, the analysis will be carried using Abaqus CAE, a software calculator that uses finite element analysis (FEA) to solve the problems. Hence, In order to make this exercise, the first part of the paper will introduce the reader through the different types of rivets and the reasons which explain why this unions are the most efficient inside aircraft assembly.

Secondly, the document will present some of the previous research made regarding the hole clearance effect, its consequences and the magnitude of its effect, which indeed will be useful in order to corroborate the thesis results. Afterwards, the software calculator will be explained in detail with the aim of understanding all the procedures and properties belonging to its interface, which will be used later on to bound the type of calculation that will be run and the simulation schema, divided into subsequent time steps.

Once all the background from both the riveting process and FEA has been presented, the third chapter will start introducing the analytic solution for single lap shear riveted structures, followed by the model presentation, defining the elements which conform the assembly and the process used to build the specimen used for the calculation. After finishing creating a template model using CAE module from Abaqus, Python will become the working tool used from that point until the final model definition. By means of programming, the work load could be reduced and the model variations could be controlled and bounded by the design variables, or in other words, the parameters which mostly influence the riveting process and will conform the design of experiment (DOE).

After selecting strategically the values for the model design variables, several trials will be run in order to identify and quantify the effect of the chosen variables, specially the hole clearance. Finally, when results are obtained, these can be analyzed and afterwards translated to conclusions based on the experiments performed, which can be summarized in the following points:

- Introducing a clearance between the shank and the hole will lead to a increase in the highest stress measured in the model.
- Consequently, the maximum load carried by the crtical fastener will be reduced.
- The bigger the rivet diameter is, the larger is the load sustained by the shank. Nonetheless, it is important to keep an eye on the rivet head since it can become critical when increasing the shank diameter.
- Pretension plane location with respect to the fastener end parts (head and collar) will determine the stress distribution along the rivet.
- When plate thicknesses are different, pretension plane shifts position, causing a reduction in the maximum load carried by the critical fastener.
- Among all effects studied, hole clearance dominates and dictates model behaviour and both stress and shear load distributions.
- This clearance will increase the stress in the region close to the edge where the fastener shank and the head are joint together.
- Since this last region is the most critical in terms of failure, clearance will make material to fail earlier.

RESUMEN

Los procesos de fabricación son muy importantes para estandarizar y controlar la producción de los diferentes productos. En concreto, en el sector aeroespacial, el remachado es uno de los procesos más utilizados en todo el ciclo de producción de un avión. Permite una unión permanente entre componentes metálicos y, debido a su importancia, el proyecto se desarrollará en torno a esta operación mecánica.

La tesis en sí misma consistirá en evaluar algunos de los parámetros que intervienen en el remachado y estudiar sus efectos sobre el comportamiento del modelo en términos de transferencia de carga y distribución de esfuerzos. Entre todas estas variables, se prestará especial atención al efecto del hueco entre los remaches y sus agujeros correspondientes. Debido a su complejidad, el análisis se realizará utilizando Abaqus CAE, una calculadora de software que utiliza el análisis de elementos finitos (AEF) para resolver los problemas. Por lo tanto, con el fin de hacer este ejercicio, la primera parte del documento guiará al lector a través de los diferentes tipos de remaches y las razones que explican por qué estas uniones son las más eficientes en el ensamblaje de una aeronave.

En segundo lugar, el documento presentará algunas de las investigaciones anteriores realizadas sobre el efecto de la presencia de holguras en los agujeros, sus consecuencias y la magnitud de su efecto, que serán útiles para corroborar los resultados de la tesis. Después, se explicará detalladamente el programa de cálculo con el fin de comprender todos los procedimientos y propiedades que pertenecen a su interfaz, que se utilizarán más adelante para vincular el tipo de cálculo que se ejecutará y el esquema de simulación, dividido en intervalos de tiempo consecutivos.

Una vez que se ha presentado todo el trasfondo del proceso de remachado y del AEF, el tercer capítulo comenzará con la presentación de la solución analítica para una estructura de cortadura simple remachada, seguida de la introducción del modelo, definiendo los elementos que conforman el conjunto y el proceso utilizado para construir la muestra utilizada en el cálculo. Después de terminar la creación de un modelo plantilla utilizando el módulo CAE de Abaqus, Python se convertirá en la herramienta de trabajo utilizada desde ese momento hasta la definición del modelo final. Mediante la programación, reduciendo la carga de trabajo se controlan las posibles alteraciones del modelo y se limitan mediante las variables de diseño, es decir, los parámetros que más influyen en el proceso de remachado y que se ajustarán al diseño experimental (DE).

Después de seleccionar estratégicamente los valores para las variables de diseño del modelo, se ejecutarán varias pruebas para identificar y cuantificar el efecto de las variables elegidas, especialmente la existencia de una holgura entre remache y agujero. Finalmente, cuando los resultados se obtienen, estos pueden analizarse y luego traducirse a conclusiones basadas en los experimentos, las cuales se puede resumir en los siguientes puntos:

- Introducir una holgura entre el vástago y el agujero provocará un aumento en el esfuerzo más alto medido en el modelo.
- En consecuencia, se reducirá la carga máxima soportada por el remache crítico.
- Cuanto mayor sea el diámetro de la caña, mayor será la carga sostenida por el remache. Sin embargo, es importante tener cuidado con las cabezas de estos, ya que pueden convertirse en zonas críticas al aumentar el diámetro del vástago.
- La ubicación del plano de pretensión con respecto a los extremos del remache (cabeza y collar) determinará la distribución de tensiones a lo largo del remache.
- Cuando los espesores de las placas son diferentes, el plano de pretensión cambia de posición, provocando una reducción en la carga máxima transportada por el remache crítico.
- Entre todos los efectos estudiados, la holgura en los agujeros domina y dicta el comportamiento del modelo, tanto su distribución de esfuerzos como las fuerzas de cortadura sufridas por cada remache.
- La introducción de una holgura aumentará los esfuerzos en la zona cercana al borde que conforma la unión entre la caña y la cabeza del remache.
- Dado que esta región es la más crítica de todas, el aumentar los esfuerzos en esa región hará que el fallo del material ocurra antes.

DEDICATION

Firstly, I want to start thanking my mentor, whose guidance and expertise have been invaluable. Your unwavering belief in my abilities and your constant push for excellence have challenged me to grow as a scholar and researcher. Since I firstly attended to your lectures I decided you were the one I wanted to end this academic stage with, and I am glad and extremely grateful you made it happen.

I want to dedicate this thesis to the unwavering support and love of my entire family, who have been my rock throughout this journey. Their encouragement and understanding have been instrumental in my pursuit of knowledge and academic excellence, and thus, I hope you can feel proud about the effort I put in this project.

I extend my deepest gratitude to my sentimental couple, my best friend and all the colleagues I had the pleasure to meet during this trip as a student, for letting me learn from them becoming not only a good student but a well-rounded individual.

Moreover, I dedicate this thesis to my first ever Math, Physics and Chemistry teacher, Jesús Cervera Pérez, who trusted me and encouraged me to show all the potential I had in order to become an engineer in the future. I hope you can feel proud of this work and keep teaching many more students with the passion you do.

Lastly, I want to thank all professionals that spend their time educating people. I am not only referring to teachers, but also parents, coaches and on the job mentors among others, which put all the resources available on the table to make other people's life better. And despite not being the ones who benefit the most, they are the last to give up. Charged with patience and humility they finally show us the path to success, leading to a personal improvement. And at the end, none of them made the work to take credit of that, but instead, they use this satisfaction caused on us to keep helping other people.

Therefore, I want to dedicate these lines from the Matthew's gospel to all these people I referred to in order to make you participants of this work, since without your contribution in making the man who I am, I would not have reached this point of my life. I feel lucky I found these people in my way, and I deeply hope all you keep lighting other people's path the same way you did with mine.

Matthew 6: 5-6

And when you pray, do not be like the hypocrites, for they love to pray standing in the synagogues and on the street corners to be seen by others. Truly I tell you, they have received their reward in full.

But when you pray, go into your room, close the door and pray to your Father, who is unseen. Then your Father, who sees what is done in secret, will reward you.

1 Introduction

This first chapter focus on the importance of metallic structures in the aerospace industry. Although composite materials are becoming more and more common inside new aircraft designs, there are still some regions that are prone to be covered by metals. Such examples are the fuselage frames, wing spars or other aircraft instances mentioned along the chapter.

Thus, next step is covering which are the most popular methods used to join metallic parts. There are two ways of joining metallic parts, which are heat or mechanical force application. Soldering or welding belong to the first kind, whereas riveting is included inside the second family. Despite these two being currently used, the strength and metal heat tolerance are the main factors which explain why mechanical joints appear more often, specifically the rivetted union.

Therefore, having shown the relevance of the rivets, the chapter proceeds explaining the different kinds of rivets, showing their shape and describing its installation procedure. This last is very similar from one another, since for the vast majority of the rivets, firstly a hole needs to be drilled so then the rivet is introduced. The main difference between the different types of installation is the way pretension is applied.

Consequently, given the nature of the installation procedure, the bigger the hole, the easier will be the installation. Because of that reason, the holes are always drilled with a certain tolerance to let the rivet go through. However, what are the consequences of letting a greater clearance between the rivet and the hole walls? This thesis will try to answer this question, and so the thesis will be developed around this sentence. To solve this question, this chapter will explain in some points which will be the main steps to follow. These steps can be summed up in the following points:

- Create an Abaqus model which simulates the riveting process between two plates (single shear).
- Develop a Python code to be able to parametrize the models in terms of the variables that influence the riveting operation.
- Make a Design of Experiment (DOE) and consequently create some sample models to evaluate the effects of the most relevant parameters.
- Come up with the conclusions stating clearly the effect of the hole clearance.

The next section of this chapter includes some previous researches about this topic and what were the results. Of course each of these three experiments where approached from different points, but all of them conclude that the hole clearance effect decreases the fatigue life or life before fatigue failure. Therefore, these solutions coming from previous works will be useful to determine if the results obtained in this paper are coherent or not.

Last section of this chapter summarizes the thesis content, where all chapters are briefly introduced, stating the most important points of their content.

1.1. Motivation and objectives

The aerospace science has been in continuous development from the past centuries until currently, and still looks forward to enhancing the experience of flying, by increasing its related safety or the maneuverability of the aircraft.

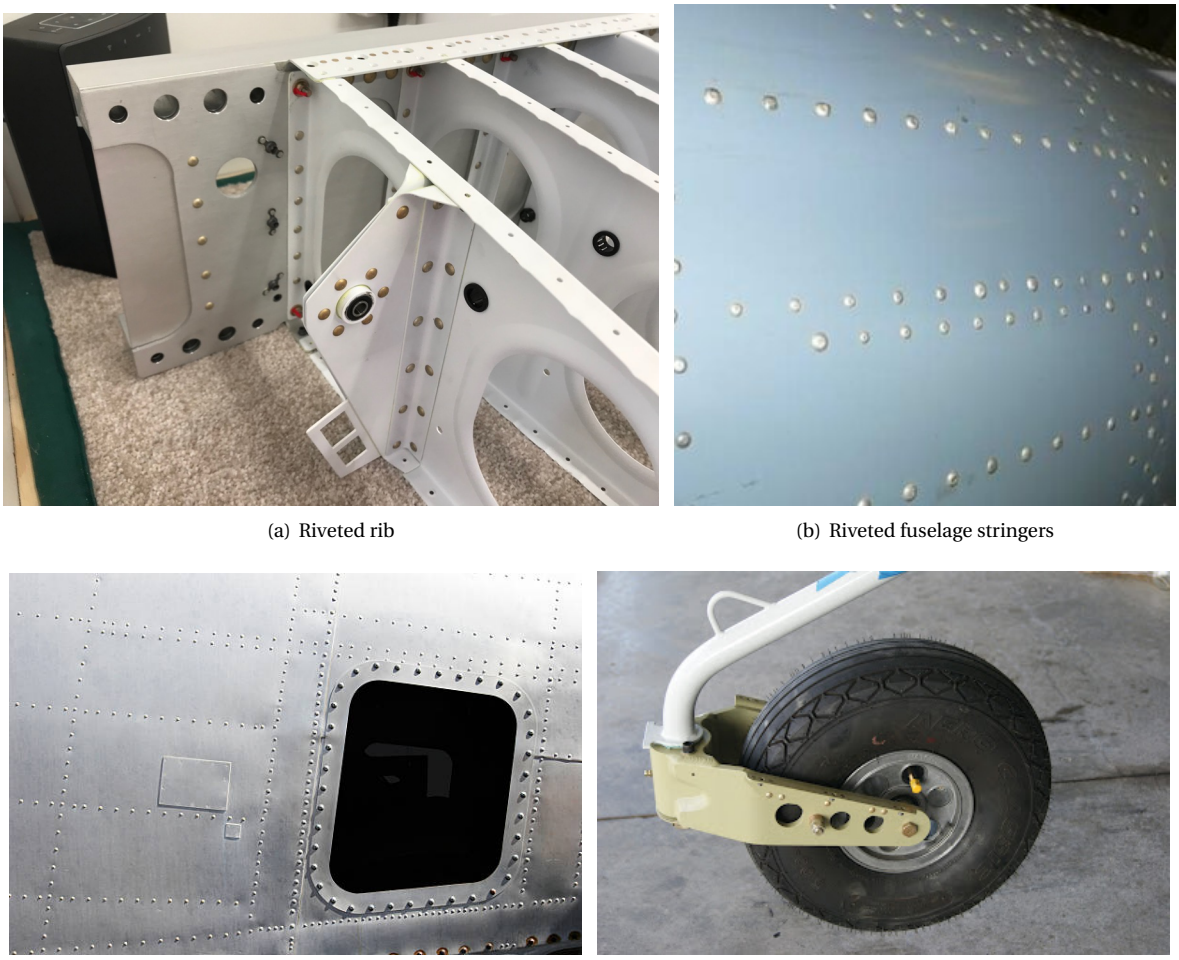
Among the different topics which exist inside this wide field of knowledge, aerospace materials has been a branch which has evolved a lot these past decades. Starting from purely metallic aircrafts, the appearance of composite materials has opened the door to new models and new opportunities inside aircraft design. In fact, composite materials are currently very present in many parts of the aircraft models. They can be present in some parts of the aircraft like the case of Eurofighter Typhoon, whose wing spars are made of composite material, or their content can be risen up to the point of being more than half of the entire plane material.

However, despite the rising-up utilization of composite materials, there are still many places inside the aircraft where metallic materials will keep appearing. Some examples of aircraft instances where metallic materials appear are:

- Fuselage frames
- Fuselage and wing panels
- Wing spars
- Wing ribs
- Fuselage stringers
- Aircraft hatches
- Landing gear

(a) Riveted fuselage panels (b) Riveted spar

Figure 1.1: Riveted structures



(a) Riveted rib

(b) Riveted fuselage stringers

(c) Riveted aircraft hatch

(d) Riveted landing gear

Figure 1.2: Riveted structures

And when dealing with metallic structures, there are two main types of part bonding. The parts can be bonded by applying heat, which is the case of welding or soldering. On the other hand, there exist mechanical bonding, which by means of pretension forces, two elements are mechanically joint. These pretension forces work as shown in figure 1.3. The fastener is subjected to a traction force along its longitudinal axis, which makes the mating plates come together. Indeed, as it is observable, the plane where pretension occurs coincides with the one where the two plates contact at first.

This aspect will be key in the future so it is important to highlight that during a pretension:

- The fastener works under traction.
- The pretension plane is located where both plates mate together.

As it will be mentioned afterwards, figure 1.3 shows a single shear structure, which will be the one analized in this project.

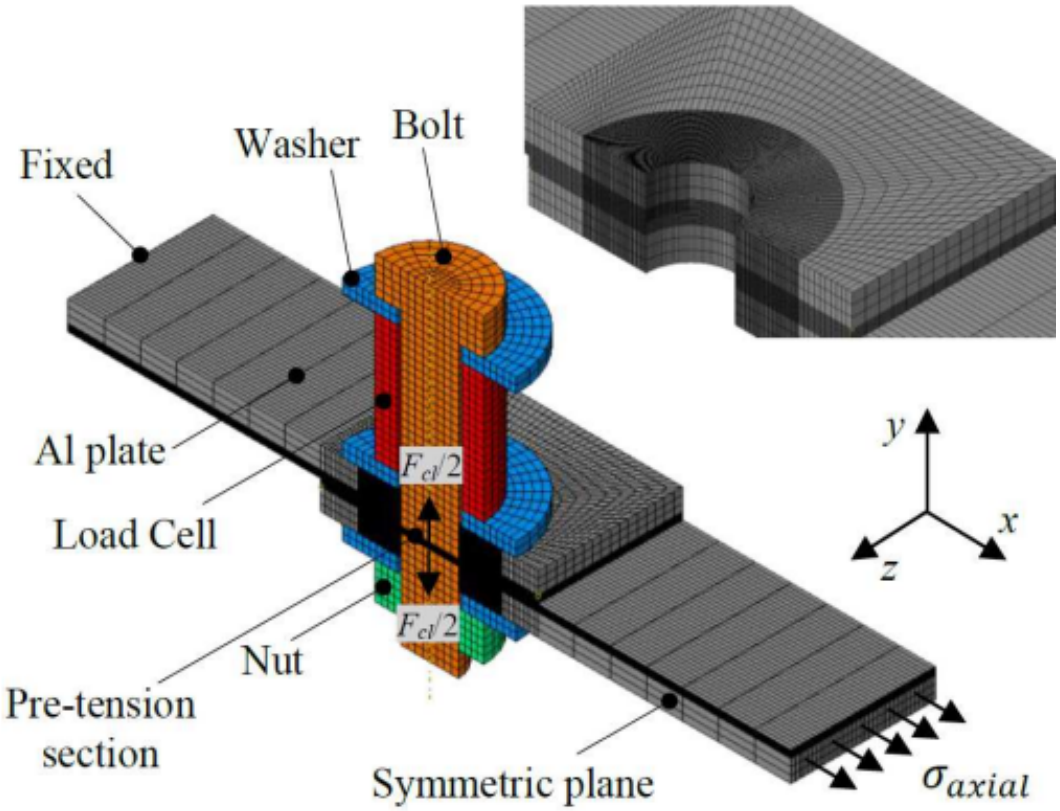


Figure 1.3: Pretension diagram [1]

However, depending on the matter, one type of bonding (whether mechanical or thermal) will be more suitable than the other alternatives. Nonetheless, it is observable that when dealing aerospace assemblies, this last mechanical joint is the one which is used the most of the time, and that is the reason why this project will focus on studying that phenomenon.

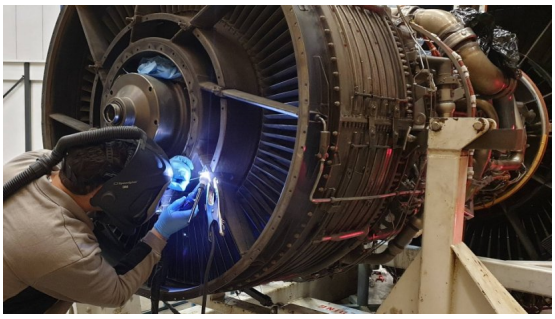
But firstly, let's start with some definition of these processes previously mentioned:

- Soldering: Joining of two metal pieces by means of a third material (filler metal) which has a lower melting temperature. Thus, when applying heat, this third metal will melt while having contact with the first two parts so, when cooling down, it makes the three pieces to be joint in an assembly.

- Welding: Joining of two metal pieces by applying heat directly to them. It follows the same principle of soldering but instead, no other filler metal is used.

- Riveting: Permanent union of two pieces made by applying a pretension force to a third element which allows the joint called rivet.

(a) Soldering



(b) Welding

Figure 1.4: Non-mechanical joints

As stated previously, the main differences between the two groups is the way the parts are assembled, whether using mechanical forces or by means of heat application. But why are the riveted unions mostly preferred? Here are some of the reasons that withstand the usage of this mechanical method [2]:

1. **Aluminium does not tolerate well the heat**: Aluminium is widely used in the aerospace industry because of its low density, which allows it to be present in some points where mechanical stresses are not that high, which is the example of the fuselage covers. Therefore, since this metal is widely used along many instances of the aircraft, it is important to consider its main limitations. Despite behaving good under the mechanical loads it is subjected, the heat makes these properties go down and this happens so with the heat absorbed during welding or soldering. Because of this reason, riveted structures are commonly preferred.

2. **Riveted unions are stronger**: Meanwhile soldering or welding only joins the parts by their exterior surfaces, a rivet has a contact zone that goes across the parts, which make it a stronger bond.

3. **Easier inspection**: When using riveted unions, these can be easily checked by a visual inspection. However, in the case of welding or soldering it is difficult to define the status of the bond without using a special process or a non-destructive test (NDT) in order to check possible cavities in the assembly. This problem does not happen with the rivets since its shank will tend to fill the space available in the hole.

However, these reasons support the fact of using mechanical joints instead of thermal ones. But why are not screws used instead of rivets? And the reason behind this is the tolerance to vibrations. As mentioned previously, the rivet shank will expand to fill the gaps in the hole which will make it more adequate to resist vibrations, whereas the rivets are joint to the parts by its threads, so when vibrations appear, this can cause wear in those threads, so they become softer until the screw comes loose [2].

Therefore, the rivets are most common option for permanent assembly of metallic structures. There exist a wide variety of rivets, some of them are the ones listed hereafter [3]:

- **Solid rivets**: These are the oldest and most common ones. They have a head and a shank that, in order to be installed, has to be deformed with a hammer or other tool depending on the strength of the material the rivet is made of.

Figure 1.5: Solid rivets

- **Blind rivets**: These rivets have a mandrel which is introduced inside the riveting pistol in order to shrink the shank of the rivet located at the opposite side of where pistol is situated, so the mandril separates from the head once it is installed.

Figure 1.6: Blind rivets

- **Drive rivets**: These rivets have a central mandrel on top of the head, which is push with a hammer towards the head as the rivets locks the assembly, making it simple to perform the installation.

Figure 1.7: Drive rivets

- **Self-piercing rivets**: These rivets are ideal for the applications where no hole has been drilled before. During its installation, the lower part of the rivet is deformed into the shape of a skirt, so the material is penetrated and the fastener is locked.

Figure 1.8: Self-piercing rivets

- **Tubular rivets**: They have a hollow tube in the part of the shank, and that hollow part can go through the head as well, and these rivets are cold worked in order to perform the assembly.

Figure 1.9: Tubular rivets

These were the most common rivets, but there exist more fasteners which are specially utilized in the aerospace sector which are:

- **Hi-lock and hi-lite rivets**: These rivets have a thread in the end of its shank so a collar can be installed. This collar is installed by applying torque so it goes through the shank up to a point where it breaks, leaving the rivet installed.

Figure 1.10: Hi-lock rivets

- **Huck rivets**: They possess a threaded mandrel which is inserted in the riveting pistol. This thread allows the tool to apply torque to the rivet, so it shrinks the shank until installation is complete, and the mandrel breaks apart from the rivet head leaving the fastener installed.

Figure 1.11: Huck rivets

These rivets can appear in the assemblies alone joining the model sub-instances, but they can appear as well in the presence of other fasteners. The most popular case is the use of anchor nuts, and in the aerospace field this is a very common situation.

Anchor nuts are composed by a center hole where the main fastener will be located, accompanied with two small holes located at its sides. The main fastener can be another rivet, but most of the times this fastener tends to be a screw. In this last scenario, the anchor nut would have a built-in hole to make possible the screw installation. However, the other two smaller fasteners are anchor nut rivets, which are commonly known as "*cherrys*", due to the name of the main supplier of this kind of rivets (Cherry Aerospace).

This way, this mechanism allows the manufacturer to introduce another type of fastener, whose position is then fixed by the two anchor nut rivets placed at its side. The most common anchor nuts have a fixed built-in hole for the main fastener. Nonetheless, there exist floating anchor nuts [4], like the one shown in figure 1.12 which allows a slight movement of the main hole axis between the cherry positions in order to make easier the fastener alignment.

Figure 1.12: Floating anchor nut [4]

For this last one the installation procedure is the following one [4]:

1. Install the anchor nut rivets in their corresponding positions so the anchor nut is fixed in the space.

2. Then, the screw or the threaded fastener can be installed, playing with the slight axis adjustment this element allows

Now, having this said, it have been stated the reasons behind the importance of using riveted structures, so let's focus on the project itself.

The project will develop around the analysis of a basic riveted structure sketched in figure A.2, which explained more in depth in appendix A. The model contains two metal plates, shown in blue, which are joined together by means of three rivets, shown in green. The upper panel is clamped to its left edge as the lower panel is pulled by an external force to the right of the paper.

The project will consist on modelling both the two metal sheets as well as the rivets which join them and, afterwards simulate the pretension process and the model behavior when suffering a traction force which tries to separate the assembly. After selecting some of the key parameters that mostly influence this phenomenon we will perform a parametric analysis, varying the model and understanding what is the influence of these specific values. Along the project different approaches will be used and all of this with the aim of understanding in deep what are the main concerns during a riveting process.

Consequently, the main steps for this project are the following ones, which will be achieved in this sequence:

1. Develop a realistic model in Abaqus CAE which precisely describe the riveting process.

2. Understand the necessary inputs and the procedure the software uses to operate to anticipate problems and thus adhere the model to real world physics.

3. Perform a parametric mesh in Python environment using RPY file retrieved from the specific job performed in Abaqus CAE.

4. Make a proper DOE (Design of Experiment) and identify both the key variables that define the riveting process and its corresponding range of interest.

5. Evaluate and quantify the influence of each of the selected parameters to end up with a brief conclusion stating the most important effects which should be considered when performing this kind of mechanical process.

Among all of the parameters which will be analyzed, the most important one will be the effect of leaving some clearance between the hole and the shank of the rivet, that as it will be seen in the next section, it has a high relevance during installation procedure. Every time a rivet is installed, the hole needs to be greater for the rivet to be introduced, but the tendence is trying to keep the gap as close as possible. This gap can be

observed in figures 1.13 and 1.14 where it is easy to check that the diameters of the fasteners and the ones of the hole are different.

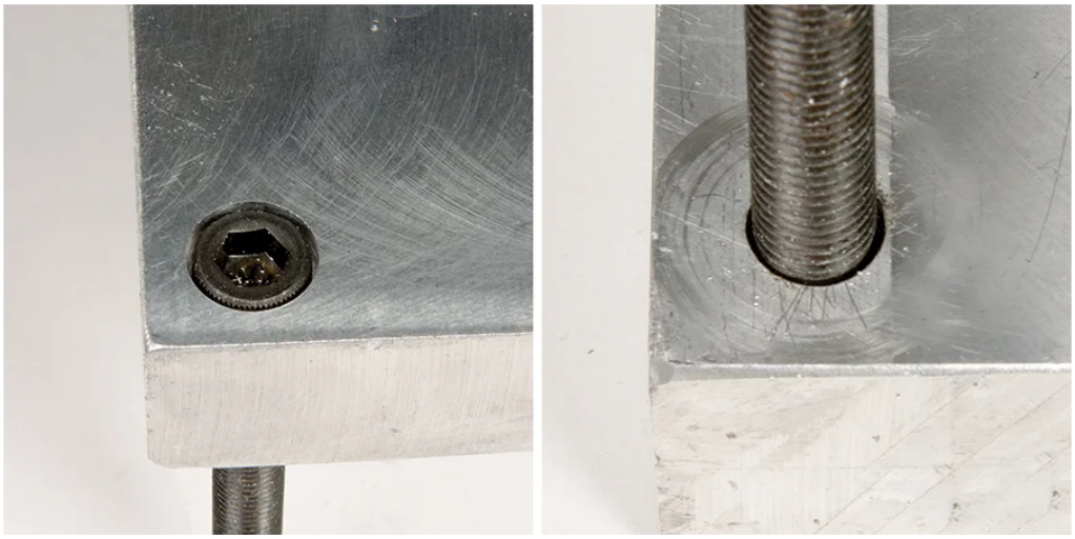


Figure 1.13: Right side shows the hole drilling and the left side shows the installation

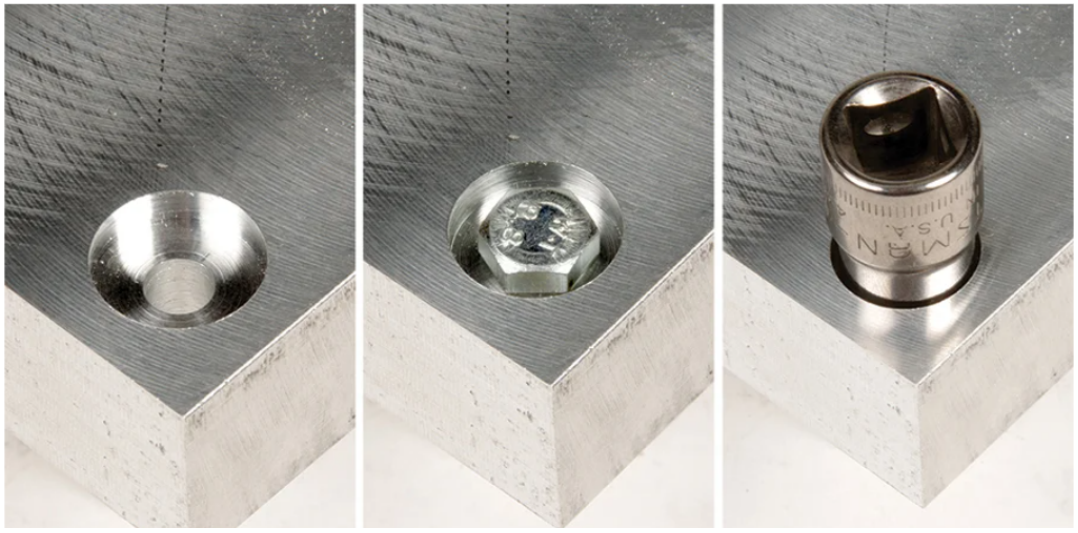


Figure 1.14: Fastener and socket installed with hole clearance

Nonetheless, it is not trivial to quantify the effect of making that clearance bigger and because of that lack of intuition, most of the times the manufacturer uses small tolerances. On the other hand, trying to keep the margin between both dimensions close entails a more difficult process and sometimes problems in the installation. If a manufacturer drills definitive at a certain diameter and that diameter ends up not being enough big for the fastener to fit, this cause a stop in the production line and sometimes the need of raising a waiver which will describe a process to fix the problem. Therefore, the main aim of the project is the following one:

- **Evaluate the clearance effect between hole and rivet and quantify its influence in stress concentration and rivet shear force.**

1.2. State of the art: Effect of hole-rivet clearance

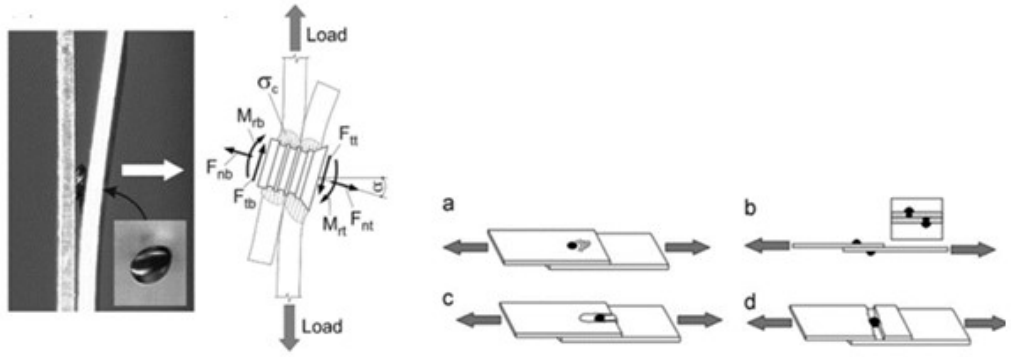
This aspect has been studied in several projects and this section of the chapter will discuss some of them in order to come up with some conclusions regarding what it should be expected when implementing this

procedure.

In the article "*The effect of rivet gun operating pressure and hole clearance on rivet shear strength in sheet metal riveting process*" developed by Saputro et al. it was stated that several parameters influence the riveting process, and these are the rivet length, the rivet diameter, the pretension force and hole clearance, among others [5]. Therefore, that article focused its evaluation on some of those parameters in order to study their effect on the installation in terms of stress.

Firstly, some considerations are have to be made when performing the riveting operation [5]:

- When performing the riveting, the plates will bend due to the momentum exerted during the operation, leading to possible plate misalignment and follower forces due to tension. This aspect can be understood watching figure 1.15(a) which shows the force and moments diagram during riveting.
- The rivets cannot be placed in the close to the edges, because of crack propagation and failure could occur as shown in figure 1.15(b).



(a) Forces and moments during riveting

(b) Fracture during shear application

Figure 1.15: Riveting information [5]

Once these consideration are taken, the experiment performed in this article consisted in building a specimen in order to recreate a three plates installation with a solid rivet. The material used for the rivet was an aluminium with properties similar to the ones of AISI 1012 standard. With the material already defined, the dimensions of the specimen, containing of course the drill through which the rivet will go through, are specified in figure 1.16. Considering this assembly, the operation parameters that were varies in order to analyze their corresponding effects were the pressure exerted in the riveting and the hole clearance.

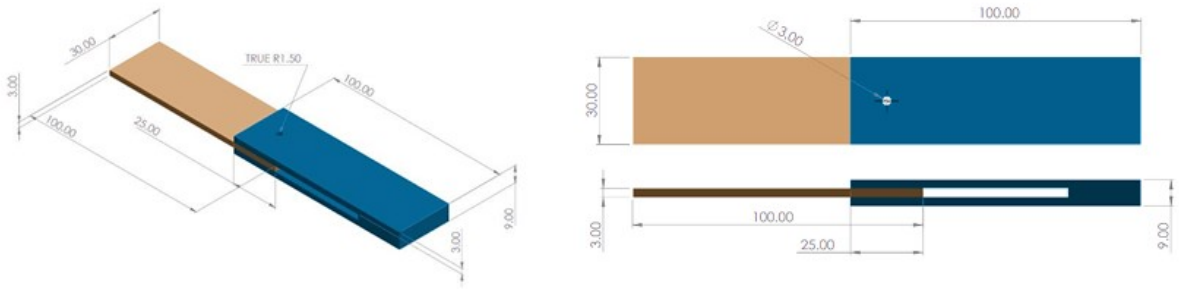


Figure 1.16: Specimen used to perform the analysis [5]

Each of these two variables adopted three different values, thus making a total of nine different combinations. In order to get a solid range to verify the results, each of the nine combinations were tested three times, and the lowest value of those tests was the one used to make the final conclusion. The twenty seven experiment results can be summarize in figure 1.17 shown below, where the property that was studied was the rivet shear strength.

Variation		Shear Strength (MPa)		
Pressure (MPa)	Hole Clearance (mm)	1st Test	2nd Test	3rd Test
0.1	0	103.76	98.76	77.12
0.15	0	229.65	257.09	189.14
0.2	0	273.75	248.92	250.23

(a) Results for hole null hole clearance

Variation		Shear Strength (MPa)		
Pressure (MPa)	Hole Clearance (mm)	1st Test	2nd Test	3rd Test
0.1	0.1	185.87	212.66	247.94
0.15	0.1	195.67	244.02	208.09
0.2	0.1	234.87	230.63	234.55

(b) Results for 0.1mm hole clearance

Variation		Shear Strength (MPa)		
Pressure (MPa)	Hole Clearance (mm)	1st Test	2nd Test	3rd Test
0.1	0.2	255.78	249.25	189.79
0.15	0.2	211.68	244.35	245.98
0.2	0.2	244.02	247.29	238.79

(c) Results for 0.2mm hole clearance

Figure 1.17: Results for all the samples tested [5]

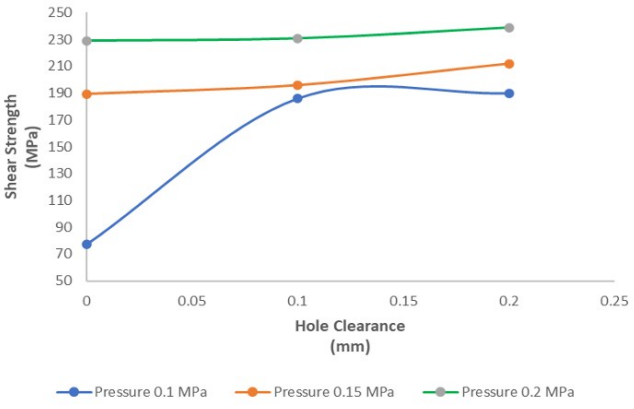
If the minimum value for each of the nine case is picked (those are the ones highlighted in figure 1.17) the results can be plotted, which is easier then to make solid conclusions out of the analysis. Figure 1.18 shows how the shear strength of the rivet varies along the domain of analysis, and thus the following conclusions can be made:

- Rivet shear strength increases with the riveting pretension force.
- Hole clearance increases as well the rivet shear stress.

Before starting the experiment, these consequences may be trivial, but the outcome of these projects is not only to identify the tendencies, but also to quantify their effects in terms of rivet loading. Therefore, for this case, the effect of hole clearance was more harmful for the lower pretension values, so it can be seen that there exist a relation between hole clearance and pretension force which both may contribute in the same way, but they do have influence between them.

Variation		Shear Strength (MPa)		
Hole Clearance (mm)	Pressure (MPa)	1st Test	2nd Test	3rd Test
0	0.1	103.76	98.76	77.12
1	0.1	185.87	212.66	247.94
0.2	0.1	255.78	249.25	189.79
0	0.15	229.65	257.09	189.14
0.1	0.15	195.67	244.02	208.09
0.2	0.15	211.68	244.35	245.98
0	0.2	273.75	228.92	250.23
0.1	0.2	234.87	230.63	234.55
0.2	0.2	244.02	247.29	238.79

(a) Table with the final values



(b) Plot of the results

Figure 1.18: Final results [5]

In the article "*Effect of the rivet-hole tolerance on the stress-severity factor*" by Behal et al. a similar analysis was performed [6]. This time it not only measured the specific load transferred by the rivet but as well the number of cycles until failure. All of these aspects were tested as a function of the hole clearance. The test consisted in a riveting operation of thin metal plates so afterwards, it could be tested with an harmonic excitation to check the number of cycles required to reach the failure.

Before proceeding with the fatigue analysis, they measured the load transferred by the rivet after its installation, and the results were the ones shown in figure 1.19. Despite having data spread all over the domain, a linear regression was performed in order to come up with the tendency and a general formula to quantify the reduction in load carriage in terms of the hole clearance.

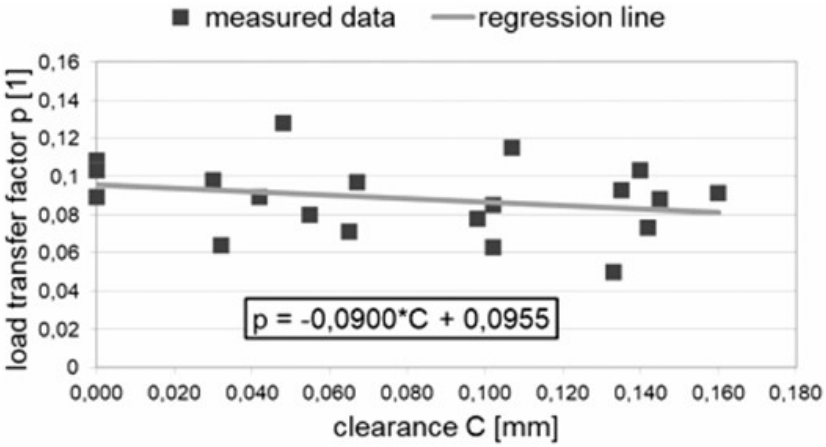


Figure 1.19: Load transfer factor vs hole clearance [6]

Having seen these results, it was expected that this negative tendency would also be reflected in the fatigue life analysis. Therefore, after performing the test the results were plotted in figure 1.20. As mentioned, the fatigue life of the elements was reduced as the hole clearance was increased. Hence, the hole clearance not only is harmful for the stress carriage, but as well as a consequence it also reduces the fatigue life of the assembly.

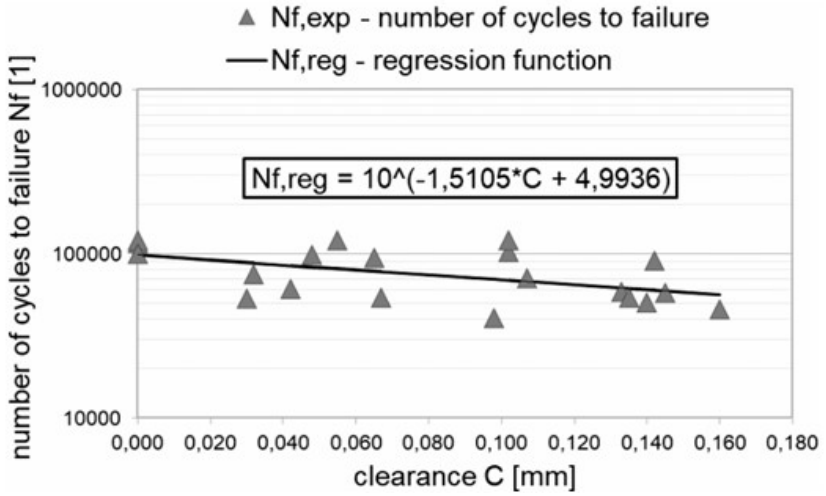


Figure 1.20: Fatigue life factor vs hole clearance [6]

Some of the reasons why clearance affect negatively the riveting stress concentration are discussed in the article "*The structure of the strength of riveted joints determined in the lap joint tensile shear test*" developed by Mucha et al [7]. One of the consequences of leaving clearance in the hole is that we are allowing the rivet to move inside the hole, letting its axis to not be strictly perpendicular to the plate surface. Therefore, when a certain tensile load occurs, the differences can be observable in figure 1.21. When no clearance exists, the

axis of the rivet and sheet normal vector remain parallel, so the loads are carried all over the sheet thickness, and as a consequence of that, it appears a secondary bending moment on the plate side where tensile stresses are minimum. Nonetheless, when a clearance does exist, when the traction force is applied the entire shank surface does not remain in contact with the hole inner surface. This hence, increases the stress concentration in the lower part of the plate, which at the end suffers a local buckling in that region [7].

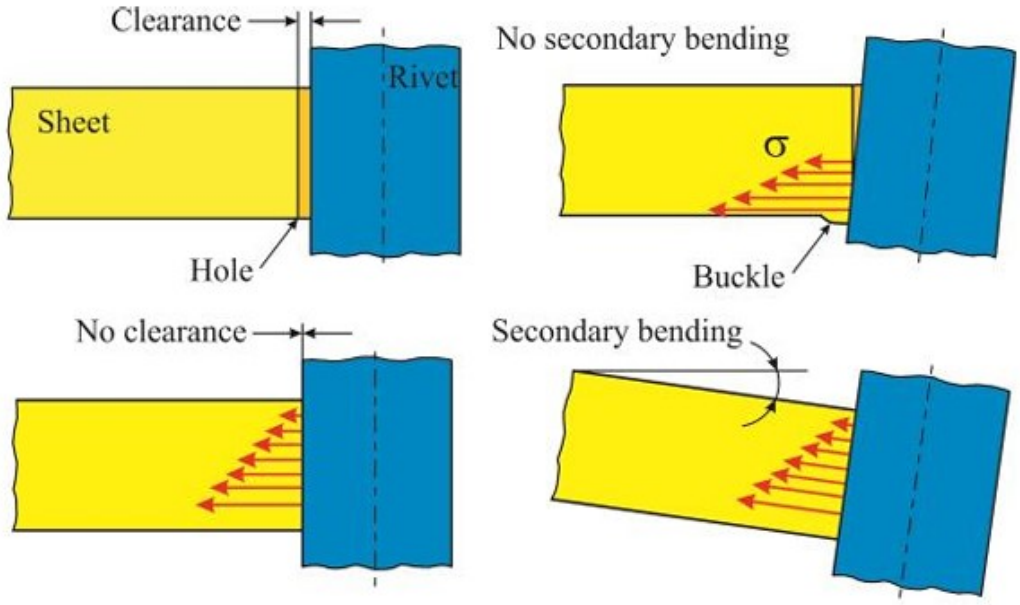


Figure 1.21: Influence of hole clearance in plate stress distribution [7]

With this information it was revealed the effect of that gap between hole and rivet, and how can this reduce the fatigue life of the assemblies, but is there a way to reduce these consequences in order to not make them that harmful?

Since hole clearance in some installations may be unavoidable, there have been some researches made to reduce this impact. Adam Lipski in his paper "*The influence of the degree of the rivet hole sizing on the fatigue life*" proposes the solution of hole sizing [8].

As mentioned previously, before performing the riveting operation the hole need to be drilled in the plate. When performing this drilling, we need to take into account that this dimension selected for the tool will not coincide with the diameter found later in the plate. Because of imperfections during the operation and the action of rotation during the drilling, the plate diameter will always be bigger than the one selected in the first moment before the operation was started.

After eliminating the leftover material, the hole can be made bigger by sizing it using compression forces exerted on the inner layers of the material. These compression stresses are applied using a special tool, and because of its operation requirements, the diameter of the hole shall not be lower than 3 mm. The bigger the hole is aimed to be, the higher the deformation exerted on the plate.

In order to quantify the effect of hole sizing, the artist performed the following analysis. The shank diameter of the test was 3mm, and the hole clearance was set to be 3.1 mm. Consequently, five different samples were created with this hole diameter. However, the difference between them was the process used to reach that diameter. Recalling to previous paragraphs, the hole size is dimensioned by these processes, done in this sequence:

1. Drilling
2. Sizing

In order to quantify the amount of drilling and sizing made in the hole, the following parameter was defined [8]:

$$k = \left(\frac{d_k - d_w}{d_w} \right) \cdot 100\% \tag{1.1}$$

where d_k is the hole diameter after sizing and d_w is that value after drilling.

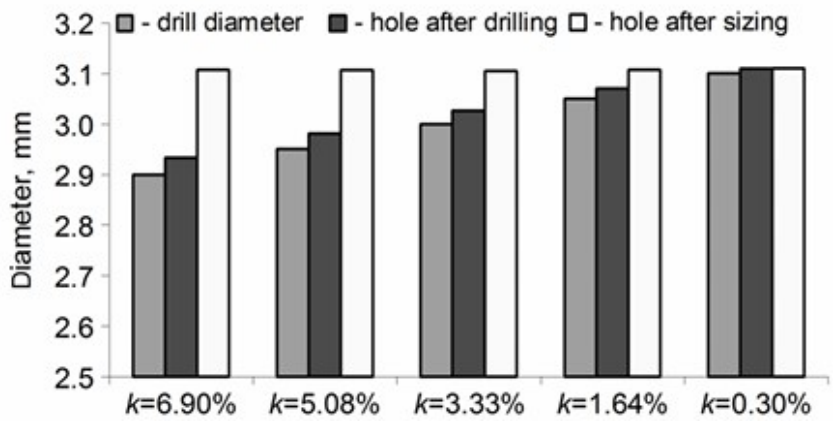
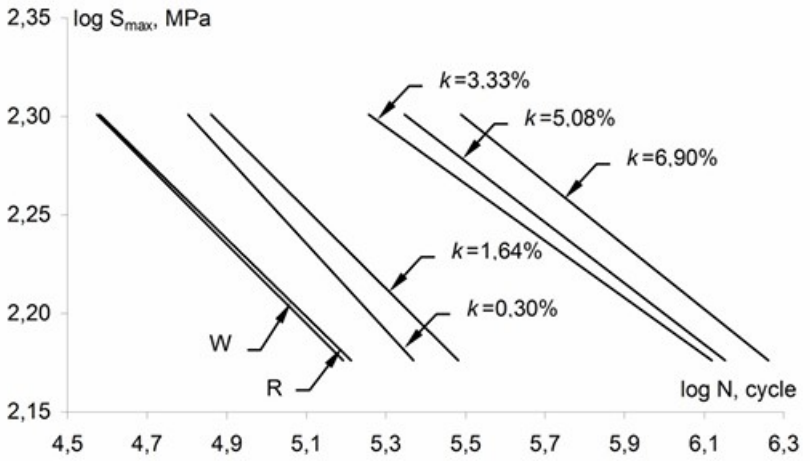
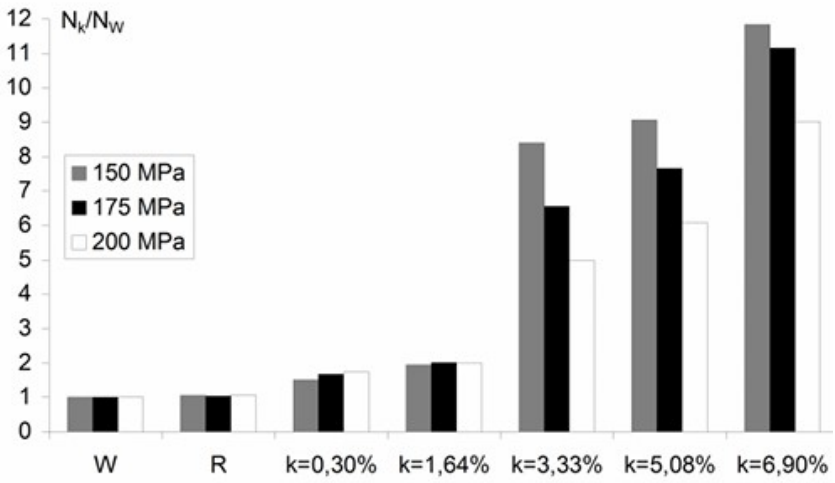


Figure 1.22: Samples with different k factors [8]



(a) Plot of S-N curves for the samples created



(b) Results expressed in a bar chart

Figure 1.23: Results of the sizing analysis [8]

The five samples had a different *k* value, as shown in figure 1.22. Then, they were subjected to a fatigue test with asymmetry factor R=0, with a load frequency of 5Hz. The results of the tests could be summarize in the plot shown in figure 1.23, and it can be seen that, the higher the k value is, the longer the time will be until failure is reached. Therefore, despite sometimes having hole clearance is unavoidable and its consequences will affect negatively the stress behavior of the assembly, there exists manners to deal with that clearance, increasing the fatigue life of the elements by means of processes like the one described, which enters inside the category of cold working or material hardening, which was previously explained in previous section.

After, observing these past works that study the hole clearance effect, it can be seen that the system proposed in figure A.2 is very similar to the one shown in figure 1.16. It will consist in a two part assembly using rivets and, afterwards the entire specimen is subjected to a traction force. The main difference between the researches shown and the model proposed is the number of rivets used, but the same principle used in these papers will be applied to the model of this thesis, and after obtaining the results, they will be analyzed in order to check that the behavior is the one expected based on previous works, which indeed shows that hole clearance leads to a higher stress concentration.

1.3. Summary of thesis content

After checking the previous works regarding this topic, the objective is to define a jobcard used to analyze its effect and check if it is relevant to take into account another parameters that we may find important during the process. Thus, the development of this thesis will be divided in the following chapters summarized right hereafter:

1. **Introduction**: In this first chapter it will be introduced the relevance of the metallic structures and after covering some of the joining processes, it develops the reasons which explain why rivetted assemblies are important and most times, unavoidable inside the aircraft, describing some of the different rivets that exist. This will be the motivation of this thesis.

   Afterwards, the model will be briefly explained in order to understand the analysis procedure, so later on, based on the specimen described, the project objectives can be set. Lastly, this chapter includes previous works regarding this field of study (hole clearance in riveting operations) and their results which will be useful to both anticipate and validate post-analysis results, checking if they are coherent or not. Summing up, what this chapter provides is a short summary of all the previous works and researches in riveting, by means of stating basic definitions needed along the thesis and studying the hole clearance effect in riveting.

2. **Mathematical background on FEM**: Since this thesis will be pursued by means of a finite element method calculator, it is important to understand how it works and its capabilities and limitations. Therefore, this chapter aims to provide a brief background regarding the main aspects of this kind of program. In the first place, the chapter will expose the main differences between FEM and other numerical methods. Afterwards, the next section inside the chapter will describe the three main phases inside the analysis process, trying to develop the physics behind each step and the different options available to model an assembly behavior. Inside the introductory section, it will be explained in depth the pre-processing phase, which is the first of the three phases, since it contains the data necessary to create the model.

   Once the data is sent to the computer, the second section will explain the two main solvers available to deal with the calculation, their characteristics and when is more suitable to use each of them. Lastly, the chapter contains some examples to illustrate the cases where each solver can be used in a real example.

3. **Model generation**: The third chapter will start providing a first approximation to the analytic solution of single lap shear riveted structures, since this will be used later in the conclusions to prove why FEM models were necessary for this kind of project. Afterwards, the following section of the chapter contains all the steps performed to build the final model, which are based on the information presented in the first two chapters. In order to be more illustrative, the model generation performed in Abaqus CAE is supported by appendix A which goes step by step describing more in depth the procedure. In addition to this last information, this chapter includes the design parameters definition, the process performed to create the parametric mesh, the procedure to create future models and the table with the the experiments used to evaluate the effects of these designed variables assigned previously. The last part of this chapter contains the procedure used to extract the shear loads out of the program results, followed by a

table with the shear load values for each of the rivets in the 15 models analyzed. The chapter ends with a section that compares the solutions retrieved from FEA with the ones obtained following the analytic procedure.

4. **Results discussion and conclusion**: In the last chapter of this thesis, the different models are compared between themselves in order to clarify and quantify the effect of each of the design parameters. To do so, several contour plots are used to identify the critical points of the models and the reason behind the obtained results. These contours will be presented strategically since the aim is to isolate each variable effect to check its influence in the overall results. After presenting all 15 models, the chapter will be closed with the final conclusions extracted from the results where it will be summarized the effect of each of the variables intended to be studied.

   Since the objective of this thesis is focused on analyzing the impact of hole clearance in stress concentration and rivet shear loading, these will be the two magnitudes that will be discussed along this last chapter.

2

Mathematical background on FEM

After defining which will be the jobcard to follow to pursuit the objectives mentioned in the past chapter, this part of the thesis will provide a short background regarding numerical methods. Starting with the most simple definition, the different numerical methods are exposed to the reader. Inside this family there exists three main branches which are:

- Finite Difference Method (FDM)
- Finite Element Method (FEM)
- Boundary Element Method (BEM)

Among these three, the FEM analysis is the most commonly used and the one utilized for this project. Thus, a small description of the other two analysis method is given to know how they work but after that, the chapter is centralized to finite element analysis (FEA).

Inside the FEA there exists three main phases along the process, and the user needs to go under these in the sequence presented hereafter. These phases are:

1. Pre-processing phase
2. Solution phase
3. Post-processing phase

This pre-processing carries some operations regarding data input. First thing is to define which are the element types that will be used in the model. This selection will be made in terms of the nature of the problem, since the objective will be always to optimize the accuracy reducing the computational cost as much as possible. In order to understand this relevance of cost optimization, a previous research is shown where different element types are shown to analyze the same problem.

Afterwards, once the element types are selected, the information regarding the material used is introduced in the software. When talking about material information, there exist a wide brand of possibilities. Starting with mechanical properties, we can find several related behaviors. Starting with the elastic region, this one can follow different trends. The most common one is the linear elastic (governed by the Hooke's law), however, there exists more options like hyperelastic or viscoelastic materials, which will be defined later in this chapter.

Another mechanical behavior which is very important is the plastic one. As it happens with the elastic behavior, there exist simple models to describe plasticity and the complexity can be incremented up to levels where this plasticity is dependent on the material temperature to introduce this effect inside material deformation. In addition, the user can introduce properties not mechanically related such as electric or heat conductivity. Nonetheless, the model covered in this project will only have mass and elastic properties, so in principle, no damage tolerance nor plastic deformation will be covered under these assumptions.

Once the materials have been defined in the project, these need to be assigned to the different elements conforming the model, which have been previously created. Next, the model itself is constructed, and by

model construction we are not specifically talking about the geometry generation, but also the interaction definition and the introduction of both external forces and boundary conditions. Like it happened with the properties of the materials, there exists many different options available to define the behavior of all the contacts occurring in the system. As well, the forces can be of many types.

In the case of this project, the interactions will be contacts which, in turn, will have a non-penetration condition as well as a friction coefficient to cover both normal and tangential interactions. In terms of forces, three pretensions are required (one for each of the rivets) and a traction force is introduced to pull from one of the plates, in our case the lower metal plate. Lastly, a clamped boundary condition is introduced to fix both the displacement and the rotation in the closest section to the wall of the upper plate.

Lastly, the ultimate step in the pre-processing phase consists on meshing the model and solving some problematic issues that may appear due to this operation, such as initial overclosures, which are explained later on in this chapter. In order to evaluate the mesh quality there exists some parameters which may be useful and these ones will also be mentioned at the end of the first section of the chapter.

Having mentioned all the steps covering the pre-processing phase the solution phase consists in solving the problem itself. When it comes to solvers, there exists two main families which are the implicit and the explicit solvers. The objective of this second section of the chapter is to understand the main differences between them and hence identify which solver is more convenient to be used for the specific case of this project.

The implicit solver is focused on static problems, so this solver will try to find an equilibrium where net force balance is equal to zero. This therefore implies that dynamic or inertia effects are not relevant in this kind of calculation, whereas for the explicit solver, these last play a huge role, since the way it works is by solving the acceleration of each of the elements. Once this one is known, by means of integration the displacement is found. The fact that this second solver solves directly the forces and accelerations, makes the solver more robust and able to handle deviations and non-linearities like the ones mentioned in this chapter. One of the mentioned non-linearities will appear in the model and this one is the existence of a follower force.

On the other hand, implicit solver needs to solve equilibrium equations, which are expressed in terms of matrices to be able to operate with the stiffness matrix, which will be the one which defines the complexity of the problem. When this matrix is singular, because of the procedure of this solver (which requires to invert the stiffness matrix) it makes more unstable and prone to failure compared to the explicit solver.

Considering the static nature of this problems, where there exists small displacements and no mass effects appear in the model, as it can be seen in figure 2.15, the most convenient solver will be the implicit one. However, the reasoning behind this selection will be explained more in depth along the second section of the chapter.

Lastly, two FEM calculations are shown as examples where each of the one solvers are used to obtain the desired results. The aim is for the user to identify the parameters mentioned in section 2 of this chapter so those can be related with the model described in each of the cases. This provides a more practical background about where to use these two solvers. Considering all the information presented in these first two chapters, the third chapter will start developing and constructing the project itself using as background this last mentioned information.

2.1. FEM analysis: Definition and characteristics

2.1.1. Introduction to numerical methods

In order to accomplish the objectives defined for this project, there may exist several approaches, going from simplified calculations up to highly sophisticated methods. In this particular case, all the analysis and model development will be conducted using Abaqus CAE, a finite element method program.

But, what's a finite element method? Why using this specific one among other numerical methods? When selecting the tool used to solve a certain problem, it does not exist a method which will be always the best option. The choice will be influenced by the nature of the problem, its complexity and the resources available. For instance, in order to solve the clamped beam problem, of course it can be used a numerical method to find its solution and this one probably will be very accurate. However, if the beam is rectangular and its weight distribution is somehow very homogeneous in the way certain moments of inertia can be easily calculated, maybe using one of the classical calculation approaches is more than enough to find a reasonably accurate solution, being this last option cheaper computationally speaking.

Nonetheless, when facing more complex problems, the models cannot be solved analytically, and in these instances is where numerical methods play a huge role. Numerical methods are able to solve models whose

behavior can be expressed in terms of ODEs or PDEs. Among the this family it can be found three main branches which are:

- Finite Difference Method (FDM): Calculations are approximations to PDEs. Limited to rectangular sections with small variations. Larger errors compared to other numerical methods.
- Finite Element Method (FEM): Calculations approximate to its solution, and thus they are long but with smaller errors. Complex geometries and boundaries.
- Boundary Element Method (BEM): Attempts to use given boundary conditions to fit boundary values rather than values throughout the space defined by a PDE. Boundary surface is modelled by surface elements instead of the continuum

Understanding how these methods work is key to identify its main differences. Lets start with the first two methods. Given a certain problem, the equations which describe the model behavior are the same, independently of the way solution is found. Therefore, the difference between these two is the road to find the solution. FDM uses differential solution techniques, whereas FEM uses variational ones [9]. Both approximations work well and if treating a certain problem using the two different approaches the result ideally would be the same. However, the errors come from the discretization step where the problem is reduced to a limited number of degrees of freedom.

Both methods are valid since these two underlie under Hamilton's principle, which in static problems (which is the one treated in this thesis) is reduced to the *principle of minimum potential energy* [9]. This principle states that a function that minimizes the potential energy expression of a model is a solution to the equilibrium differential equations. Therefore, we have two options, either solving directly the differential equations, which would be the differential solution, or minimizing the potential energy function of the system which would be the variational solution. As mentioned before, these two methods find approximate solutions due to the discretization made, but this discretization is different for each of the two calculation methods.

For FDM, the differential equations are directly solved. To do so, the algorithm picks a point to be calculated and approximate the value of its derivative at that point. That value is thus introduced in the differential equation to make the calculations. This step is repeated for various number of points until most of the domain is covered and calculation is completed [9]. For instance in the problem shown in figure 2.1, easy to understand in one dimension, the derivative is approximated to a slope covering a certain length. Thus once these values are substituted in the differential equation, this transforms to an algebraic equation. Repeating this operation picking point along the entire line would cover the problem in its completeness.

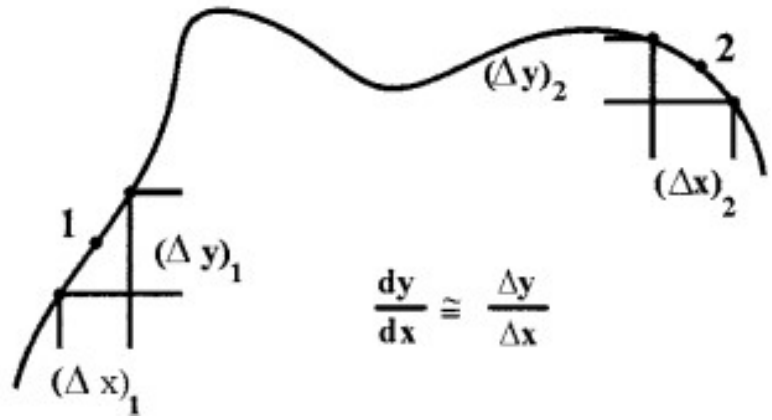


Figure 2.1: FDM resolution example [9]

Nevertheless, the discretization in FEM models is different. Since its objective is to minimize the entire potential energy function, the method will divide the model into small parts of mass, with each if them having its own potential energy function. Therefore, the overall function will be the contribution of all the parts. Lastly, once the contribution is calculated the objective is to minimize it in order to find its minimum and reach convergence. Figure 2.2 explains the procedure in a schematic way.

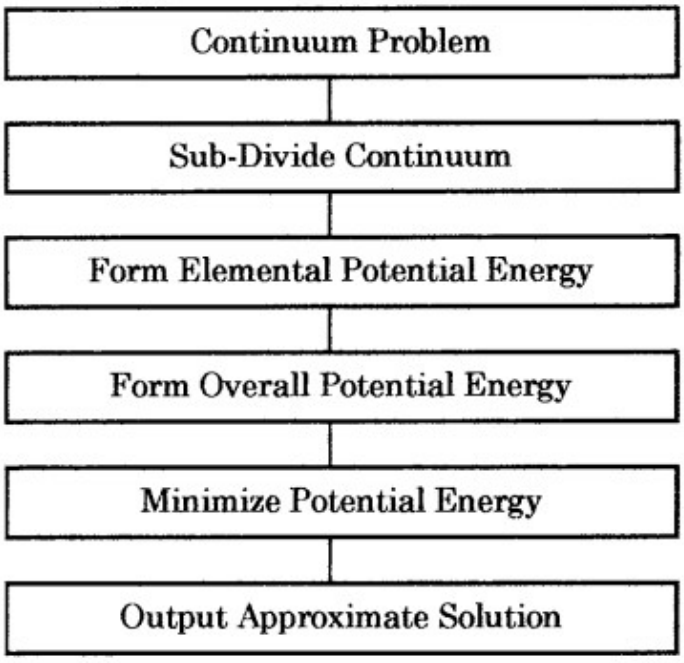


Figure 2.2: FEM resolution mechanism [9]

Lastly, the BEM works similarly in the discretization compared to the FEM. In FEM the entire domain needs to be partitioned, whereas in BEM only the boundaries are considered [10]. This reduces therefore one order of magnitude the equations to be handled and depending on the applications this method is more suitable than FEM.

Specifically, a lot of electromagnetic and acoustic problems are currently solved using this method. Inside BEM we can find the direct BEM (DBEM) where all the physical variables are considered in one side of the surface and thus this last one need to be closed, and indirect BEM (IBEM) that considers both sides of the surface and therefore can solve an open surface [10]. Using BEM boundary conditions coupled with Helmholtz equations is a better option to solve open acoustic problem rather than using FEM [10].

2.1.2. FEM analysis: phases and considerations

Nonetheless, the most accepted solution to solve complex models is the finite element methods, and there exist multiple software which work with this methodology. When creating a model using this tool, there are common multiple phases to go through which are shown in figure 2.3 and listed below [11]:

1. Pre-processing phase: Definition of the model itself, which includes the geometry, load and boundary condition definition among other items.

2. Solution phase: Solving the problem once model has been built up.

3. Post-processing phase: Data output from the solution calculated in the previous phase.

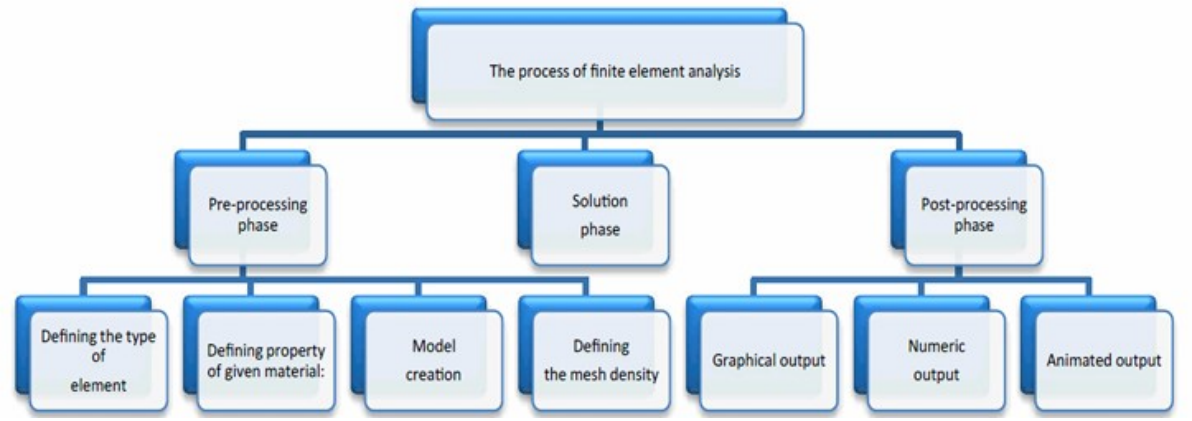


Figure 2.3: FEM model phases [11]

There exists several element types. The utilization of each of them will be determined by the object aimed to be modelled, as well as by the type of analysis that is going to be performed.

Type	Actual Models	Finite Element Expressions (Geometric Properties Defined by Nodes)	Additional Requirements(Actual Volume Calculation)
1D	Rod (Truss) Beam	Length (L)	Area (A, cross-sectional shape) → V = L×A
2D	Shell, Plane Stress, Plane Strain, Axisymmetric, etc	Area (A)	Thickness (t) → V = A×t
3D	Solid	volume (V)	None (volume calculation possible)
Misc.	Spring, Mass, Rigid Link, etc.	-	

Figure 2.4: Examples of element types [11]

Despite most of the problems can be constructed using all of the tree options mentioned in figure 2.4, it is very important to select the most appropriate one for each case. In order to illustrate better this aspect, we can refer to the article *"Modelling optimization involving different types of elements in finite element analysis"* by C.M. Wai et al [12]. In this work it is performed an analysis of a beam subjected to a load using three different types of elements, corresponding to the three options mentioned before. The beam was I-shaped and all the parameters regarding its geometry and material were defined in the document. It turns out that after performing the calculations, the model made with 1D elements was the one with less deviations compared to the theoretical value, as seen in figure 2.5 and 2.6.

Results	Theory	FE Model		
		1D	2D	3D
Maximum Tensile Stress (MPa)	32.251	32.300	33.950	26.650
Maximum Compressive Stress (MPa)	245.881	246.000	252.500	247.500
Maximum Beam Deflection (mm)	38.132	38.100	38.400	38.700

Figure 2.5: Stress results of the experiment [12]

Length of Beam, x (mm)	Deflection of Beam, y (mm)			
	Theory	1D	2D	3D
500	0.4648	0.4650	0.3520	0.3650
1000	1.6125	1.6100	1.4900	1.5100
1500	3.5159	3.5200	3.3800	3.4100
2000	6.1790	6.1800	6.0400	6.0700
2500	9.6027	9.6000	9.4600	9.4900
3000	13.7869	13.8000	13.6000	13.7000
3500	18.7319	18.7000	18.6000	18.6000
4000	24.4376	24.4000	24.3000	24.3000
4500	30.9040	30.9000	30.7000	30.8000
5000	38.1311	38.1000	38.1000	38.0000

Figure 2.6: Deformation results of the experiment [12]

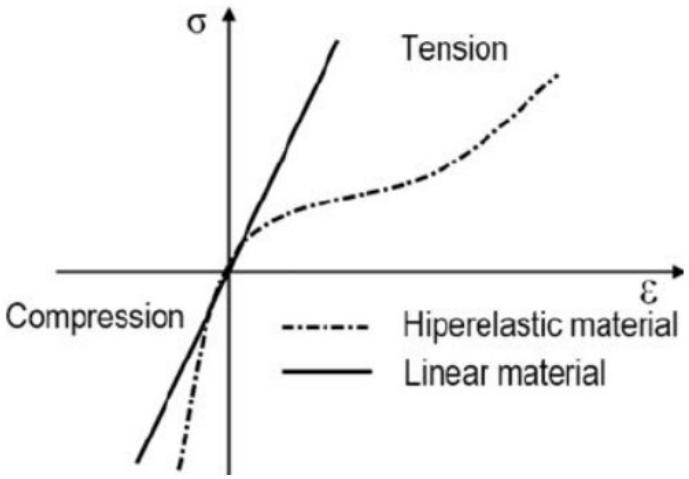
Moreover, the results for the beam made out of 1D elements were easier to understand since the axial stresses and the bending displacements could be extracted directly. Nonetheless, for the 2D and 3D element beams, it required some deep understanding and extra work since the data exits in the form of tensors. Therefore, for that kind of problems where forces and displacements can be kept in one plane of analysis it is convenient to use 1D elements because despite being simpler to create and understand, they can obtain very accurate results. Of course a beam can be modelled using 2D and 3D elements, but it is not the most appropriate option. In fact, Nastran has, among others, elements called *CBAR* and *CBEAM* which both are by definition 1D elements.

Jumping to the material properties, Abaqus allows to add a wide range of parameters when creating a material or introducing it as an input. There exist general variables as the density of the material, as well as mechanical and thermal variables, among others. Starting with the mechanical properties, the most basic one is the elastic properties. Inside this specific field, the user can find different types of behaviours. The most common ones are [13]:

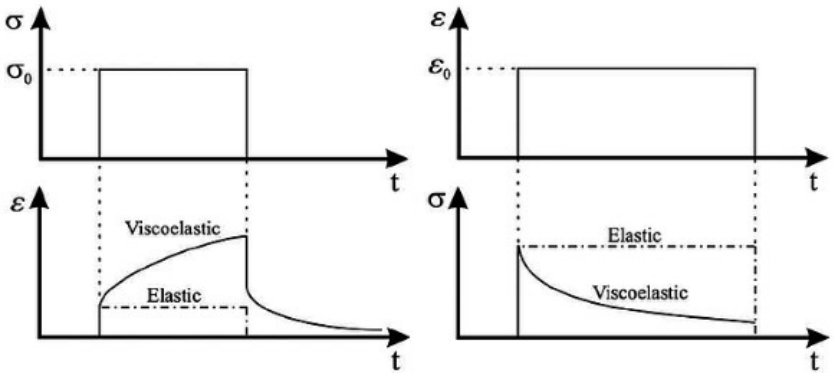
- Elastic materials: The materials that have this property will obey the Hooke's law. The deformation will have a linear relation with the stress applied determined by the Young modulus and the Poisson ratio. As soon as the stress is released, the material will come back to its original shape.

- Hyperelastic materials: This material will also recover its shape once the load disappears. However, unlike Hookean elastic materials the relation between stress and strain is not linear anymore.

- Viscoelastic materials: The relation between the stress and strain has a time dependency since strain rate affects the material behaviour. Figure 2.7(b) show the comparison between viscoelastic and elastic behaviour for two situations. In addition, when removing the stresses on the material, the unloading path does not coincide with the loading path.



(a) Elastic vs hyperelastic material



(b) Elastic vs viscoelastic material

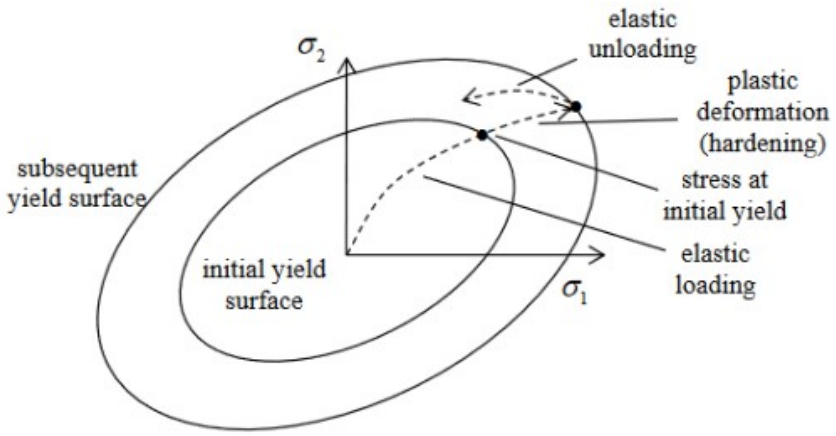
Figure 2.7: Possible material behaviours regarding elasticity [13]

Apart from these ones, there exist more options inside this field, such as porous elastic or hypoelastic materials.

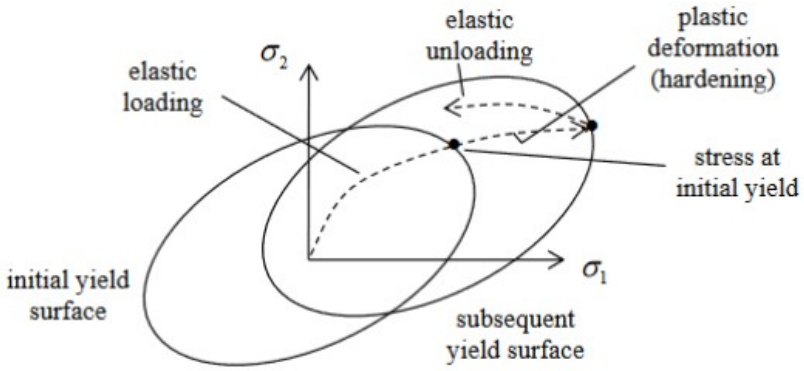
Next mechanical property that can be added is the plasticity. This attribute is very important because it applies to the regions of permanent deformation. Reaching this region is interesting since, if done strategically, a plastic deformation will make the material more resistant to further deformation and thus it will withstand higher stresses. This action of applying on purpose a stress above the yielding in order to make the material stronger is called hardening. For general plastic behaviour, the more important types of hardening are [14]:

- Isotropic: In isotropic hardening, once the applied stress surpass the yield stress, the yielding surface is expanded uniformly in the all the three axes of the body, making the material thus stronger in all directions no matter what is the stress vector applied as long as its magnitude is higher than the yielding

- Kinematic: Unlike it happened with isotropic, with kinematic hardening the yielding surface does not increase its size, but instead moves depending on the direction of the applied load. Therefore, by definition, this is one type of an anisotropic hardening.



(a) Isotropic hardening



(b) Kinematic hardening

Figure 2.8: Material hardening [14]

- Combination: Combining the two previous effects is possible to define different types of anisotropic behaviours, by both increasing and translating the yielding surface as shown in figure 2.8.

- Johnson-Cook: A more detailed model which takes into account not only mechanical strain hardening but also material softening due to the effect of temperature. This model follows this equation shown below, where variables such as melting temperature, reference temperature and strain rate appear. This is the most appropriate option to fully analyze problems with temperature influence [15].

$$\sigma_{eq} = \left(A + B\epsilon_{eq}^{pl\,n}\right) \cdot \left(1 + C ln \frac{\dot{\epsilon}}{\dot{\epsilon}_0}\right) \cdot \left(1 - \frac{T - T_0}{T_{melt} - T_0}^{m}\right) \tag{2.1}$$

Apart from elasticity and plasticity, current programs allow to introduce damage tolerance models which can indicate when a material will fail under the defined circumstances. There exist multiple models to simulate this phenomena, but once again, Johnson-Cook developed a formula to calculate the failure strain in terms of 5 parameters called D_1, D_2, D_3, D_4 and D_5 [16].

$$\epsilon_f = \left(D_1 + D_2 \cdot e^{D_3 \cdot \sigma^*}\right) \cdot \left(1 + D_4 \cdot ln\dot{\epsilon}^*\right) \cdot \left(1 + D_5 T^*\right) \tag{2.2}$$

where

$$\dot{\epsilon}^* = \frac{\dot{\epsilon}_{eq}}{\dot{\epsilon}_0} \tag{2.3}$$

and

$$T^* = \frac{T - T_{room}}{T_{melt} - T_{room}} \tag{2.4}$$

Moreover, the material can be assigned more mechanical characteristics such as viscosity, brittle cracking information or expansion coefficient. Of course, most of these last mentioned properties can be introduced as uniform or temperature dependent. The user can introduce a chart with several values covering a range of temperatures so when the software starts the calculation it can extrapolate using the data provided in this section.

Considering the problem that is being faced in this thesis, only mechanical properties are going to be added since they are the only ones that play a significant role in this project. Nonetheless, if the temperature has an impact in the problem, some variables such as thermal conductivity, latent heat or Joule heat fraction can be also introduced inside the material information. Same happens with electric or magnetic parameters, but as mentioned these last will not be considered in the matter of this document since they are completely irrelevant considering the project purposes.

Once the materials have been defined in the project file, these are introduced in the model by means of sections. These sections indicate how the material is distributed along the domain the user is referring to, and how this one will behave. There are plenty of options regarding this aspect, like solids or shells. Each single part included in the model needs to have a section assigned previous to the calculation phase.

Third step inside pre-processing phase consists in the model creation, which as previously mentioned, involve not only the geometry creation of each of the parts, but also defining the interactions between the elements of the model, the external loads applied and the boundary conditions which govern the problem. Furthermore, the time steps used during the calculation need to be specified in this phase.

Starting with the interactions definition, there exist plenty of categories. However, for most of the mechanical problems, the only interaction type that will appear will be contacts. Firstly, because initially the model can have some parts contacting each other before any step is initialized and secondly, because of the relative motion of the parts during the analysis, some parts may collide between them.

When these collisions occur, because of Signorini nonpenetration conditions, the user needs to define two items [11]:

- Master or primary surface: Surface responsible of generating the collision, which indeed induces contact forces and reactions. Contact behaviour is defined and controlled by this surface.

- Slave or secondary surface: It suffers the contact and thus responds to this collision adapting its shape and generating reactions.

As an example, a model can be represented in figure 2.9. The purple body is moving and it is close to collide with the blue body. Since the purple body is the one which initiates the contact, its nodes will conform the master surface and consequently, the external nodes of the blue body will behave as slaves, adapting its motion to the upcoming collision once it occurs.

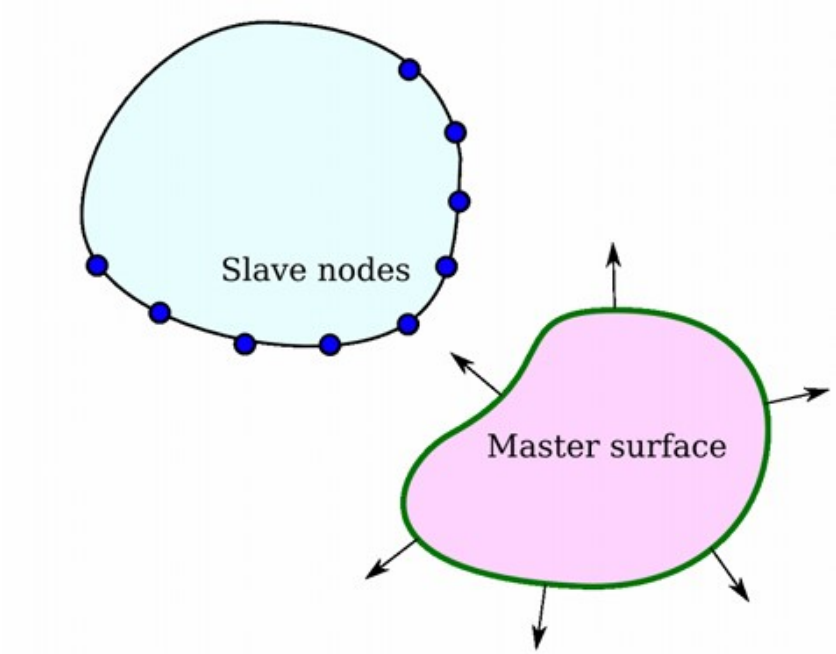


Figure 2.9: Master and slave surfaces description [11]

Depending on the problem and its geometry, there exist a wide variety of contact types. Some of them are [17]:

- Surface to surface
- Surface to edge
- Edge to edge
- Vertex to surface

Moreover, every contact interaction need to be defined, introducing parameters regarding its tangential and normal behaviour (frictionless or not), as well as other non-mechanical properties such as heat generation or electric conductivity if applicable.

Diving more in depth into Abaqus software, there exist two main types of contact assignation:

- General contact: Is the simplest way to introduce the contact interaction. It allows an efficient and automated contact definition ensuring conditions between the surfaces are enforced efficiently. It has very few restrictions and using a single definition all the multibody contacts can be defined.

  The greater advantage of this choice is that it allows all the surfaces of the model contact between themselves without having to specify which contact pairs can collide between them.

- Contact pairs: The user assigns an specific interaction to two surfaces which may come to contact during the analysis. Inside this option it is required to define which will be the master surface and it allows to adjust the clearance between the surfaces in case needed.

Despite defining properly the interactions in its corresponding module, when running the model the user needs to be aware that the following situations may lead to failure during calculation [17]:

- Initial overclosure: This occurs when the two surfaces are overlapped before calculation starts. This can occur because of an error made when creating the assembly or poor meshing when gaps and tolerances are tight. To solve this issue there exist two ways.

  The first solution is to use strain free adjustment, where the slave nodes will reconfigure themselves reducing the overlap until this one is totally removed. This motion or correction does not carry stress to the model, so this solution is the most appropriate one when initially the parts should not be in contact.

The second solution is the utilization of interference fits. Unlike previous option which was performed before calculation starts, this choice removes the overlap in the early steps, creating its corresponding stresses in the surfaces due to the tension generated by that movement.

- Initial gap: Finding gaps between the contact surfaces before starting the simulation can lead to convergence issues. This happens since there are no restrictions between the parts, and this could cause extremely large displacements reaching hence the failure. Therefore, it is important to adjust this clearance to the minimum possible. In case the model has by definition big gaps between the surfaces, contact stabilization can be used in order to reach model convergence.

- Mesh quality: If the seeds are not small enough or the curvature of the two contact surfaces is very different between them, some nodes of the master surface can penetrate the slave surface before simulation even started, as shown in figure 2.10, inducing unrealistic forces and leading to errors in the calculations. Some of the tips to solve this problem is to use contact to contact interaction when knowing in advance the contact pairs that will collide, as well as a refinement in the slave body mesh in order to increase smoothness in the contact region.

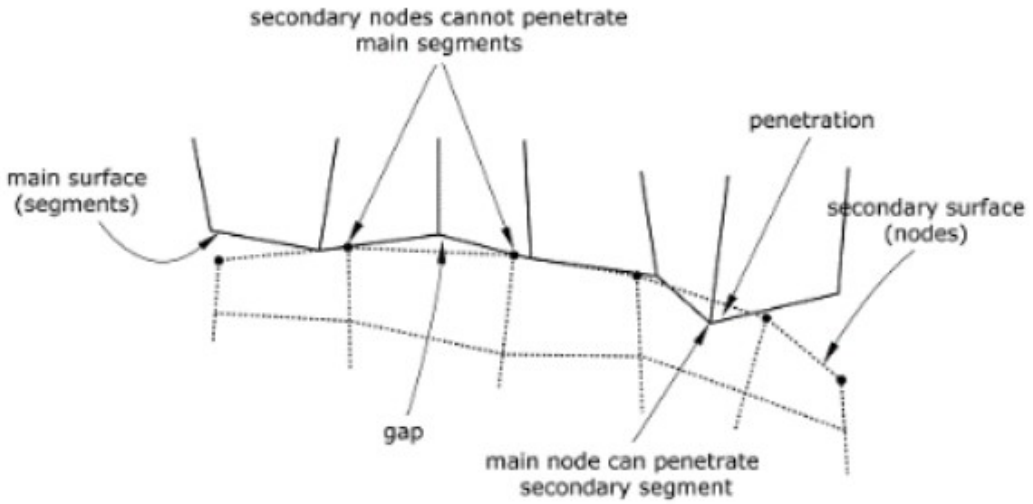


Figure 2.10: Errors caused by poor mesh quality [17]

Once interactions have been properly introduced, doing so with the loads and boundary conditions is trivial.

However, the load definition may be straightforward when the element where force is applied has been already defined (this may be a point, an edge or a surface). Nevertheless, when the force is applied at a point which has not been defined in the model construction, it is required an intermediate step.

Of course, it would be required to locate this point and identify it inside the project, although this would not be enough. In order to connect this point with model behaviour, it is precise to make a coupling between these two. By doing so, the point will be considered as a part of the assembly, allowing to introduce forces in it.

Lastly, when all the assembly is finished, with its interactions defined, and the boundary conditions and external loads have been specified, it is time to discretize the domain of analysis. This last step is crucial in order to achieve both accurate results and model convergence.

As it happened with the element types, the mesh seeds can have many different shapes, like the ones shown in figure 2.11, and depending on the overall geometry there will be a seed which will make an optimum balance between accuracy and computational cost. In the case of this project, given the nature of the model, only 3D seeds can be used. Among all of them, because all of the members which conform the assembly are straight extrusions, the best option is to choose an hexahedron seed to create the mesh. This allows the program to create smoother boundaries which allows better calculations in the interaction zone.

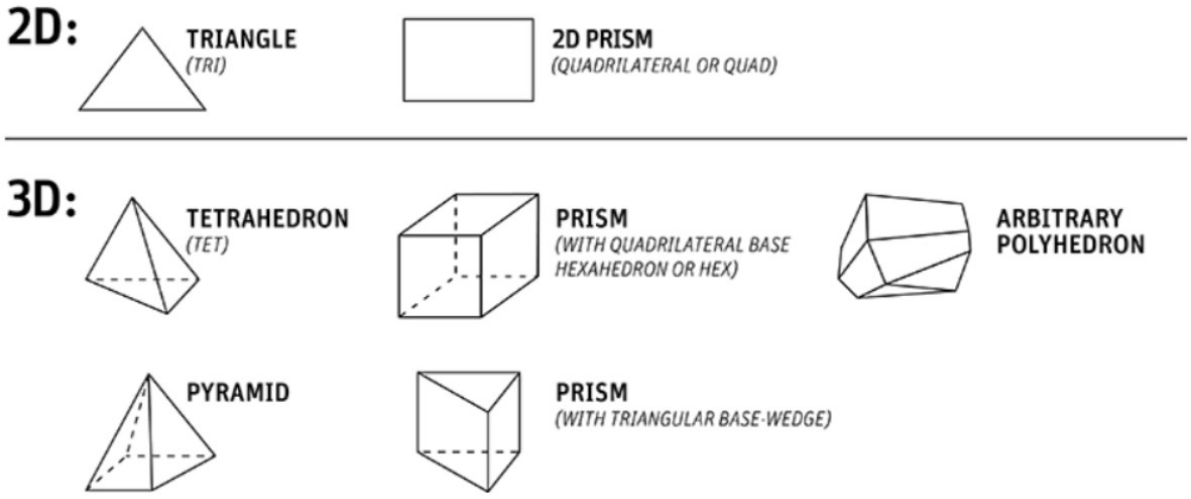


Figure 2.11: Examples of mesh elements

Moreover, given a certain seed, there may exist infinite possibilities. For instance, a prism seed can be thin and tall, or on the other hand small and wider. Thus, despite selecting the correct mesh configuration, there exist certain parameters which allow the user to evaluate the quality of the mesh, and these are [11]:

- Aspect ratio: As a general rule, it should not be higher than 4:1. However, depending on the region and the model, this upper bound limit can change.
- Skewness: Quadrilateral elements should be kept as squared as possible.
- Taper check: Similar to the previous one, the objective is to keep this taper ratio the closer as it can be to the unity.
- Warp: The less distorted the mesh is, the better quality it will have. Usually up to 5% of warp is acceptable.

If these aspects are taking carefully into consideration the model will be able to run and obtain a good result with the discretization made.

Next step would involve the numerical calculations which are explained more in detail in the next section. Depending on the nature of the system, the problem will be solved differently and thus the calculation procedure will be consequently adjusted.

Lastly, once the calculation have been completed, the post-processing phase starts. By default, since the output properties will be related to the nodes where they have been calculated, the system shows contours since they are visual and intuitive when used to compare magnitudes. These graphical outputs can be 2D or 3D depending on the domain asked to extract the desired values. Usually they are represented with colours along with a legend which states the value corresponding to certain colours, as shown in figure A.14, located in Appendix A. Alternatively, if requested, the results can be exported into files where exact numerical results are expressed for each of the nodes specified. This can be useful if the user wants to perform an outside operation with the data extracted such as field integrations.

It has to be mentioned that only the values requested by the user during the pre-process phase will be the ones available to show in this phase. Therefore, it is very important to not forget defining the desired outputs in each of the steps created.

In addition, the software also represents the motion of the pieces during the experiment creating an animation. This video can be manipulated by making cuts, controlling its speed or selecting the domain which is intended to be shown. Furthermore, it allows to scale the results in order to make appreciable the smaller changes, which with all the mentioned possibilities, makes this software very powerful to both calculate and analyze complex models and experiments.

2.2. Solution phase: Two different solvers

After finishing all the steps corresponding to the pre-processing phase mentioned in the previous chapter, the model would be set for calculation.

During the resolution process, there are certain elements which are key to solve the model. Precisely, when describing the resolution process inside FEA, the general equation which defines the system can be summed up in the following one [11]:

$$[K] \cdot \{\vec{u}\} = \{\vec{F}\} \tag{2.5}$$

where:

- $[K]$ is the global stiffness matrix
- $\{\vec{u}\}$ is the displacement vector
- $\{\vec{F}\}$ is the vector of external nodal forces

The displacement vector is an unknown and it is the target of the solver for each of the time steps. The vector of external forces is already known from the problem statement and the conditions provided. Therefore, since the displacement vector is the target solution and the forces vector is external to the system, all information about it will be contained in the global stiffness matrix. This last element is one of the most important items to consider when solving a problem. This matrix describes the behaviour of the system when subjected to external excitations and, as it will be seen later on, its shape and content will be key to determine the robustness and the complexity of the problem and its resolution.

This matrix in constructed by introducing each of the equations that govern the system in matrix form, so all the system information can be kept inside. Let's consider for instance the following problem, represented in figure 2.12

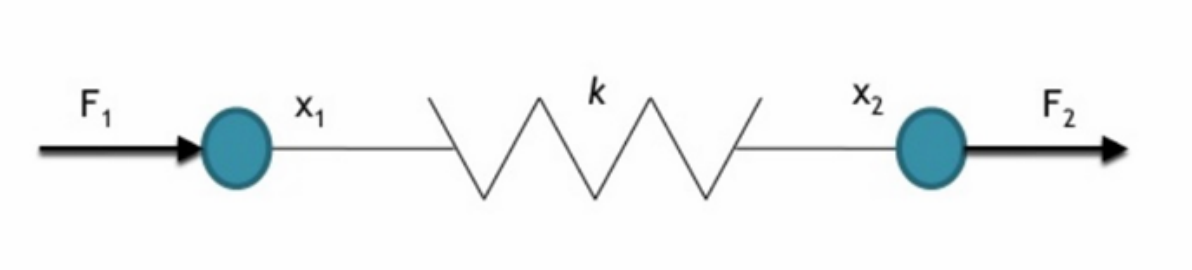


Figure 2.12: Sample system [11]

We have two elements joint by the spring with stiffness k. Recalling the one dimension Hooke's law, the relation between the forces and the displacement for a spring is the following one:

$$F = -k \cdot \Delta x \tag{2.6}$$

where:

- F is the force applied on the element
- k is the spring stiffness
- Δx is the increment in length of the spring

For each one of the two elements, we can derive an equation. Starting with the left ball of the figure, the equation that governs its motion will be:

$$F_1 = k \cdot (x_1 - x_2) \tag{2.7}$$

However for the element located on the right side, we can derive the equation hereafter:

$$F_2 = k \cdot (x_2 - x_1) \tag{2.8}$$

Of course, a method to solve this system could be to find x_2 in terms of x_1 (or the other way around) from one equation and substitute in the second equation to find the unknowns. However, these two equations can be represented in a matrix form. To do so, we can recall equation 2.5 and express all the items in the following way:

- $\{\vec{F}\} = \{F_1, F_2\}$
- $\{\vec{u}\} = \{x_1, x_2\}$

Thus, we could rewrite our previous equation into this matrix form:

$$\begin{bmatrix} k & -k \\ -k & k \end{bmatrix} \cdot \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix} \tag{2.9}$$

By comparison with equation 2.5 it can be seen that the stiffness matrix is the same as the one present in equation 2.9. As shown, the way to obtain the stiffness matrix of the system is to propose the equation which governs the motion of each one of the elements and by joining them all we can find this matrix. It can be realized that the dimension of the matrix coincides with the degrees of freedom of the system, in this case two.

If this sample system is changed to the one shown in figure 2.13, as one can see trivially, there are three degrees of freedom, corresponding each of them with the x-coordinate of the three elements conforming the model.

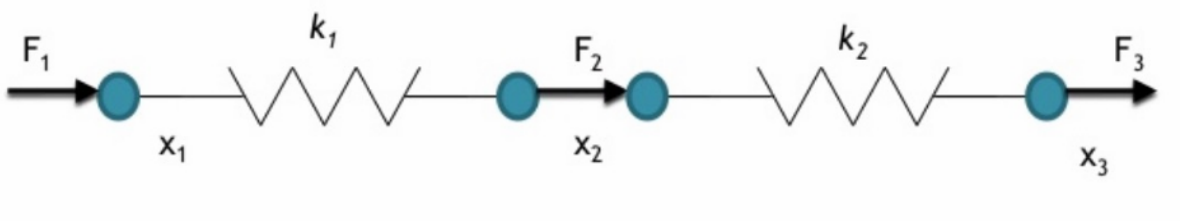


Figure 2.13: Sample system extended [11]

For the previous problem we could derive two equations for each of the strings. Likewise, for this case we will find four equations since there are two strings. For the first string we can find the following equations:

$$F_1 = k_1 \cdot (x_1 - x_2) \tag{2.10}$$

$$F_2 = k_1 \cdot (x_2 - x_1) \tag{2.11}$$

And from the second string, the following equations come up:

$$F_2 = k_2 \cdot (x_2 - x_3) \tag{2.12}$$

$$F_3 = k_2 \cdot (x_3 - x_2) \tag{2.13}$$

which can be rewritten in matrix form as follows:

$$\begin{bmatrix} k_1 & -k_1 \\ -k_1 & k_1 \end{bmatrix} \cdot \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix} \tag{2.14}$$

$$\begin{bmatrix} k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \cdot \begin{Bmatrix} x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} F_2 \\ F_3 \end{Bmatrix} \tag{2.15}$$

From here we can see the two stiffness matrix of each subsystem, which describes the behaviour inside that interaction.

$$[K]_1 = \begin{bmatrix} k_1 & -k_1 \\ -k_1 & k_1 \end{bmatrix} \tag{2.16}$$

$$[K]_2 = \begin{bmatrix} k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \tag{2.17}$$

In order to get the global stiffness matrix, it is required to apply the continuity boundary condition between both subsystems in order to get the final stiffness matrix which is:

$$[K] = \begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & (k_1+k_2) & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \tag{2.18}$$

So the system can be represented with the following equation:

$$\begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & (k_1+k_2) & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \cdot \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \end{Bmatrix} \tag{2.19}$$

And once again, since there are three degrees of freedom in the system, the rank of the stiffness matrix will be also three. As the problem gets more complex, this matrix gets greater and its rank increases with the degrees of freedom of the overall system. Once this matrix is found, the solving procedure can start.

However, depending on the nature of the problem, it can be found two different types of solver: explicit and implicit. But, what are the differences between them? Let's go through them one by one [18]:

- **Implicit solver:** Used in static analysis, when neither mass nor damping have a considered impact in model behaviour. Departing from last calculated stage, it solves a set of equation which defines the problem in order to arrive to the solution at the next stage. Therefore, these calculations require a several number of attempts for the system to solve each of the time increments. The way of solving the system of equation always require to invert the stiffness matrix, and when this one is singular, it leads to convergence problems.

  However, it does not exist an upper limit for time steps which concerns the convergence. Since this solver is not used for dynamic problems, Courant number does not play any role and therefore, this explains why the time steps in implicit solvers are relatively larger than the ones existing for explicit.

- **Explicit solver:** Used for dynamic analysis, when nodal forces related to mass and included in the calculation. Since mass forces are considered, the solver directly proceeds to calculate the nodal accelerations. The main advantage of this aspect is that, since accelerations are calculated directly, there is no need to invert the stiffness matrix, and so it does not requires multiple iterations per time step and it is able to handle better the nonlinearities. Once accelerations are calculated, integrating it can be found the velocities and displacements. From displacements it can be calculated the strain and, hereafter the corresponding stresses which would close the cycle and this would be repeated for each of the time steps.

  Nonetheless, on the other hand, since dynamic effects are considered, the Courant number which measures somehow the steadiness of the system, will determine the upper bound for the time steps in order to reach convergence.

Both solvers use the data calculated in the previous time step, but the difference between them is how to jump to the next stage of the system. The computational cost of solving an explicit time step is lower than the implicit one because it consists on a direct extrapolation departing from previous state [18]. The fact that implicit solver needs to solve the equilibrium equations governing the system makes it expensive since several iterations need to be carried. Moreover, the iterations and time step along a model calculation can vary depending on the point of the analysis, since the conditions also change while calculation is running.

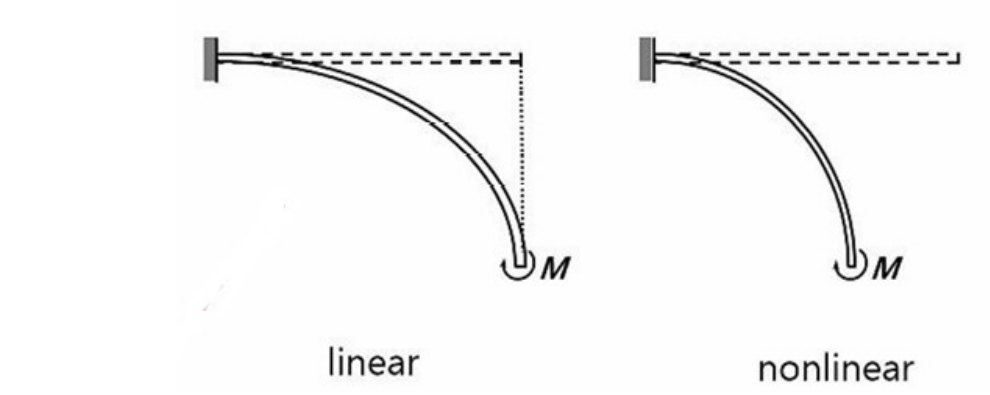
On the other hand, since explicit solver consists on an extrapolation, it makes sense that the distance from departing stage to the next one should not be very large, because that would cause errors to be dragged all over the analysis, allowing them grow exponentially. This problem does not apply to implicit solver since the system of equations already bounds the system. Nevertheless, these limitations cause the implicit solver to handle poorly non-linearities, being the explicit solver more convenient for those cases.

Regarding non-linearities, there exist three main types [11]:

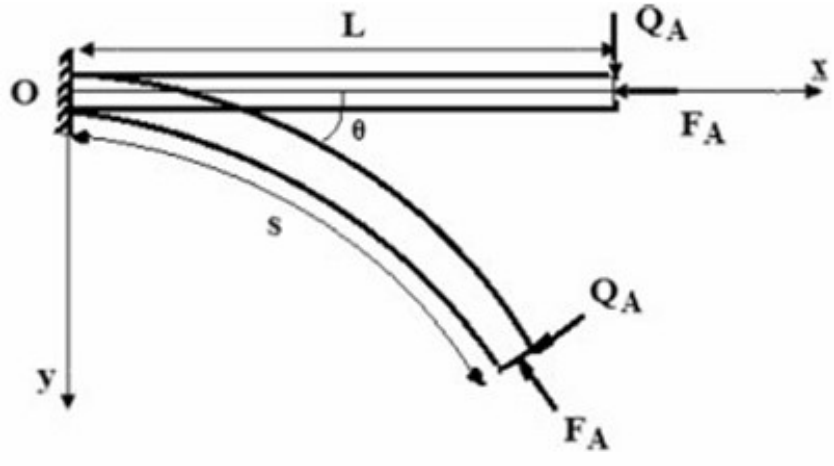
1. Geometrical non-linearities: non linear strain-displacement relation

2. Material non linearity: non linear constitutive relation (hyperelastic or viscoelastic materials)

3. Boundary non linearity: Non constant boundary conditions, contacts or follower forces. These last two are particularly found in the model studied in this thesis.



(a) Geometric nonlinearity



(b) Boundary nonlinearity: following force

Figure 2.14: Types of nonlinearities [11]

Therefore, the following points summarize this first introduction to the different solvers:

1. Implicit solver require a higher computational cost per increment in time compared to explicit solver.

2. Implicit solver allows the user to utilize larger time steps.

3. Time increments are more regular for explicit solver compared to the ones found in implicit analysis.

4. Explicit solver handles better non-linearities.

2.2.1. Differences between both solvers: Dynamic effects

Previously, it has been stated the differences in time step calculation and convergence for both solvers, and they have been derived by understanding how they worked independently. The same approached is performed when explaining the differences between them when describing how easy can the handle the dynamic effects.

In the case these dynamic effects do not exist, there must exist an static equilibrium to which implicit solver will try to arrive. In implicit analysis, inertia is not taken into account which means it cannot be used to stabilize the system [18]. This also means that rigid body motion is forbidden, and this opens the doors to

errors like the one described in previous chapter where a large gap exists. To prevent this, the model must be linked to the ground during an static analysis. As well, internal energy cannot be converted into kinetic energy in static analysis, so the only way to stabilize the system in the presence of dynamic effects is by applying damping. Therefore, implicit solver is directly linked to more steady problems.

Unlike implicit solver, explicit calculations solve the second Newton's law, which explains indeed the relevance of time when using this solver. Because of this time influence, the results will tend to be more oscillatory and the loading rates will have a big impact during the calculation. If it is intended to not consider dynamic effects, the solution is to apply the loads so slow so that the product of mass and body acceleration is almost negligible. Thus, it is not impossible to carry a static analysis using an explicit solver, but it will take longer compared to the case run with the implicit one.

However, this is not usually the case, so it can be stated that dynamic analysis where inertia and mass play a big role are directly linked to an explicit solver. Another aspect which was mentioned in the previous section is that explicit solver handles better the nonlinearities, and one of these nonlinearities are contacts. Changes in these ones along the simulation and new collisions can be solved easier using an explicit solver, and for extreme cases like bullet impact, this may be the only alternative between the two already proposed.

Thus, the following statements sum up the difference between both solvers regarding dynamic effects [18]:

1. Implicit solver is more related to static and steady analysis

2. Its capacity of handling better the different nonlinearities makes the explicit solver more suitable for models where damping or inertia dominates the motion.

2.3. Restrictions and range of applicability

Recalling the already mentioned information, all the different cases can be characterized by the two main following properties [18]:

- Steadiness of the system. Quantity of motion in the overall domain.

- Amount of nonlinearities taking place in the model and their impact in its behaviour

Previous sections have been calling these two aspects when describing the two different solvers. So taking into account each of the strengths for each solver, a map where all the cases are considered can be drawn in order to identify which option is the most suitable depending in the region where the model belongs.

Figure 2.15 represent the regions where each of the solvers is more suitable to be used. The map hence relies on the statements hereafter:

- Implicit solver shall be used for static problems with not many nonlinearities.

- Explicit solver covers a wider range of cases, but it is more efficient when utilized for high nonlinear problems which are dominated by inertia and mass, which are very unsteady and contain a lot of motion.

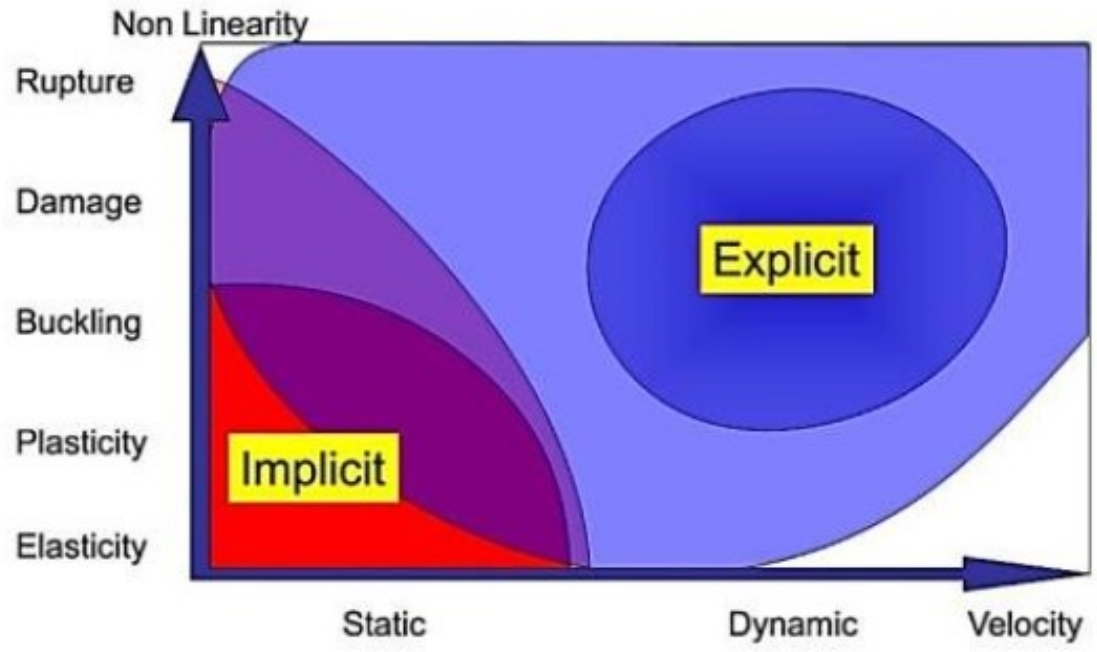


Figure 2.15: Explicit vs Implicit solver [11]

Recalling the target model of this thesis, it will consist in a riveting operation and a traction force which will create a small displacement. In addition, the material will be purely elastic, without rupture nor damage tolerance. Considering these aspects which quickly summarize the nature of the problem, it can be seen that the proper solver to be used is an implicit solver, as the problem to be solved is static. Indeed, when trying to introduce certain loads as the pretension forces in the rivets, it can be seen that these ones can only be introduced if the analysis is static, which consequently implies the use of an implicit solver.

2.3.1. Implicit solver sample case: Static analysis of a USV feeder boat

As an example for a case suitable to be solved by the mentioned solver, it can be observed the work performed by Aknaf Sam Dabit et al. in their paper "*Finite Element Analysis (FEA) on Autonomous Unmanned Surface Vehicle Feeder Boat subjected to Static Loads*" [19]. The aim of the project was to build an unmanned surface vehicle (USV) feeder boat using CAD and afterwards test the model using finite element analysis (FEA).

Once the model was built the test consisted in an static analysis where the load was applied at the hull of the boat. Since the fish can be spread in the boat in different ways, the paper took into account this fact by performing several analysis where the loads where applied at different positions of the hull. Since the objective was to check if the material of the boat did not yield, the analysis was kept from the beginning in the linear region [19].

When geometry and loading conditions were defined, they selected the material, which in this case was balsa wood, and proceed with the meshing selecting an overall seed density using equilateral triangle prisms to shape the seeds, as seen in figure 2.16.

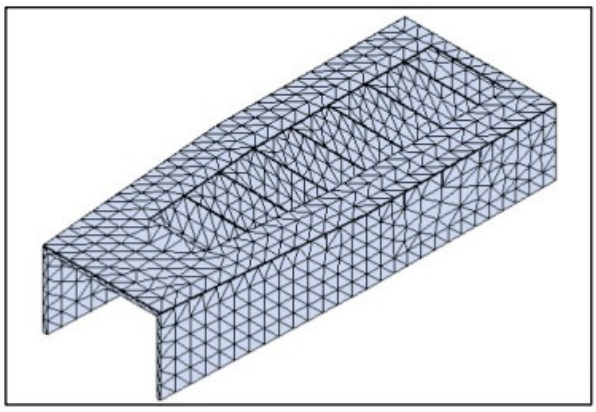
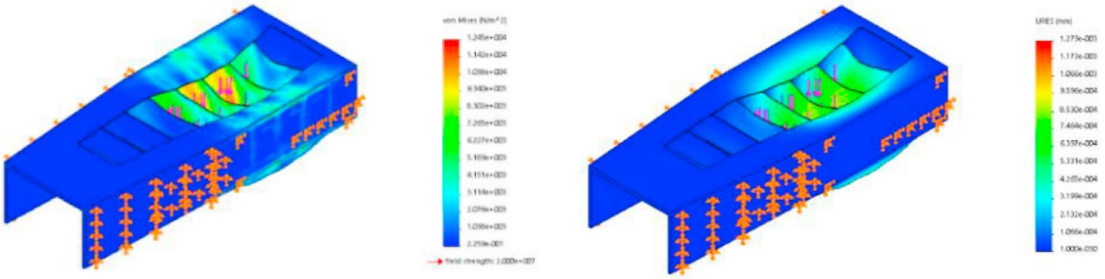


Figure 2.16: Boat mesh [19]

The calculation outputs are shown in figure 2.17, and they were the stress contour, as well as the strain and displacements . These last were small compared to the model dimension, and the maximum stress did not reach the yield value, which leads to the conclusion that the analysis was kept in the linear region of the material, as expected.



(a) Stress results

(b) Displacement results

Figure 2.17: Analysis results [19]

2.3.2. Explicit solver sample case: An impact loading simulation of Rubber-Toughened Poly MethylMethacrylate (RT-PMMA)

The selected example is explained in depth in the paper "*An impact loading simulation of Rubber-Toughened Poly Methyl Methacrylate (RT-PMMA) using Abaqus Explicit*" performed by Tahir et al [20]. Polymethyl methacrylate (PMMA) is an amorphous thermoplastic that is lightweight and transparent, being able to replace glass in certain applications, since it also has high resistance to impact.

Therefore, the overall objective of the project was to test two different types of PMMA which suffer an impact from steel projectiles coming from a gas launcher. This test was conducted in reality and captioned by high speed cameras which captured the critical moments when impact occurred. But as well, an explicit analysis was run in Abaqus in order to check the fidelity of the model created.

To do so, in order to model the material introduced in the software, the Extended Drucker Prager (EDP) model was used to define the inelastic behaviour of the material considering the percentage of rubber contained in each of the samples [20]. Furthermore, the damage initiation criterion and further propagation was also defined as ductile, which meant that the material stiffness was degraded progressively.

The result of combining both the test realization and the model implementation was successful, as they were able to relate the stress distribution inside the PMMA with each of the critical states where impact occur. Figures 2.18, 2.19 and 2.20 show both the captions of the cameras and the stress distribution for the time values of 30μs, 50μs and 70μs, respectively.

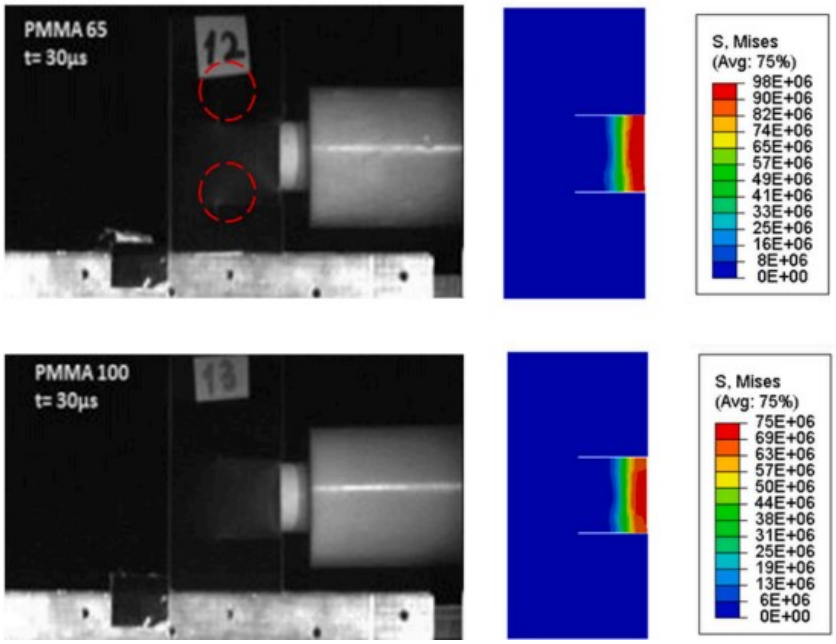


Figure 2.18: Stress distribution at 30μs [20]

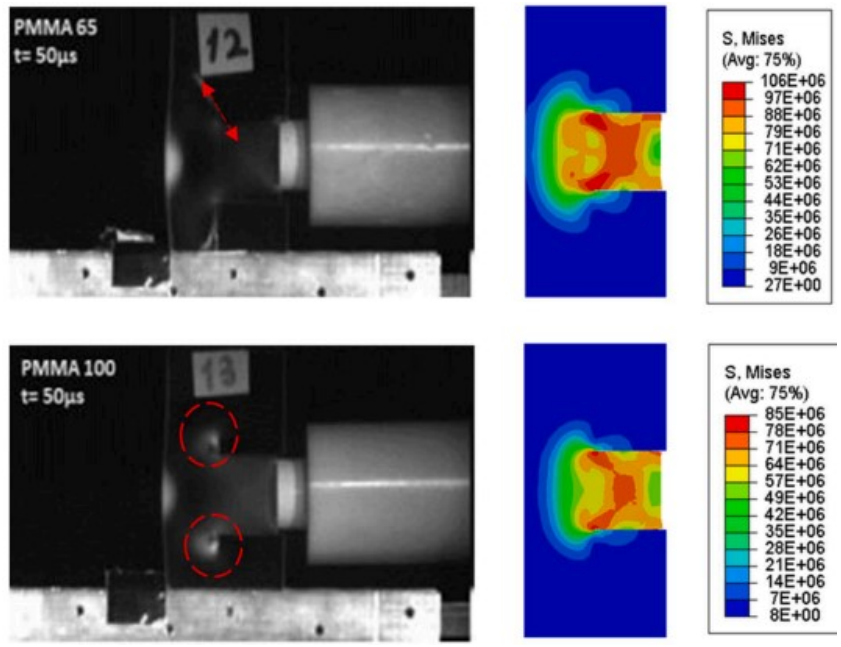


Figure 2.19: Stress distribution at 50μs [20]

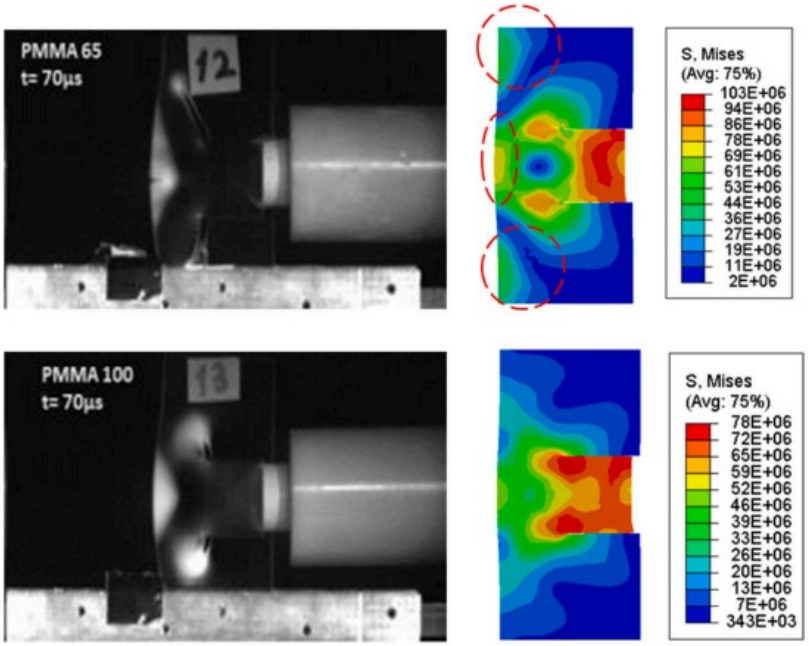


Figure 2.20: Stress distribution at 70μs [20]

With this analysis they could identify the main differences between the two samples tested in their behaviour against an impact of such magnitude. This case shows the relevance of time and motion inside an explicit analysis, where of course the kinetic energy of the projectile was able to generate stress above the yield value entering the plastic zone, which indeed is coherent with the previous definitions inside the material information. Actually, the conclusions of the project states that this impact would lead to crack propagation in the samples tested.

3

Model generation

Having provided a brief mathematical background about the FEA, it was concluded that, in order to study the intended phenomenon, it was required to use an static implicit analysis where the steps are added subsequently in the same order the course of events occur. Having that said,

the first step performed inside this chapter is presenting the analytic solution for single lap shear riveted assemblies. The reason behind this is because, when having finished the FEA, we will be able to compare how close are these two results and thus, determine what are the reasons behind their differences. But once this analytic solution is presented, the next step proceeds to the core of this project, creating hence the first model using Abaqus CAE. The aim of this model was not extracting relevant values, but instead introduce in the program all the variables which afterwards will be parametrized. At the same time, this trial model would be useful to provide us information about how does the assembly works under the proposed conditions.

After finishing this first model, and having verified that the behaviour is the expected one, the next step consist in extracting the RPY file corresponding to that model since, the upcoming progress is performed using Python. That RPY file would work as the basis of the following achievements, since programming allows to create all the dependences relative to the model. Thus, in order to create these dependences, it was crucial to convert the dummy values introduced during the model creation into the design variables aimed to be studied, in such a way the operations regarding that magnitudes are governed by the inputs specified by the user. The chapter contains a list of the design variables and a graphic representation of them, as well as the nomenclature used for each one of them inside the python script.

Secondly, after defining the design variables in the script, it was crucial to introduce them also in the operations which depended on them, since sometimes more than one variable was related to the other. This leads to constrains that need to be implied to make the assembly work correctly. The document covers the specific operations involved more in depth and more figures are shown to support the explanation and the reason behind each of the steps taken towards the model parametrization. Lastly, some adjustments need to be made to the code itself, since some of the operations of the trial model somehow restricted future progress in terms of parametrization. Thus, some specific changes need to be made to make the script as flexible as possible, allowing more study possibilities.

Having this done, the script was able to create a model with the parameters introduced by hand. Since the aim of the project was studying several cases, it was not efficient enough to create the models doing so, so another script was created to generate the model files. This function, if introduced in a loop, it could generate a big amount of files within seconds, saving a great amount of time. Therefore, the only step left was to create a table with rows, corresponding each of them with each of the models, designating each column for one of the design variables. The overall procedure would be summed up in these points:

1. The python file reads the table containing all the information about the models and its design variables
2. The file generator code starts to run, but being inside a loop.
3. In each iteration (corresponding each to the number of rows contained in the table mentioned in step 1) each column is assigned one of the design variables in python.
4. When having these values, the function generates the file for this iteration.

5. Subsequent iterations occur until all model files are created.

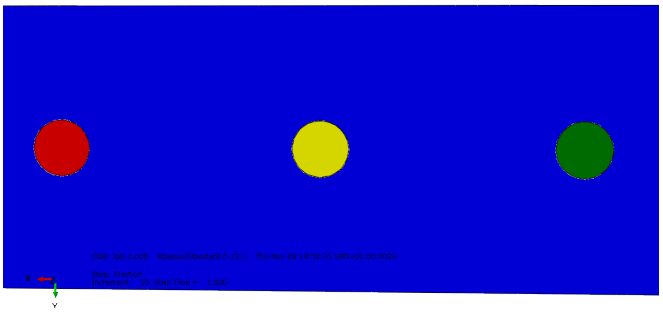
The question that may come up is how many models are needed. The model relies on 20 different variables and it is not feasible to vary every single one of them multiple times since this has a huge amount of computational cost and processing time. Therefore, a design of experiment analysis is performed to determine which are the most relevant variables to study and, with the ones selected, a sensitivity analysis is performed with the 15 models created to evaluate the effect of each of them in both stress concentration and shear load distribution.

The last part of the third section focus on extracting the shear load values for the three rivets. The procedure is based on fundamental beam theory and it consists on performing free-body cuts to get these shear values. This procedure is verified by the numbers obtained which are indeed coherent, so after performing this operation for each of the 15 models, the section is closed with a table that summarizes all the shear loads for the three rivets along the 15 models.

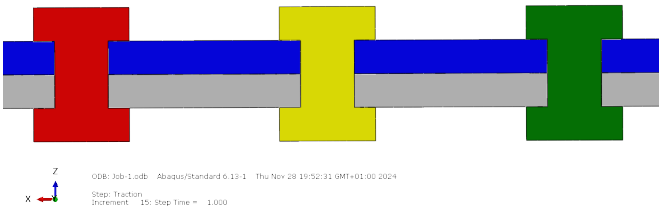
Lastly, the fourth section of the chapter, as it was anticipated previously, calculates what would be the analytic solution for the 15 different models, so then this solution is compared to the one obtained from the FEM calculations. The differences between them are quite large, and these are explained because the analytic solution does not consider out of plane effects, which in these project play a huge role. Taking them into consideration, we can observe that the external momentum exerted on the fastener cluster center of mass will make the assembly to rotate and hence load will be more concentrated in the inner rivet. As a conclusion, this aspect backs the necessity of performing a 3D analysis using FEM calculation in order to understand the entire picture and obtain realistic results.

3.1. Analytic shear load solution for a riveted assembly

The rivets shear load distribution provides extremely relevant information since it is the only way possible of transferring the traction loads between the two plates in the case of single shear. Thus, knowing the shear loads exerted by the rivets will allow us to find the shear load distribution along the plates which conform the assembly. As mentioned before, the model will be similar to the sketch shown in Appendix A(figure A.2), but in order to provide a clearer view of the model, the figure 3.1 can anticipate the main views from the model which, indeed, will be necessary to understand how this analytic procedure is applied.



(a) Upper view from model subjected to analysis



(b) Side cut from model subjected to analysis

Figure 3.1: Main views from the model

In order to find the analytical solution to our problem, in the book "*Airframe stress analysis and sizing*" by Michael C. Y. Niu it is explained how a external load is distributed between the fasteners conforming the assembly [21]. In section 9.4 of this book, where eccentric joints are analyzed, it is described how to proceed when making the calculations of the shear loads depending on the external excitations. In concentric riveted connections, when no moment is applied, all the fasteners are loaded proportionally to their cross-sectional area. In the case all the rivets have the same area, all these will be loaded evenly [21]. This also happens in the case where all the rivets are presented in line, despite this could be misleading.

Where fastener clusters must carry moment load (M) as well as shear force (P) in members they must be investigated for combined loads. on the fastener clusters, as seen in figure 3.2.

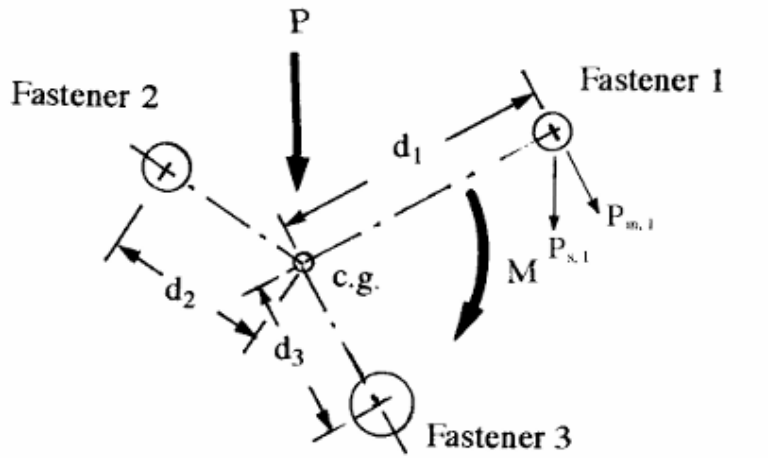


Figure 3.2: Riveted union with both shear force and external moment [21]

Therefore, the load carried by each fastener can be divided in two different contributions, the one coming from the shear force and the other coming from the external momentum. Notice that all loads can be translated into a combination of a shear load going through the centroid and an external momentum about the rivets centroid.

Stating with the first contribution, the shear load carried by each rivet based on the overall shear load (P) can be defined by the following equation [21]:

$$P_{s,i} = P \cdot \left(\frac{A_i}{\sum_{j=1}^{n} A_j} \right) \tag{3.1}$$

where A is the area of the fastener.

The second contribution provided by the momentum about the centroid can be expressed as follows [21]:

$$P_{m,i} = M \cdot \left(\frac{A_i \cdot d_i}{\sum_{j=1}^{n} A_j \cdot d_j^2} \right) \tag{3.2}$$

where:

- A is the area of the fastener.
- d is the distance between the fastener and the fastener clusters centroid.

Therefore, the total shear load expected for any of the fasteners is the sum of these two contributions:

$$P_i = P_{s,i} + P_{m,i} \tag{3.3}$$

These above equations are based on the following aspects [21]:

- Fasteners materials are the same

- Fasteners being on the same material and thickness
- Fastener shear load assumes straight line distribution

Therefore, the way to proceed when obtaining the shear load distribution of the fasteners would be as follows [21]:

1. Find the fastener areas.
2. Find the centroid (c.g.) of the fastener clusters (x_{cg} and y_{cg})
3. Calculate the shear load carried by each fastener due to the external load contribution, $P_{s,i}$.
4. Calculate the shear load carried by each fastener due to the momentum exerted about the centroid of the fastener clusters, $P_{m,i}$.
5. Perform the vectorial summation to find the final shear loads for each fastener.

Going back to the model that will be studied, we can start analyzing if this theory can be applied to our specific case. Finding the fastener area is trivial since the shank of the fasteners used are circular, and so the area of each specific fastener will be πr_i^2.

Secondly, the traction force in the case being analyzed in this project , is applied at the half width of the lower plate in its right end as per figure 3.1(a), and its direction is parallel to the x-axis. Hence, it is important to know which is the information retrieved from past studies regarding this aspect, which will be shown in the following section. Since the fasteners are aligned, it is also trivial to state that the centroid of the fastener clusters will coincide with the center axis of the yellow rivet. Since this rivet is also located half width of the plate, it can be concluded that the line of action of the external force will go through the centroid, which means no external momentum in this plane can be considered.

Hence, under these circumstances, we could think that, if the rivets were the same, they should be loaded equally. However, this will not be the case and the reason why this happens is because up to this point we have not considered out of plane effects, meaning it is relevant to be aware of the properties out of xy plane.

Therefore, considering the z coordinate in the analysis, we need to define our centroid with an extra coordinate value, so now, our centroid wont be located at the axis of the middle fastener, but more specifically, it will be right at the half way of the shank. In other words, the centroid will lay on the plane where both plates mate each other.

As it has been mentioned, the force will be applied in the lower plate (grey plate as per figure 3.1) at its free end, which is the one close to the red fastener. In addition to this information, it must be said that the force will be applied at half thickness of the plate. Since the force is not applied at the same z coordinate as the centroid, it is important to notice that a momentum will appear and consequently, will try to tilt the plate upwards by means of rotation. However, the upper plate (blue plate as per previously mentioned figures) will be clamped on its opposite end (close to the green fastener). Recall that a clamped boundary conditions implies, not only null displacement, but also a null inclination angle at that point with respect to the wall it is clamped to (meaning that the plate at its root is purely perpendicular to the wall).

If from one side the slope of the plates will be increased due to the momentum exerted by the external force, and from the other side this slope is constrained to be changed, this actually means that the slope of the plates will be changing spanwise. This makes the condition for each of the rivets different from one another, which basically tells us that they will not be carrying the same shear load. These out of plane effects are the reason why this analytic theory will not match the FEM results.

By intuition, we could expect the inner rivet (green rivet in figure 3.1) to be more loaded than the other since it is the one closest to the clamped edge where slope is restricted. However, since it is not trivial to find this distribution, it is one of the reasons why this project could provide us with such good information about the behaviour of the fastener clusters under these circumstances.

Summing up, we can state that the analytic procedure is not enough valid and it is required to perform a FEM analysis based on the following evidence:

- Out of plane effects are not considered by this theory
- Other manufacturing effects such as hole clearance is not taken into account.

This last aspect has a lot of influence if both the stress distribution and the shear loading of the fasteners. However, there are plenty of studies regarding this aspect which can reveal us which are the tendences and consequences of creating this effect.

3.2. First model

Recalling information from past chapters, in the first place the geometry of the model was briefly sketched, including its boundary conditions. Then, the second chapter focused more in the inside of the mathematics and defined the calculation procedure that will be used for this project. Specifically, it stated the reasons that supports the use of an implicit static analysis for this case, as well as the assumptions used in the material behaviour and the modelling criterion when defining model interactions such as contacts.

Therefore, having all the information needed to build the model and also the objectives set, the very first step consists in creating a sample model from which information can be extracted and thus, it will be used for future calculations which will be more accurate.

The model will consist in a two plate structure joined by three rivets, as seen in figure A.2, located in Appendix A. The top plate will be clamped to the wall, but the bottom one will be subjected to a traction force which tries to pull the plates out of the wall. The rivetted unions are represented in green, and these ones will also influence the behaviour of the model. What in reality would happen is that firstly, the three rivets will be installed by applying a pretension force in the inner cross-section of the shank of each rivet. Once the installation has been executed, the traction force will be applied to test how the structure reacts to this external action.

Therefore, in order to construct the model, it is required to follow the next steps described more in depth in Appendix-A:

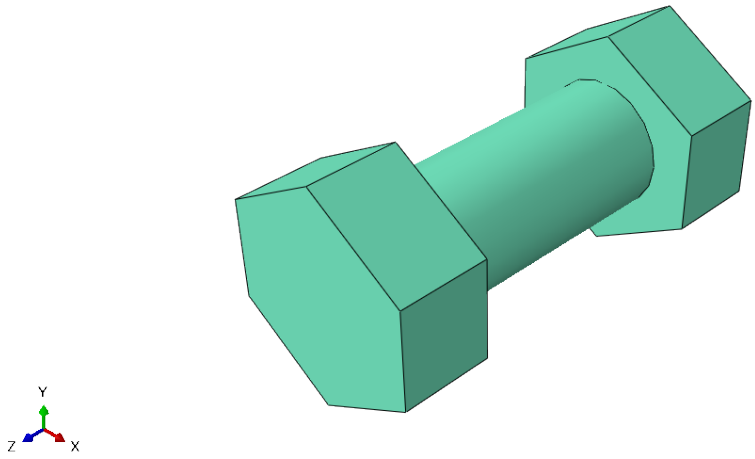
1. **Create the parts that will conform the assembly**: For this first model, both plates are the same and so are the fasteners used. Thus, only two independent parts are required to be modelled.

   Inside Abaqus we can create, as mentioned in chapter 2, many different element types. In the particular case of this thesis, the model will be made out of 3D deformable elements. One could think that because of metal sheets are very thin they could be modelled as a shell. However, since this project is interested in understanding what happens inside the mechanical assembly of the rivet, studying the contact area of the rivet shank and the hole, in order to do so it is key to select 3D elements to create the specimen subjected to analysis.

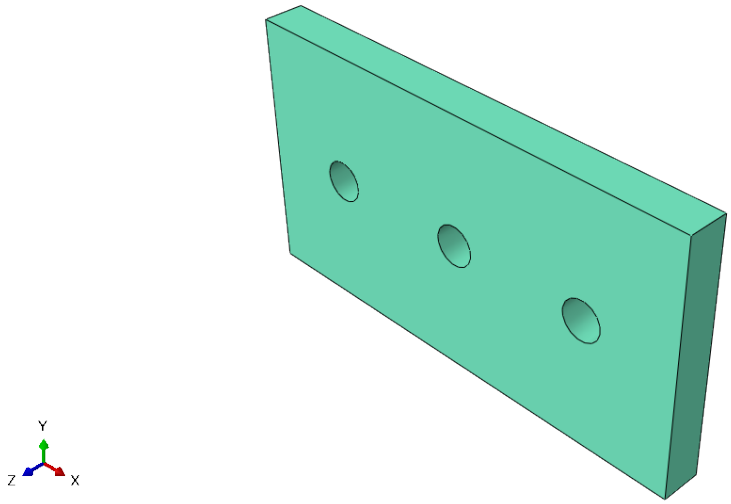
   Despite being both of them made out of 3D elements, for each one of the parts we need to take into account the following considerations:

   - Rivet: Rivet head dimensions (height and diameter) need to comply with the norm DIN660, which is the selected fastener [22].

   - Plate: Because of stress concentration requirements, there must be a pitch distance of at least four times the hole diameter. Moreover the distance between the hole center and the edge of the plate must be greater or equal to two times the diameter of the hole.



(a) Rivet for the test model



(b) Plate for the test model

Figure 3.3: Parts created for the model

2. **Create the materials used in the model and assign them to each on the parts**: In this model only one material will be used. Thus it is only necessary to introduce its values in the system and assign it to the parts by means of sections, which are created in "Property" module. As mentioned in chapter two, because of the nature of the problem, the material will be assigned only elastic behaviour, not considering plastic region nor damage tolerance effects.

   For this first model the material was steel with the following characteristics [23]:

   - **Density**: 7.9 g/cm^3
   - **Young's Modulus**: 198.5 GPa
   - **Poisson's Ratio**: 0.33

3. **Make the assembly**: Firstly, all the different instances are introduced, and each of them is identified with a different colour in order to differentiate them. The colour code that will be used for the parts is the one described below:

   - The **closest rivet** to the clamped section will be represented in **green**.

- The **middle rivet** will be represented in **yellow**.
- The **furthest rivet** to the clamped section will be represented in **red**.
- **Top plate**, corresponding to the one clamped, will be shown in **blue**.
- **Bottom plate**, where traction force is applied, will be represented in **white**.

Therefore in figure 3.4 it is shown the position of the rivets with respect to each other inside the assembly.

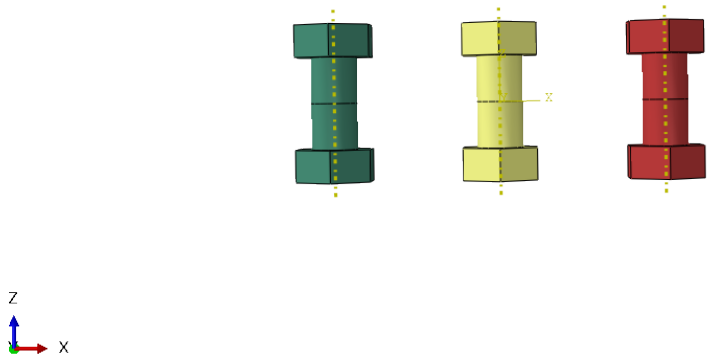


Figure 3.4: Screws placed in the space ready to be assembled

Then, by means of translations and rotations with the different instances they can be set in the way the result is reflected in figure 3.5.

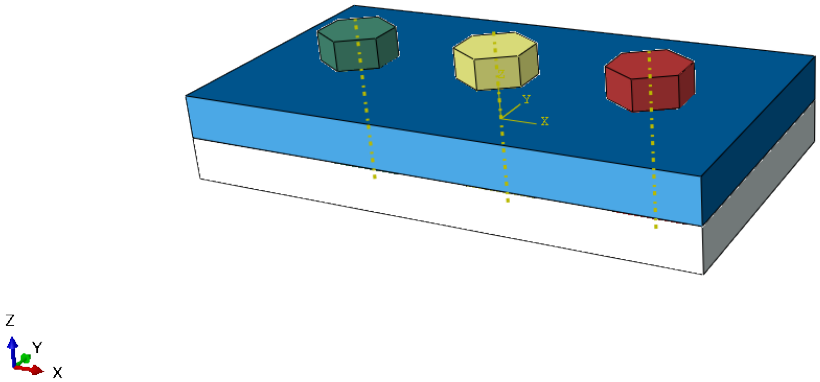


Figure 3.5: Assembly completed

4. **Define the time steps**: As it was predicted in the beginning of this section it is necessary to differentiate where the pretensions are made and when the traction occurs. Thus, these time steps allow to create and order the sequence of events.

Both pretension and traction steps have been assigned the following properties:

- Maximum number of increments: 10^6
- Initial increment size: 0.01
- Increment size range:$\left(10^{-4} - 0.1\right)$

5. **Define the interactions**: In this specific case, only contact interactions may appear, since there is no heat added but a purely mechanical model. These contact pairs need to be assigned a property, which in the end can be filled with multiple data.

   The contact pairs are the following ones:

   - **Rivet head contacting the top plate surface**: Since this is a rivet-plate interaction, the model will have three of this kind, one for each rivet. Here, as the rivet is the one trying to drag the top plate, the rivet head will be the master surface (shown in red contour in figure 3.6) and the plate will contain the slave surface, shown in purple.

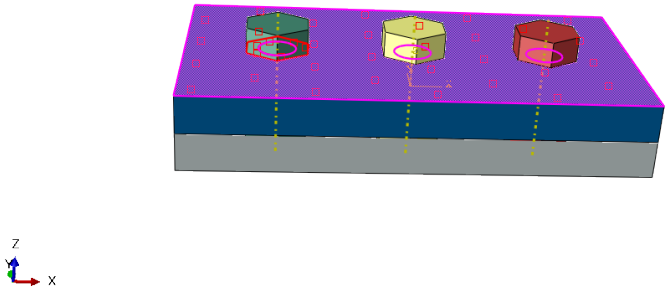


Figure 3.6: Head-plate interaction

   - **Rivet collar contacting bottom plate surface**: As it happened in previous item, this is a rivet to plate interaction, having thus a total of three of this kind. This time, since the bottom plate is the one which is being directly pulled, this time the plate will contain the master surface and the rivet collar, consequently the slave surface, as shown in figure 3.7.

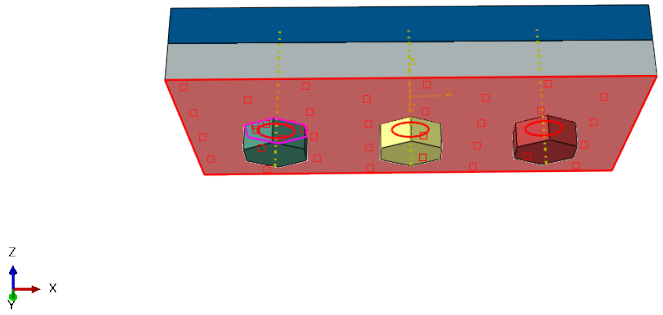
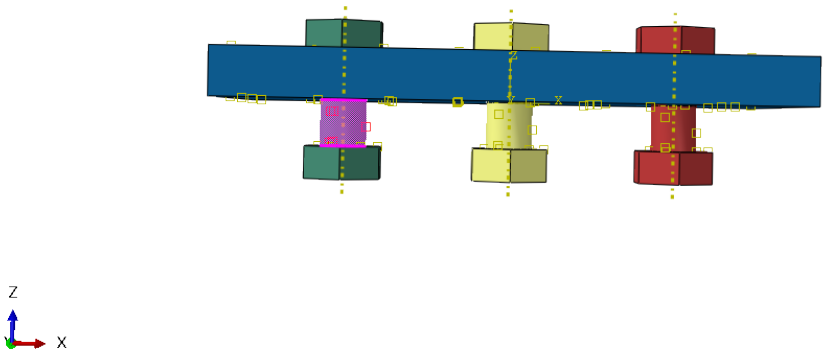


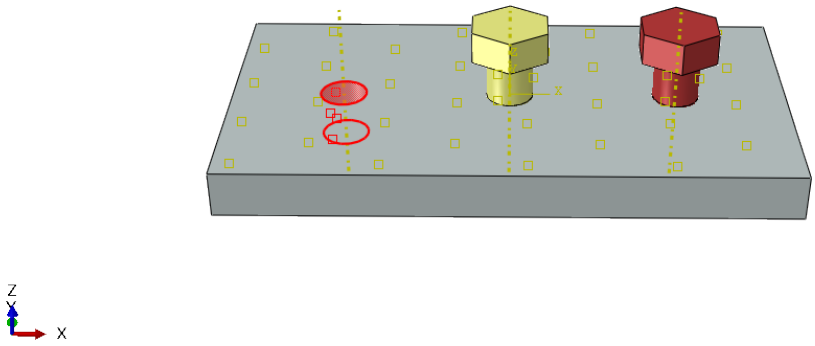
Figure 3.7: Collar-plate interaction

- **Rivet shank to plate interaction**: This is the most trivial and the one which will have the most relevance. Because the model consist on two plates and three rivets, the total number of interactions of this type is six, two plates for each rivet, three rivets for each plate.

  Following the same procedure as in previous interactions, the bottom plate will be the master surface since it is the instance that initiates the motion and leads to collisions. Hence, the part of the shank contacting this plate will be treated as slave surface, as shown in figure 3.8.



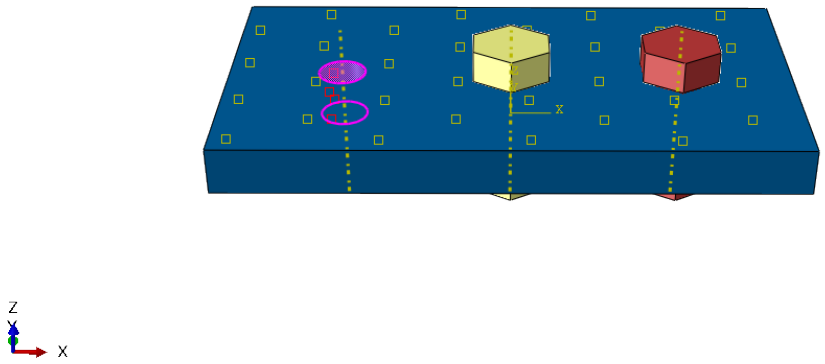
(a) Slave surface



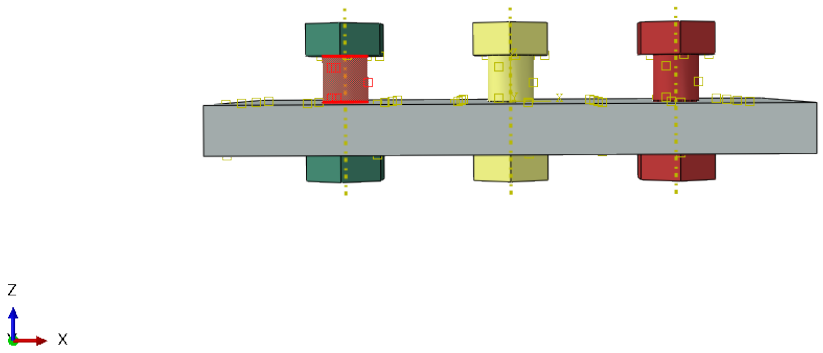
(b) Master surface

Figure 3.8: Lower plate - Rivet shank interaction

However, for the top part of the assembly the opposite will occur, since the rivet will carry the load from the bottom plate to the top one, leading thus to rivet containing the master surface and the top plate being the slave surface, as referenced in figure 3.9.



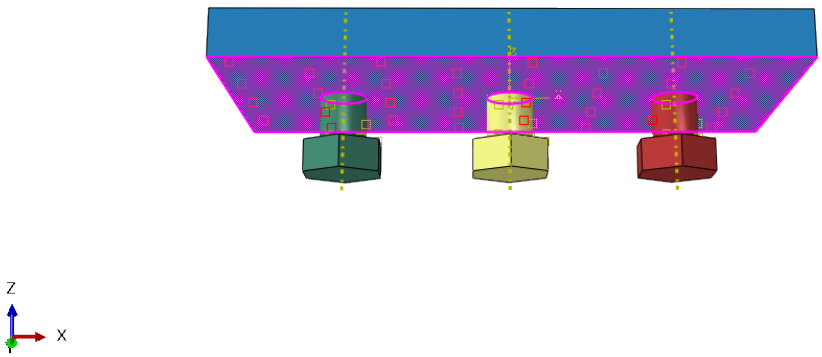
(a) Slave surface



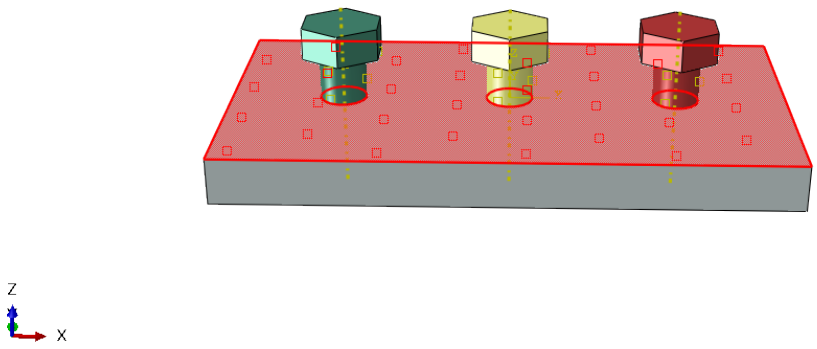
(b) Master surface

Figure 3.9: Upper plate - Rivet shank interaction

- **Plate to plate interaction**: This last interaction corresponds to the friction between the faying surfaces of the mating plates. As expected based on previous information, the master surface will be once again contained in the bottom plate.



(a) Slave surface



(b) Master surface

Figure 3.10: Upper plate - Lower plate interaction

All these make a total of **13 interactions** which all of them were assigned both normal (no penetration) and tangential behaviour parameters (this last was defined by introducing a friction coefficient of 0.3).

6. **Define the loads**: Create the different loads existing the model, which are the pretension forces for each one of the rivets and the external force applied at the bottom plate.

   First thing applied in the model are the pretension forces, which are indeed enforced in the cross-section of each rivet shank, as shown in the yellow arrows of figure 3.11

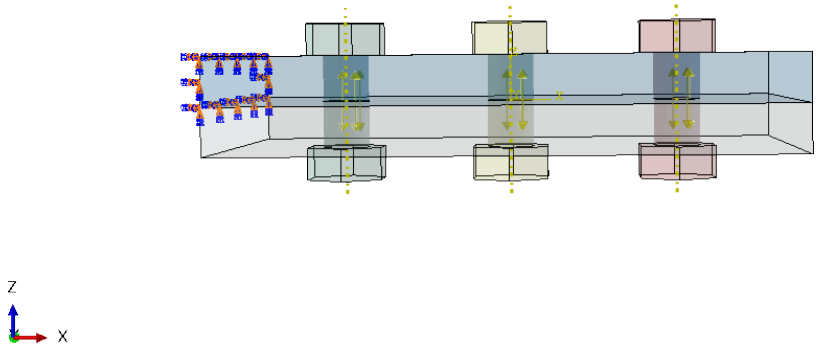


Figure 3.11: Pretension forces

Secondly, the traction force is introduced once the plates have been joined by the rivets. For this test, the traction force will be simulated as a surface force as represented with purple arrows in figure 3.12

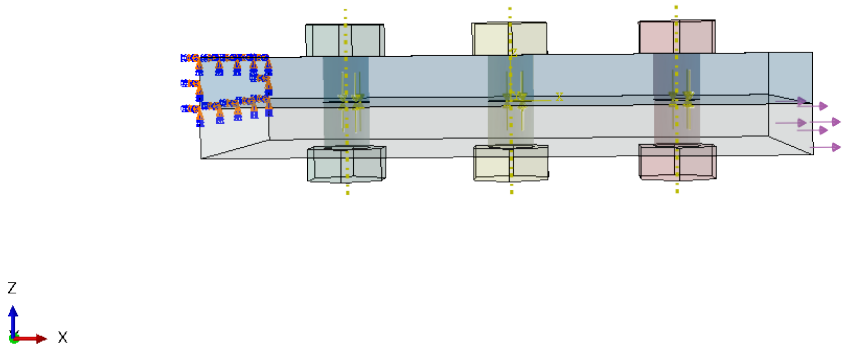


Figure 3.12: Traction force

Lastly, the clamped boundary condition is introduced here, and it can be seen in both figures 3.11 and 3.12 represented with orange cones surrounded by blue points. Having this defined, all model inputs have been specified.

7. **Mesh the assembly**: Like every other FEM software, the calculations are always discretized in 3D small elements (cells). Therefore it is important to define the size of this elements, their shape and the strategy of how these cells will be placed to cover up the entire domain of the model.

   As it can be seen in figure 3.13, the strategy followed was to perform a sweep around the axis of the three fasteners applying a smooth transition between the seeds close to the hole edge and the ones situated in other places inside the two plates, which can be appreciated in the deformation of the rectangular seeds close to the three holes. The result is a good mesh which, by looking at it, it is clear to see that both

orthogonality and aspect ratio are great. Recall these two parameters measured the mesh quality. Also the size was selected making a balance between computational cost and calculation accuracy, which having it combined with the quality of the mesh makes this model ideal and ready for the calculation.

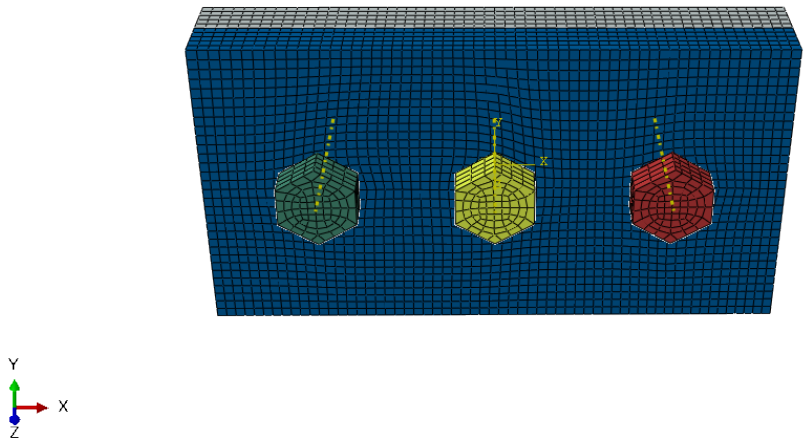


Figure 3.13: Mesh of the model generated

8. **Run the job**: Finally the last step consists in running the model and observe it convergence has been reached.

After all these steps have been completed, it is crucial to check if the results are coherent with the loads and conditions assigned to the model. Let's analyze the physics going through each of the steps.

During the first step, the riveting operations are performed, and these ones consist in applying a force to the shank cross-section which makes the head of the rivet press the plate surface. Since the collar, simulated with same shape as the head makes the same action in the other plate, consequently the plates are pressed against each other. Therefore, not only a force, but also a momentum is exerted to the plates by the rivet in the way described in figure 3.14.

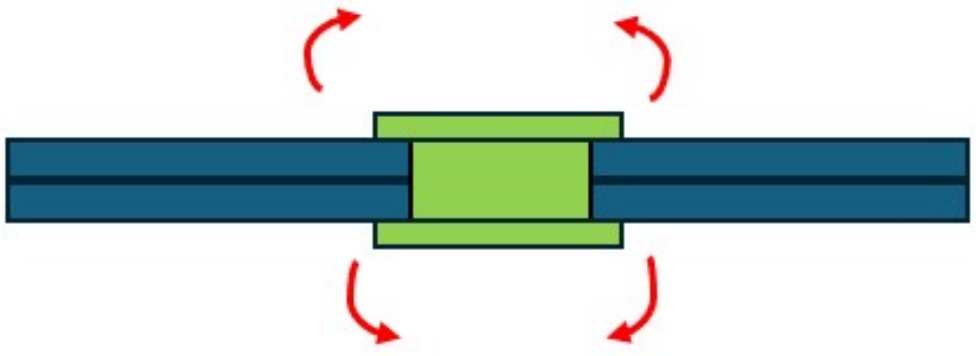


Figure 3.14: Moment created when riveting on the plates

Driving to the next time step, since the traction force is applied in the negative side of plane XY, this force will make the plates, that when rivetted against themselves will act like a single instance or similar, rotate in

counterclockwise direction (angular vector pointing towards negative y axis). Another aspect to consider is that the force direction is following the rotation of the plate, which is indeed an unstable effect for the model.

Coupled with the moment described in the previous paragraph, what user should expect is to watch the plates rotate in the described way, as these two start to separate at the inner rivet position, consequence of that momentum exerted by the pretension, and because the top plate is clamped whereas the bottom one is not.

As well because as mentioned the force follows the rotation, this unstable condition will lead the force concentration to be located at the inner rivet which will be the critical point in the system to prevent failure, as the other two rivets will start to transmit load as the inner rivet cannot withstand the load by itself. Therefore, it is also expected that the inner rivet will carry more load than the middle one, and the same will happen between the middle one and the outer, which will be the one more relieved.

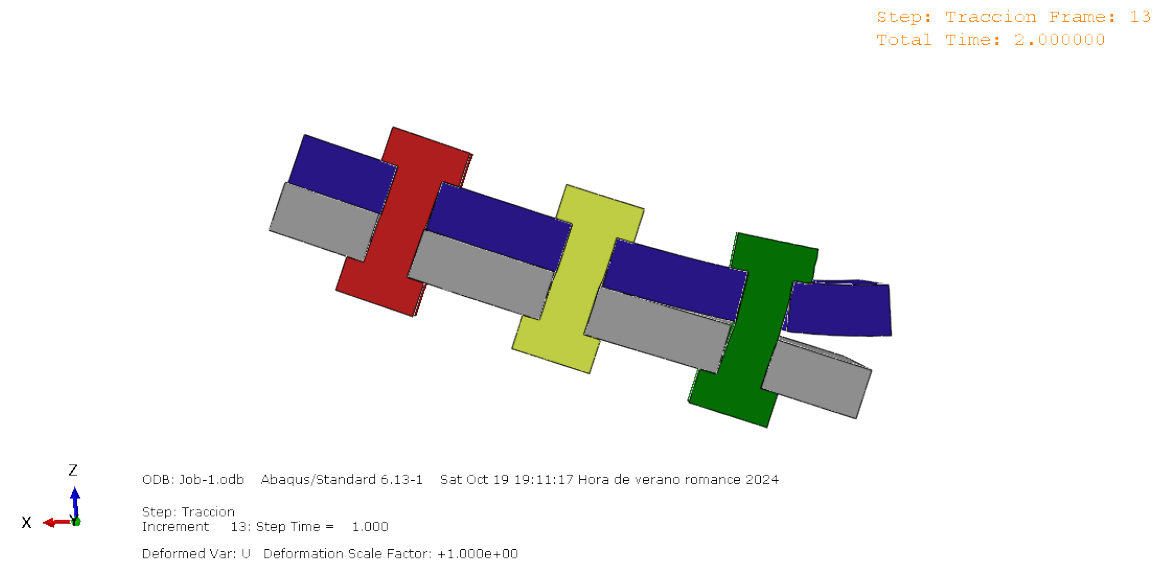


Figure 3.15: Job results in terms of deformation

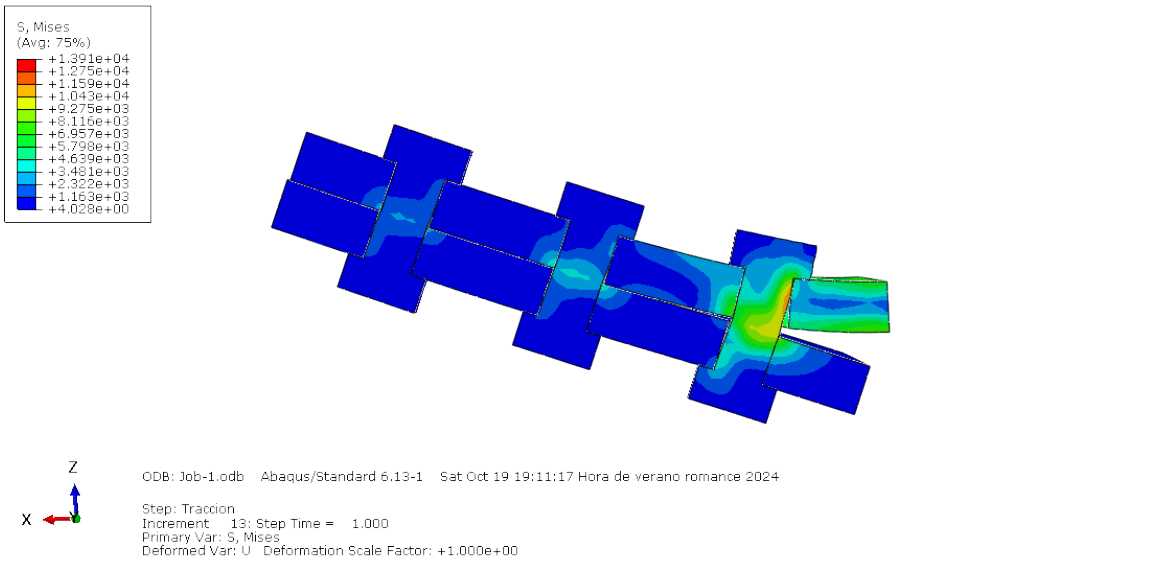


Figure 3.16: Job results in terms of stress distribution

As seen in figure 3.16 the stress is concentrated in the inner rivet, as expected. Moreover both plates rotate in the expected direction, as the gap between the two plates increase while plate rotates.

In addition, figures 3.17 and 3.18 are included since they were used to verify that indeed the rivets were being installed during the pretension step. By using the proper augmentation factor the two statement hereafter can be seen:

1. **Rivet head deformation**: Due to the pretension, the head of the rivet is pressed towards the plate.

2. **Plate bending**: Since the plates are fastened by three points, the rest of the plate, thanks to the moment exerted by the rivets, will bend, creating a separation with respect to the other plate as seen in figures 3.18 and 3.19

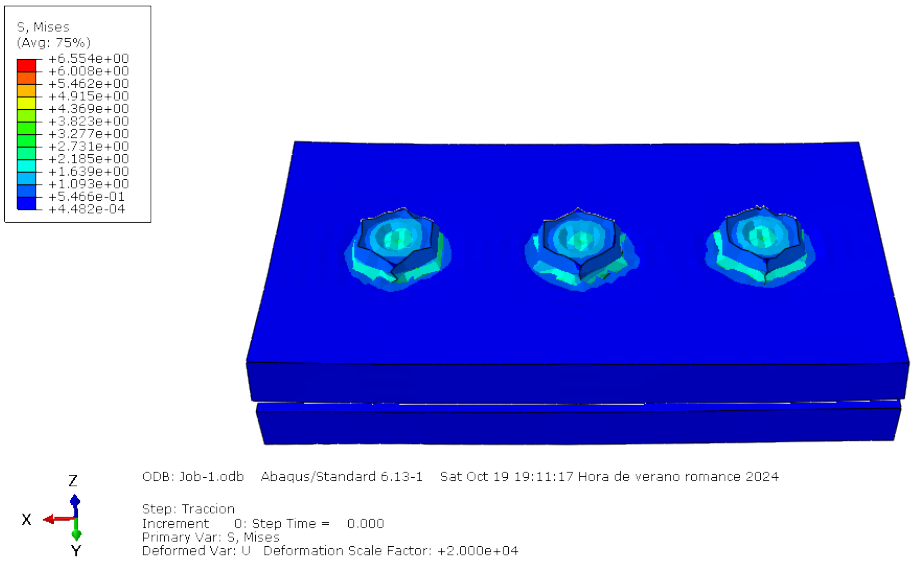


Figure 3.17: Rivet head deformation

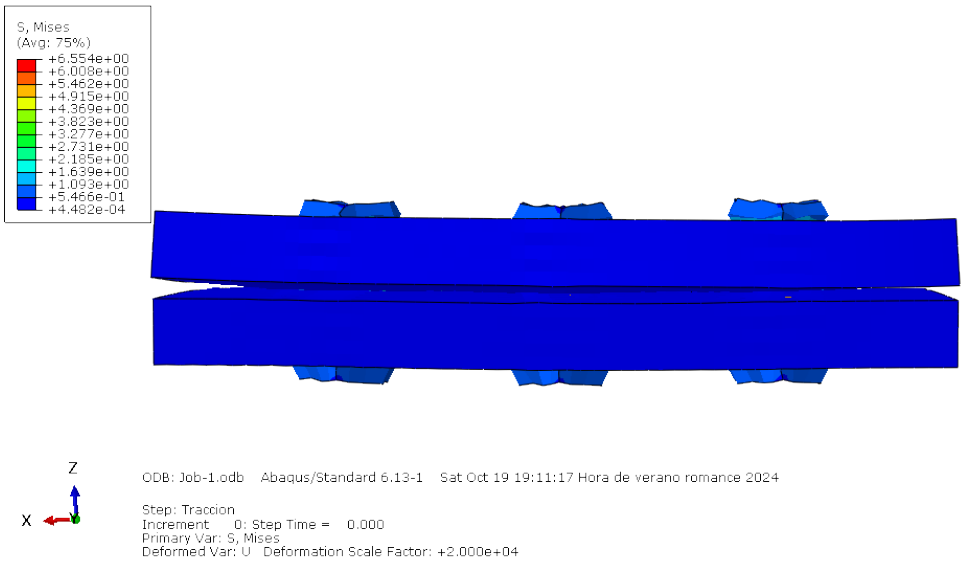


Figure 3.18: Plates bending during pretension (x-z plane)

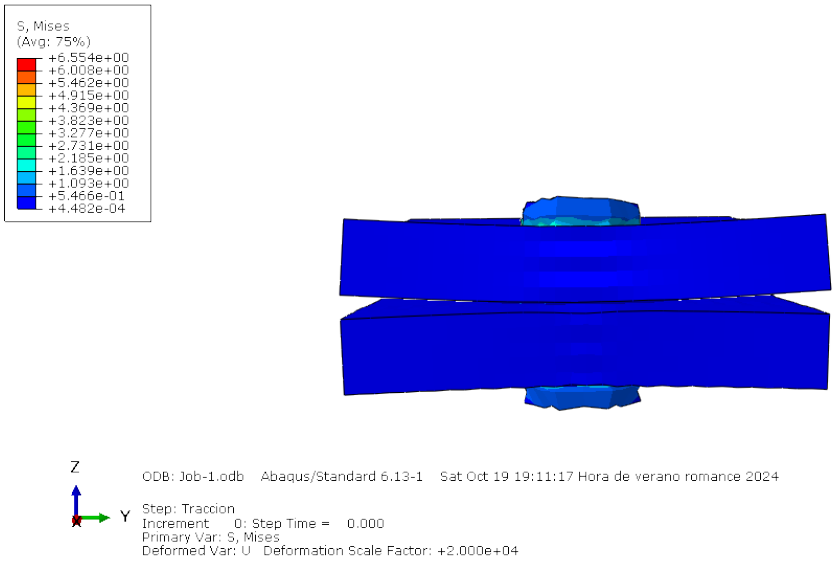


Figure 3.19: Plates bending during pretension (y-z plane)

Hence, it has been verified that results are coherent and thus the interactions defined during the process work as they should. Nonetheless, the model had two plates which had no offset between them. In other words, the top plate was entirely laying on the bottom one. However, as it can be seen in figure A.2 there exists a part of both plates which is not in contact with the other plate, so once this trial model was verified, a second model was created introducing this distance which is afterwards explained and shown in figure 3.24(a). Having as a starting point the already performed trial model, it was straightforward to construct this second model as the only step which was different was the plate design.

After creating this model, the results were also verified and the model behaviour was the same as in the trial model, so once this was finished, all the information collected was enough to jump to the next section of this thesis.

3.3. Parametric model creation

3.3.1. Rpy analysis

After building the entire model taking into account all the modules mentioned in Appendix A, it was relevant to check that indeed the job output was coherent, because from that CAE file would be the basis for the next steps.

Until this moment, only CAE file was manipulated and all the data related to the project was entered inside that environment. For one single study case, that might be the most efficient way to work because the options and buttons are very intuitive and thus, easy to understand for all users. However, since the aim of this project is to analyze the influence of each of the design variables in the load case, it will require performing several iterations. Making this calculations modifying everything in CAE environment would take a huge amount of time. Instead, the objective is to take advantage of the outputs Abaqus returns while performing the project.

As the user performs the project in Abaqus CAE, a RPY file is generated in the same folder where the project is being run (if not saved this file will be saved in temporary folder by default). This RPY contains the same information entered in Abaqus CAE but translated to python commands. Thus, if it is possible to understand those commands and interpret them correctly, by only changing some wording of that file we can change the model without having to create it again from nothing.

Therefore, the next step to do once the model has been finished is to extract the RPY file associated with the model and link the different operations performed in the model with their corresponding coding lines. In order to make it easier, while performing the model, each time a single step was made, the project was saved. Every time this happened, a coding line appeared indicating this aspect so when making the revision looking for the lines related to a certain operation it was straightforward to find them. For instance, it can be found hereafter in figure 3.20 and 3.21 the lines created by the own program while performing their corresponding actions:

```
session.viewports['Viewport: 1'].setValues(displayedObject=None)
session.viewports['Viewport:
1'].partDisplay.geometryOptions.setValues(
    referenceRepresentation=ON)
s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
sheetSize=10.0)
g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
s.setPrimaryObject(option=STANDALONE)
s.CircleByCenterPerimeter(center=(0.0, 0.0), point1=(2.4, 0.0))
p = mdb.models['Model-1'].Part(name='DIN660-4,8-4',
dimensionality=THREE_D,
    type=DEFORMABLE_BODY)
p = mdb.models['Model-1'].parts['DIN660-4,8-4']
p.BaseSolidExtrude(sketch=s, depth=4.0)
s.unsetPrimaryObject()
p = mdb.models['Model-1'].parts['DIN660-4,8-4']
session.viewports['Viewport: 1'].setValues(displayedObject=p)
del mdb.models['Model-1'].sketches['__profile__']
mdb.save()
#: The model database has been saved to
"C:\Users\Usuario\Desktop\TFM\INTENTO BUENO RPY\MODELO
BUENO.cae".
```

Figure 3.20: Coding lines used to create the rivet shank

```
session.viewports['Viewport:
1'].assemblyDisplay.setValues(interactions=ON,
    constraints=ON, connectors=ON, engineeringFeatures=ON,
    adaptiveMeshConstraints=OFF)
mdb.models['Model-1'].ContactProperty('Contact')
mdb.models['Model-
1'].interactionProperties['Contact'].TangentialBehavior(
    formulation=PENALTY, directionality=ISOTROPIC,
slipRateDependency=OFF,
    pressureDependency=OFF, temperatureDependency=OFF,
dependencies=0, table=((
    0.3, ), ), shearStressLimit=None,
maximumElasticSlip=FRACTION,
    fraction=0.005, elasticSlipStiffness=None)
mdb.models['Model-
1'].interactionProperties['Contact'].NormalBehavior(
    pressureOverclosure=HARD, allowSeparation=ON,
    constraintEnforcementMethod=DEFAULT)
#: The interaction property "Contact" has been created.
mdb.save()
#: The model database has been saved to
"C:\Users\Usuario\Desktop\TFM\INTENTO BUENO RPY\MODELO
BUENO.cae"
```

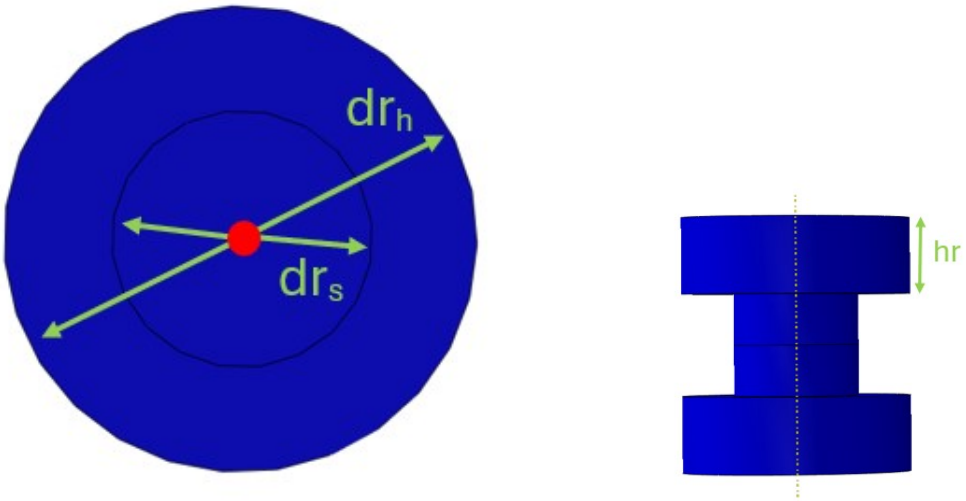
Figure 3.21: Coding lines used to create the "contact" property

The code generated by the software contained in the order of one thousand lines, so it took some time to fully understand all of them. After doing so, next step was to define the parameters prone to be used, which in other words are the design variables, and introduce them inside the generated code.

The design variables related to the rivets are:

- Inner rivet shank diameter, dr_{s1}
- Center rivet shank diameter, dr_{s2}
- Outer rivet shank diameter, dr_{s3}
- Inner rivet head diameter, dr_{h1}
- Center rivet head diameter, dr_{h2}

- Outer rivet head diameter, dr_{h3}
- Rivet head height, hr

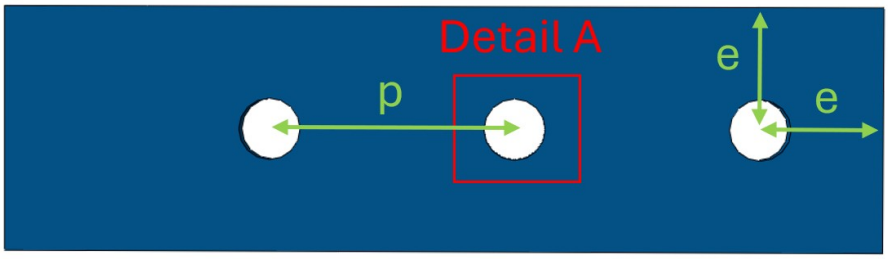


(a) Rivet diameters (b) Rivet head height

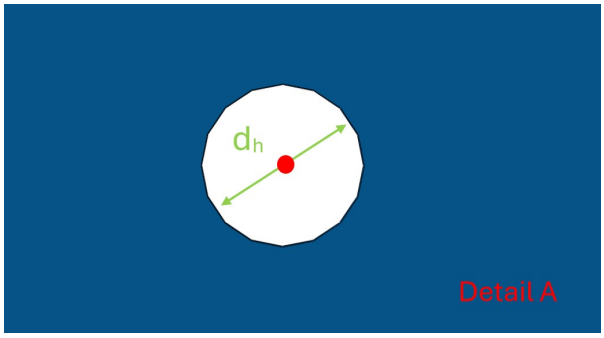
Figure 3.22: Rivet design parameters

Moreover, it is required to add the variables related to the plates which are:

- Inner hole diameter, d_{h1}
- Center hole diameter, d_{h2}
- Outer hole diameter, d_{h3}
- Pitch between holes, p
- Distance between the edges of the plate and hole center, e
- Offset measurement in the x-direction, o_x
- Thickness of plate 1, t_1
- Thickness of plate 2, t_2

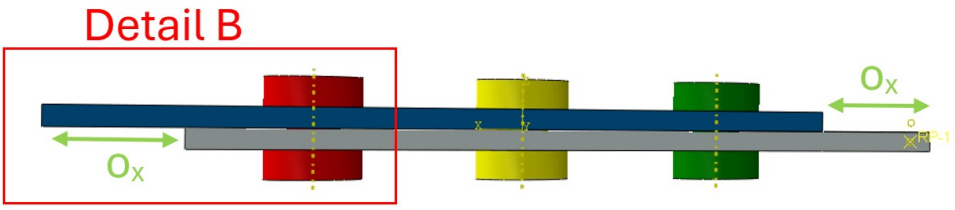


(a) Pitch and edge disance

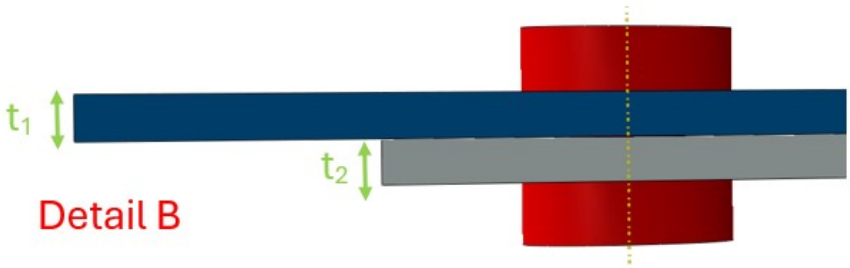


(b) Hole diameter

Figure 3.23: Plate design parameters plane XY



(a) Offset between planes in x-direction



(b) Plates thicknesses

Figure 3.24: Plate design parameters plane XZ

And so the ones related to the loads are also added:

- Inner rivet pretension force, F_{in}
- Center rivet pretension force, F_{cent}
- Outer rivet pretension force, F_{out}
- Traction force, F_{trac}

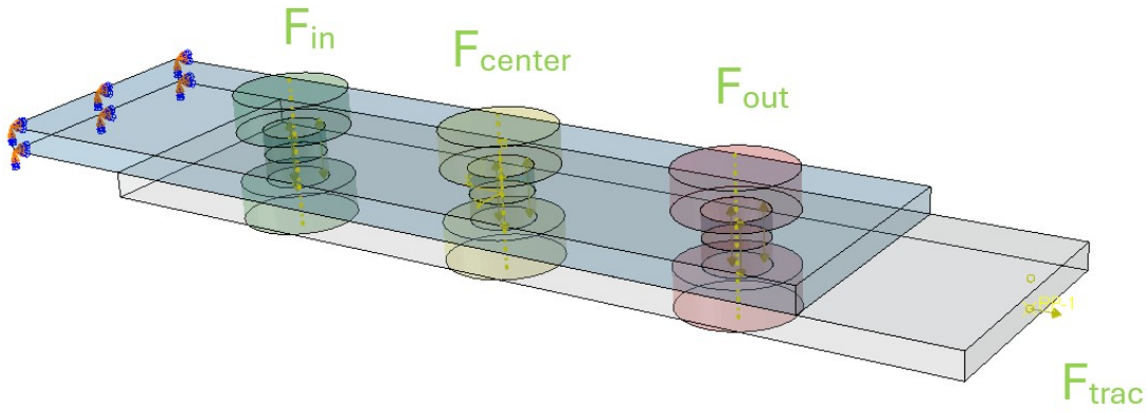


Figure 3.25: Forces applied in the model

As well of the rest of the variables not geometry related:

- Mesh seed size, m_{size}

Thus, all the parameters can be summarized in table 3.1, and they are represented in figure 3.27.These parameters previously mentioned had to be introduced as variables in the code. Firstly, the ones associated to the rivet geometry were easy to substitute since they were presented isolated in the code and their presence was limited to the part of the code where the rivet indeed is created. More precisely, when performing the first extrusion that forms the rivet (its shank) the second point which defines the circle is the rivet shank diameter divided by two. The other two variables are introduced in both second and third extrusion (head and collar of the rivet, respectively) when defining the radius of the head and afterwards, the height of that extrusion.

Similarly, the plate hole diameter and the plate thicknesses which are introduced isolated, those are straightforward to find and substitute their scalar value by the parameter itself. However, the rest of the variables related to the plate are not defined explicitly and needed to be introduced in the presence of other parameter previously mentioned. This aspect is easier to understand when looking at figure 3.26, which may be used as support when understanding the strategy behind the plate design.

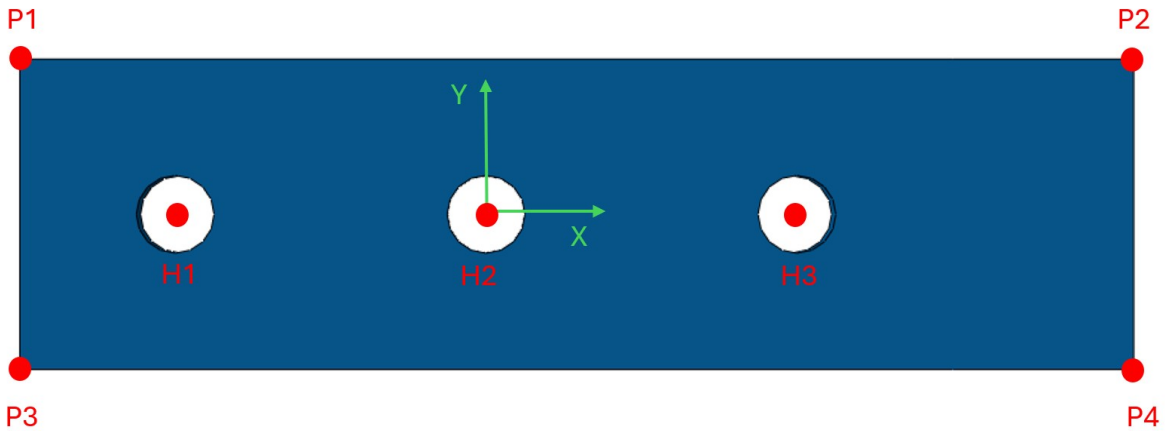


Figure 3.26: Reference points used to create the plate

In order to be easy to introduce the parameters, the plate perimeter was defined by straight lines from point to point. Taking into account the reference system shown in figure 3.26, the coordinates of the four points which enclose the area of the plate where:

$$P_1[x,y] = [-(p+e), e] \tag{3.4}$$

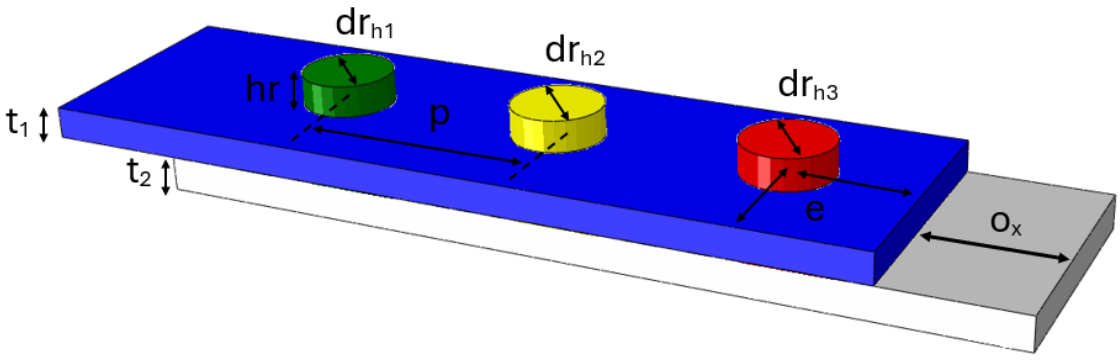
$$P_2[x,y] = [p+e+o_x, e] \tag{3.5}$$

$$P_3[x,y] = [-(p+e), -e] \tag{3.6}$$

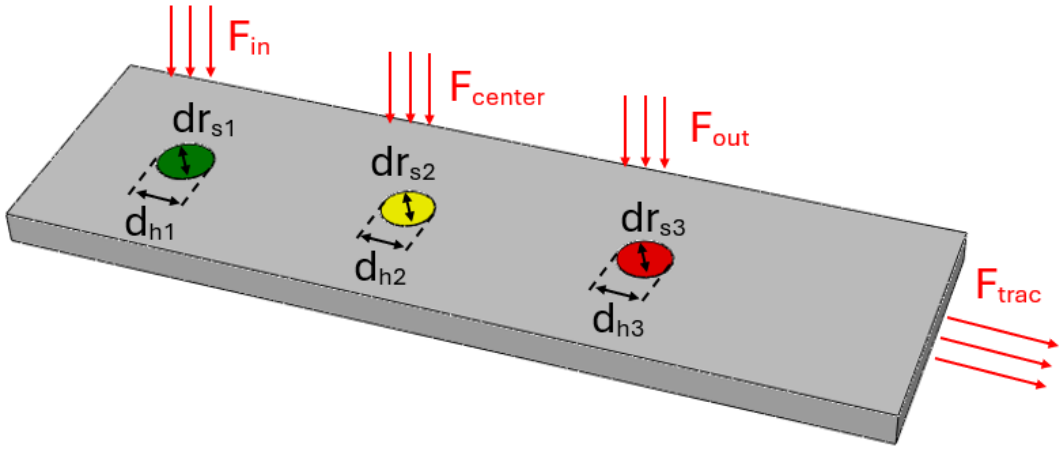
$$P_4[x,y] = [p+e+o_x, -e] \tag{3.7}$$

Design variables nomenclature	
Variable	Nomenclature
Inner rivet shank diameter	dr_{s1}
Center rivet shank diameter	dr_{s2}
Outer rivet shank diameter	dr_{s3}
Inner rivet head diameter	dr_{h1}
Center rivet head diameter	dr_{h2}
Outer rivet head diameter	dr_{h3}
Rivet head height	hr
Inner hole diameter	d_{h1}
Center hole diameter	d_{h2}
Outer hole diameter	d_{h3}
Pitch between holes	p
Distance between the edges of the plate and hole center	e
Offset measurement in the x-direction	o_x
Thickness of plate 1	t_1
Thickness of plate 2	t_2
Inner rivet pretension force	F_{in}
Center rivet pretension force	F_{center}
Outer rivet pretension force	F_{out}
Traction force	F_{trac}
Mesh size	m_{size}

Table 3.1: Design variables nomenclature



(a) Design parameters isometric view



(b) Design parameters middle cut

Figure 3.27: Design parameters represented in the model

It is observable that these last points already include three of the plate variables. Nonetheless, it was still necessary to include them when creating the geometry of the holes. In order to create them, it was necessary to define two points for each one of those, their center point and a second point where the circumference went through. The y-coordinate for these points, as seen in figure 3.26 is null.

Therefore, the two points (x-coordinate) used to create each of the three holes are:

$$H_1[x1, x2] = [-p, -\left(p + \frac{d_{h1}}{2}\right)] \tag{3.8}$$

$$H_2[x1, x2] = [0, \left(\frac{d_{h2}}{2}\right)] \tag{3.9}$$

$$H_3[x1, x2] = [p, \left(p + \frac{d_{h3}}{2}\right)] \tag{3.10}$$

With these changes introduced, the plate definition was completed. However, since in the pilot model only one plate was defined as a part, there was no way to make the thickness of the two plates different from one another. The same happened with the diameters of the rivets (for both head and shank) since all fasteners were the same in the pilot model. Therefore, a change in the model code was necessary.

One option would have been repeat the model again defining each of the parts separately in part module, but this option required a lot of time and start from nothing once again. Instead, a fairly easier solution was to identify the lines used to create both the rivets and the plate in the pilot model and paste them once again just in the place of the code where items are created. Only thing that was necessary was to introduce the different variables of each part and name them differently according to the properties used. All these changes were only introduced in the lines which referred to the part module, since afterwards, when creating the assembly the different parts are renamed as instances. Making the instances coincide with the ones created in the pilot model was the optimum way to proceed in order to need no further changes regarding this aspect in the script.

Additionally, two extra changes were performed in the development process of the new model. Fisrtly, the new rivets would change their shape for the heads, becoming circular instead of hexagonal. The reason why they were hexagonal in the first place was because the first trials were performed using that shape as a head, but looking carefully at the DIN660 norm, the shape of the rivet head is circular.

The second change introduced consisted in updating the materials used. In the trial model, steel was used for both the plates and the fasteners. Instead, this time the plates will be made out of aluminum 2024-T3, which is more realistic since it is a lighter material prone to be used in covers or fuselage panels.

The properties of this aluminum were the following ones [24]:

- **Density**: 2.78 g/cm^3
- **Young's Modulus**: 73.1 GPa
- **Poisson's Ratio**: 0.33

The rest of the variables are introduced once again isolated so they did not require a lot of effort to find their corresponding scalar values in the code and replace them. Since most of the forces and contacts involve surfaces created during the interaction definition, those did not need to be adjusted. Nevertheless, since the traction force was exerted at the center of the plate, this point needed to be defined with the mentioned variables using this definition:

$$F_{point}[x, y, z] = [(p + e_d + o_x), 0, \left(\frac{-t_2}{2}\right)] \tag{3.11}$$

Finally, once all the parameters were introduced in the code, the last step was to verify all the movements required for the assembly to be completely adjusted. Firstly, the places where the instances are introduced have to be separated by enough distance so they do not collide as soon as they are called. Once that problem was solved, it was necessary to keep track of those distances between elements defined so when performing the assembly, the parts were connected correctly.

For instance, the center of the middle hole coincides with plate's reference system origin. So does the centre of the shank diameter with respect to its local system of coordinates. Therefore, since in the assembly these two instances are introduced by introducing an offset between them in x-direction (so they do not

collide when introduced), in order to join them it is only required to translate the rivet the same amount of the offset back in the opposite x-direction, so it will end up lined up in the perfect position.

Likewise, a similar exercise was made for the rest of the instances in order to complete the assembly. In the code, these offset distances are parametrized so it is easier to perform the assembly and handle the different translational and rotational operations.

Joining all these concepts made possible to create a code which could create a model making variations in each of the design variables mentioned.

3.3.2. Rpy file generator

With the code derived in previous subsection, the model could be modified inside the range defined by the design variables, but still was not that efficient. If the user wanted to try many different combinations, it would need to change the file and run the program, or alternatively, generate the files which cover all the desired possibilities, which may take a very long time. Hence, if the objective was to run several attempts, a more efficient solution was needed. Therefore, the next task was to create a file which, by itself, was able to generate all these files not desired to be created manually, so it can save the user some time.

Considering the work developed in previous subsection, creating the desired code was as simple as copying the lines but telling the program that instead of running those lines, it has to write them into another file. Therefore the structure of this new script follows this pattern:

```
filename = 'name of the file' + '.rpy'
file = open(filename,'w')
file.write('Line 1' + str(parameter_1) + '\n')
file.write('Line 2' + str(parameter_1) + '\n')
file.write('Line 3' + str(parameter_2) + '\n')
file.write('Line (n)' + str(parameter_x) + '\n')
file.close()
```

Figure 3.28: Rpy generator pattern

so the major part of the file will have a look similar to the one of figure 3.29:

```
file.write("p = mdb.models['Model-1'].parts['DIN660-" +
str(diameter) + "-" + str(shank) + "']" + "\n")
file.write("f, e = p.faces, p.edges" + "\n")
file.write("""t = p.MakeSketchTransform(sketchPlane=f[1],
sketchUpEdge=e[0],
    sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, origin=(0.0,
0.0, """ + str(shank) + ")" + "\n")
file.write("""s1 = mdb.models['Model-
1'].ConstrainedSketch(name='__profile__',
    sheetSize=15.75, gridSpacing=0.39, transform=t""" + "\n")
file.write("g, v, d, c = s1.geometry, s1.vertices,
s1.dimensions, s1.constraints" + "\n")
file.write("s1.setPrimaryObject(option=SUPERIMPOSE)" + "\n")
```

Figure 3.29: Sample code of the file generator

Having this set, it is possible to make the computer generate the files for us.

But if this last script is transformed to a function type, this function could be introduced in a loop (design matrix) where all the parameters are covered and changed consequently. This allows to create all the rpy files desired by the user in a matter of seconds.

Considering the design parameters as the inputs, and taking into account that the output of the code is nothing wanted inside the file itself (the output is another file generated in the same folder where the script is located), the function definition is the following one:

def **RpyFileCreator**(dr_{s1}, dr_{s2}, dr_{s3}, dr_{h1}, dr_{h2}, dr_{h3}, hr, d_{h1}, d_{h2}, d_{h3}, t_1, t_2, o_x, e, p, F_{in}, F_{cent}, F_{out}, F_{trac}, m_{size}):

3.3.3. DOE: Design of Experiment

As anticipated previously, the main advantage of defining the file generator script as a function is the fact that it is possible for the user thus calling it all the times desired and generate multiple files if this script is placed inside a loop.

The main problem with the model which has been developed is that it contains 20 independent variables (defined in section 3.2.1 of this thesis). Considering that we only took two values for each of the varibles, all the different posible combinations result in 400 models. Of course, it is not viable to perform that ammount of calculations because of the following reasons:

- **Time**: Despite being able to generate the files in seconds, the calculation period that it takes to perform those analysis is extremely large.
- **Not all variables have the same relevance**: Depending on the object of study and the model, there will exist variables that play a more important role compared to other despite being all of them necessary to build the assembly.

Therefore, the next step was to perform a design of experiment, which consists in analyzing the importance of each quantity to the final output and thus perform the sensitivity analysis with the ones which are more relevant.

Recalling to the first chapter of this thesis, the main motivation of this project is to analyze the effect of gap between the shank and the hole inner surface when installing a line of fasteners since this one made the installation easier but it was not straightforward to quantify its effect. Therefore, taking this into consideration, it can be made the following statements:

- **Regarding mesh size**: This parameter will handle the balance between computational cost and accuracy. However, neither of these aspects is desired to be quantified nor analyzed. Thus, it is important to find a value which makes the results accurate and is able to run. But once this value is found, there is no ground behind to keep changing it during the iterations. This concludes that in the DOE this value should remain constant with a value which fits the requirements recently mentioned.
- **Regarding forces**: The pretension forces are usually defined in standard norms used during manufacturing processes. Of course this makes sense in order to avoid product damage or possible failures. Therefore, this forces should not vary either inside the DOE since they are bounded by the process or the project norms in which the assembly is manufactured. Moreover, the traction force is what makes the rivet carry the load, it is not part of the assembly itself. Therefore, since this is the external excitation used to try the model, if the aim is to compare the behaviour of different samples, it is convenient to remain the external excitation unchanged along the DOE activity.
- **Regarding plate distances**: Both pitch and edge are usually adjusted to meet a minimum distance criteria no matter what the model is. The edge distance shall be at least two times the diameter of the hole close to the side of the plate. As well the pitch between holes must be four times the diameter of each. These rules are applied in order to avoid too large stress concentrations in the borders of the hole which may lead to crack initiation and further propagation. However, since the greater the number of fasteners is, the stronger is the assembly, usually the mentioned distances do not change by a big amount compared to their corresponding lower bound. Therefore, since the margin of eligibility is not that significant, for this project these two values have been set close to their lower bound but they will not change during the DOE.
- **Regarding rivet head height**: This variable has no impact in stress concentration, and that is the reason why in the first placed it was not differentiated for each of the rivets, thus making it irrelevant for the project purposes

These aspects leaves the analysis subjected to the following variables, which will be the ones that will suffer changes all over the DOE matrix:

- **Shank and hole diameters**: These are the ones which make the gap exist in the first place so they should be definitely considered in the analysis. Of course, it is also important to analyze the position of that gap and the number of gaps if there were inside the model.
- **Plates thicknesses**: If both plates have not the same thickness this could vary the stress concentration and thus they should be considered in the DOE.
- **Offset**: This parameter will indicate the distance between the inner hole and the clamped section, as well as the distance from the outer hole to the traction force point of application. This will have a direct influence in the moment created, thus making it essential in the DOE.

With this quick reasoning the number of variables which will be varied during the DOE have been reduced from 20 to only 9. In addition, some of them are somehow related between them. For instance, in order to create a gap it is only required to change the shank diameter or the hole diameter, but not both. This type of relations bound the DOE to a lower scale which easier to handle. Thus, a chart of 15 different models were created to evaluate these parameters mentioned recently.

As mentioned before, 11 of the 20 initial design variables have been fixed along the DOE analysis. Therefore these variables will have their corresponding value according to this list:

- **Rivet head height**: 3 mm
- **Inner rivet head diameter**: 8.6 mm
- **Middle rivet head diameter**: 8.6 mm
- **Outer rivet head diameter**: 8.6 mm
- **Edge distance**: 12 mm
- **Pitch**: 22 mm
- **Inner rivet pretension force**: 1000 N
- **Middle rivet pretension force**: 1000 N
- **Outer rivet pretension force**: 1000 N
- **Traction force**: 1000 N
- **Mesh size parameter**: 0.9

Meanwhile the other 9 parameters will be changed along the 15 models created to perform the sensitivity analysis in accordance with table 3.2. This exercise will allow us to evaluate the different results and compare them between themselves, so the influence of each parameter is shown.

The variables contained in table 3.2, joint with the previous list of fixed values, could be summarized in a table of 20x15 (number of design variables x number of models) so each one of the rows correspond to a specific model. The result is the one shown in figure 3.30

MODEL	shank diameter 1	shank diameter 2	shank diameter 3	head diameter 1	head diameter 2	head diameter 3	rivet head height	diameter_hole 1	diameter_hole 2	diameter_hole 3	thickness_1	thickness_2	offset	edge_distance	pitch	pret_force_1	pret_force_2	pret_force_3	trac_force	mesh_size	converger
1	4.8	4.8	4.8	8.6	8.6	8.6	3.0	4.8	4.8	4.8	3.0	3.0	11.2	12.0	22.0	1000.0	1000.0	1000.0	1000.0	0.9	YES
2	4.8	4.8	4.8	8.6	8.6	8.6	3.0	5.4	5.4	5.4	3.0	3.0	11.2	12.0	22.0	1000.0	1000.0	1000.0	1000.0	0.9	YES
3	4.8	4.8	4.8	8.6	8.6	8.6	3.0	4.8	4.8	4.8	3.0	3.5	11.2	12.0	22.0	1000.0	1000.0	1000.0	1000.0	0.9	YES
4	4.8	4.8	4.8	8.6	8.6	8.6	3.0	4.8	4.8	4.8	3.5	3.0	11.2	12.0	22.0	1000.0	1000.0	1000.0	1000.0	0.9	YES
5	4.8	4.8	4.8	8.6	8.6	8.6	3.0	4.9	4.9	4.9	3.1	3.0	11.2	12.0	22.0	1000.0	1000.0	1000.0	1000.0	0.9	YES
6	5.2	4.8	4.8	8.6	8.6	8.6	3.0	5.2	4.8	4.8	3.0	3.0	11.2	12.0	22.0	1000.0	1000.0	1000.0	1000.0	0.9	YES
7	4.8	5.2	4.8	8.6	8.6	8.6	3.0	4.8	5.2	4.8	3.0	3.0	11.2	12.0	22.0	1000.0	1000.0	1000.0	1000.0	0.9	YES
8	4.8	4.8	5.2	8.6	8.6	8.6	3.0	4.8	4.8	5.2	3.0	3.0	11.2	12.0	22.0	1000.0	1000.0	1000.0	1000.0	0.9	YES
9	5.2	4.8	4.8	8.6	8.6	8.6	3.0	5.4	5.0	5.0	3.0	3.0	11.2	12.0	22.0	1000.0	1000.0	1000.0	1000.0	0.9	YES
10	4.8	5.2	4.8	8.6	8.6	8.6	3.0	5.0	5.4	5.0	3.0	3.0	11.2	12.0	22.0	1000.0	1000.0	1000.0	1000.0	0.9	YES
11	4.8	4.8	5.2	8.6	8.6	8.6	3.0	5.0	5.0	5.4	3.0	3.0	11.2	12.0	22.0	1000.0	1000.0	1000.0	1000.0	0.9	YES
12	5.2	4.8	4.8	8.6	8.6	8.6	3.0	5.2	4.8	4.8	3.2	3.0	11.2	12.0	22.0	1000.0	1000.0	1000.0	1000.0	0.9	YES
13	5.2	4.8	4.8	8.6	8.6	8.6	3.0	5.4	5.0	5.0	3.0	3.2	11.2	12.0	22.0	1000.0	1000.0	1000.0	1000.0	0.9	YES
14	4.8	4.8	4.8	8.6	8.6	8.6	3.0	4.8	4.8	4.8	3.0	3.0	13.0	12.0	22.0	1000.0	1000.0	1000.0	1000.0	0.9	YES
15	5.2	4.8	4.8	8.6	8.6	8.6	3.0	5.4	5.0	5.0	3.0	3.2	13.0	12.0	22.0	1000.0	1000.0	1000.0	1000.0	0.9	YES

Figure 3.30: DOE chart

Thus, the next task was to change the python script in the way it could read the table created in Excel and consequently run the file generator function according to the values defined in the excel. To do so, the panda library was imported to make Python able to read the excel saved in the same folder.

Design variable values (mm)									
Model	dr_{s1}	dr_{s2}	dr_{s3}	d_{h1}	d_{h2}	d_{h3}	t_1	t_2	o_x
1	4.8	4.8	4.8	4.8	4.8	4.8	3.0	3.0	11.2
2	4.8	4.8	4.8	5.4	5.4	5.4	3.0	3.0	11.2
3	4.8	4.8	4.8	4.8	4.8	4.8	3.0	3.5	11.2
4	4.8	4.8	4.8	4.8	4.8	4.8	3.5	3.0	11.2
5	4.8	4.8	4.8	4.9	4.9	4.9	3.1	3.0	11.2
6	5.2	4.8	4.8	5.2	4.8	4.8	3.0	3.0	11.2
7	4.8	5.2	4.8	4.8	5.2	4.8	3.0	3.0	11.2
8	4.8	4.8	5.2	4.8	4.8	5.2	3.0	3.0	11.2
9	5.2	4.8	4.8	5.4	5.0	5.0	3.0	3.0	11.2
10	4.8	5.2	4.8	5.0	5.4	5.0	3.0	3.0	11.2
11	4.8	4.8	5.2	5.0	5.0	5.4	3.0	3.0	11.2
12	5.2	4.8	4.8	5.2	4.8	4.8	3.2	3.0	11.2
13	5.2	4.8	4.8	5.4	5.0	5.0	3.0	3.2	11.2
14	4.8	4.8	4.8	4.8	4.8	4.8	3.0	3.0	13.0
15	5.2	4.8	4.8	5.4	5.0	5.0	3.0	3.2	13.0

Table 3.2: Design of experiment

The Python command structure used to read the data contained in the excel was the one shown in figure 3.31:

```
import pandas as pd

cases= pd.read_excel('Cases.xlsx')
```

Figure 3.31: Code used to introduced the data as a table into Python

Once the table was read, all the work left was to assign each of the columns with its corresponding variable, so the previously developed function could be run inside the loop which covers the entire domain of analysis, performing 15 iterations, which corresponds to the number of models to be analyzed. Hence, the code used was a *for* loop described as per figure 3.32

```
models= cases.iloc[:,0]
for iter in models:
    #diameter of rivet 1
    dr_s_1= cases.iloc[iter-1,1]
    #diameter of rivet 2
    dr_s_2= cases.iloc[iter-1,2]
    #diameter of rivet 3
    dr_s_3= cases.iloc[iter-1,3]
    #diameter from the head of rivet 1
    dr_h_1= cases.iloc[iter-1,4]
    #diameter from the head of rivet 2
    dr_h_2= cases.iloc[iter-1,5]
    #diameter from the head of rivet 3
    dr_h_3= cases.iloc[iter-1,6]
    #rivet head height
    hr= cases.iloc[iter-1,7]
    #diameter from hole 1
    d_h_1= cases.iloc[iter-1,8]
    #diameter from hole 2
    d_h_2= cases.iloc[iter-1,9]
    #diameter from hole 3
    d_h_3= cases.iloc[iter-1,10]
    #upper plate thickness
    t_1= cases.iloc[iter-1,11]
    #lower plate thickness
    t_2= cases.iloc[iter-1,12]
    #offset between plates in x-coordinate
    o_x= cases.iloc[iter-1,13]
    #edge distance
    e= cases.iloc[iter-1,14]
    #hole pitch
    p= cases.iloc[iter-1,15]
    #pretension force for rivet 1
    f_in= cases.iloc[iter-1,16]
    # pretension force for rivet 2
    f_cent= cases.iloc[iter-1,17]
    # pretension force for rivet 3
    f_out= cases.iloc[iter-1,18]
    #traction force
    f_trac= cases.iloc[iter-1,19]
    #mesh size
    m_size= cases.iloc[iter-1,20]


    RpyFileCreator(dr_s_1, dr_s_2, dr_s_3, dr_h_1, dr_h_2,
dr_h_3, hr, d_h_1, d_h_2, d_h_3, t_1, t_2, o_x, e, p, f_in,
f_cent, f_out, f_trac, m_size, iter)
```

Figure 3.32: Code used to create the files with the already read data

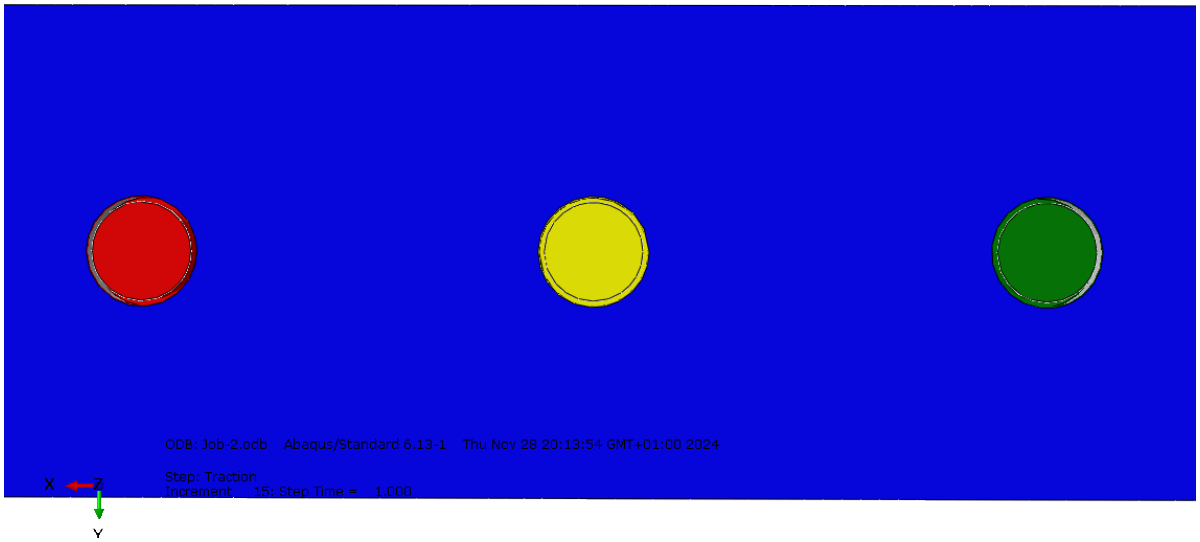
Then after all the files were created, they only needed to be run in order to observe the different results.

In order to be coherent with the trial model, the colours used for each of the instances conforming the model will be the same as before. In order to recall them, the colour legend for these 15 models is the following one:

- The **closest rivet** to the clamped section will be represented in **green**.
- The **middle rivet** will be represented in **yellow**.
- The **furthest rivet** to the clamped section will be represented in **red**.
- **Top plate**, corresponding to the one clamped, will be shown in **blue**.
- **Bottom plate**, where traction force is applied, will be represented in **white**.

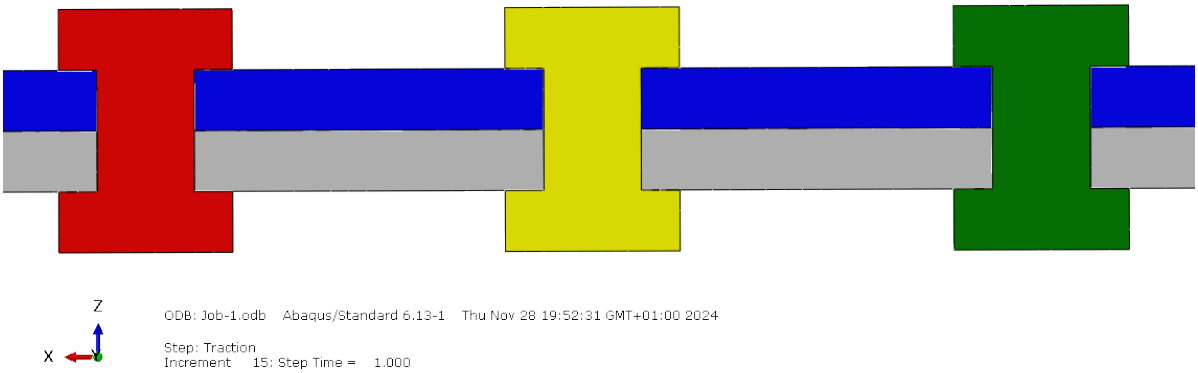
Starting with the first two models, according to table 3.30, these two should be identical but the fact that the holes of model two have clearance, so this gap should visible in the model before the calculation starts. We can check the model generator worked correctly as figures 3.33 and 3.34 show the gap from two different perspectives.

(a) Upper view from model 1 (without clearance)

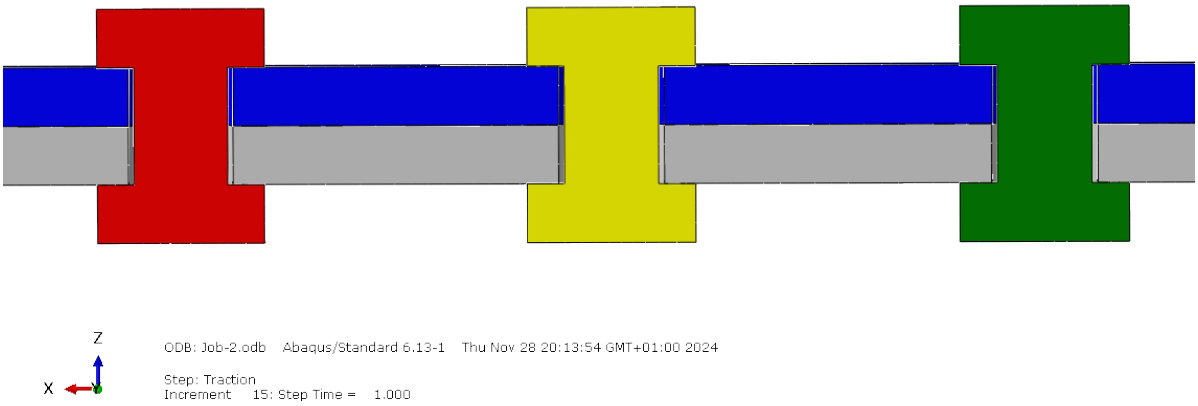


(b) Upper view from model 2 (with clearance)

Figure 3.33: Differences between models



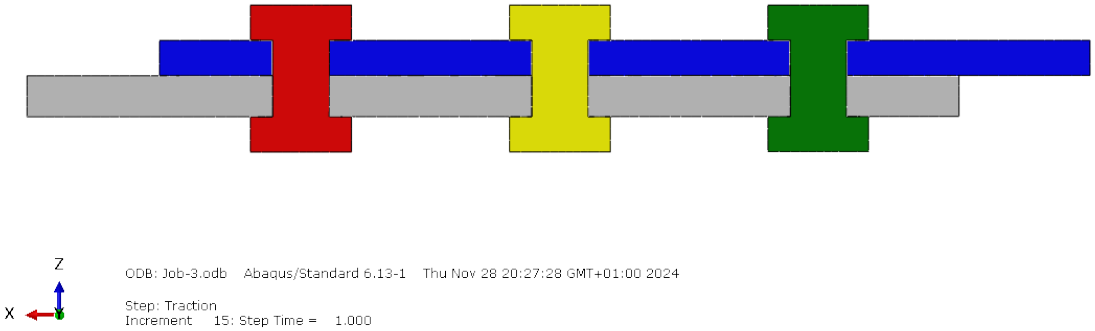
(a) Side cut from model 1 (without clearance)



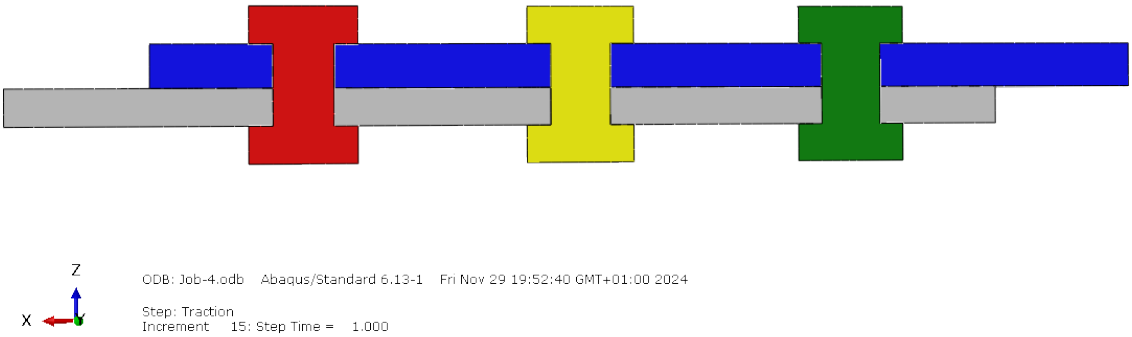
(b) Side cut from model 2 (with clearance)

Figure 3.34: Differences between models

Model 3 and 4 are characterized by having two plates with different thicknesses, but neither of them have clearance in each of the three holes. This can be observed in figure 3.35, so once again, we can observe that the model generator is working correctly in both cases where we define a clearance in the holes and when defining a different thickness for the plates. Actually, one of the main issues that could happen dealing with the second item is that some overlaps appear due to some incorrect measurements which are thickness dependant. This could be the case of the shank length, and as seen in the figure 3.35 there exists no overlap between the different instances.



(a) Side cut from model 3 ($t_2 > t_1$)



(b) Side cut from model 4 ($t_1 > t_2$)

Figure 3.35: Thickness difference between model 3 and 4, both without clearance

Another aspect that can be checked is the shank and hole radius adjustment when these are different in one position with respect to the other two rivets. This will be the case for most of the models starting from model 5. And once again, we can check for example model 7 and see that both the hole and the shank diameter from middle position is larger than the other two. This could be checked by measuring directly in the software. However, this difference is to small to be visible in a picture, and that is why there is no picture shown like the ones presented previously.

The same happens with the last two models where the offset is incremented by almost two millimetres. The measurements can be checked directly in the software but the differences are so tiny that are not observable in a picture. Nevertheless, the model generation operation could be checked successfully, so once this aspect was verified we could jump directly to extract the results.

Being more precise, the result aimed to obtain is the shear load transferred by each of the rivets. In order to do so, there exist plenty of options. One way to extract this shear load is to obtain the nodal forces on the surface of the rivet, and perform and integral around it to obtain the overall shear force transferred. This way

of proceeding entails exporting all nodal forces which is expensive if we consider we have to do this operation for the total of 45 rivets contained in the 15 models proposed.

Instead, by using structural analysis knowledge, these values can be extracted in an easier way. Recalling the interactions defined in figures 3.8 and 3.9, lower plate will be the master surface when contacting with the rivet shank, whereas for the upper plate, the master surface will be instead the rivet shank since it is the element which initiates the contact.

Having this into consideration, we will thus perform a similar to beam analysis. It must be also taken into account that the model is calculated using an implicit solver with static steps. This implies that, since no inertia nor damping play action into this, an equilibrium must be always be fulfilled. Therefore as it were a beam analysis, the model will be cut into several sections where equilibrium needs to be fulfilled as stated before.

This equilibrium analysis will be only performed with forces in the x-direction, since this is the item and direction where relevant information is contained. Doing so, we can draw all the forces presented in the model, arriving to figure 3.36. The nomenclature used to name the reaction forces is naming firstly the element which applies or transfers the force, followed by the element which receives that force. Thus, the reaction forces that can be found are:

- F_{r1p1}: Force exerted by the inner rivet to the upper plate.
- F_{r2p1}: Force exerted by the center rivet to the upper plate.
- F_{r3p1}: Force exerted by the outer rivet to the upper plate.
- F_{r1p2}: Force exerted by the inner rivet to the lower plate.
- F_{r2p2}: Force exerted by the center rivet to the lower plate.
- F_{r3p2}: Force exerted by the outer rivet to the lower plate.
- F_{reac}: Reaction force exerted by the clamped support to the upper plate

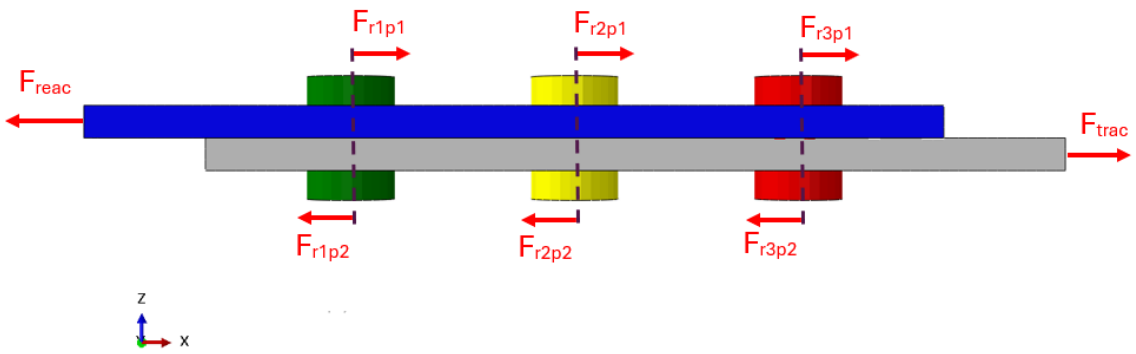


Figure 3.36: Model with external and reaction forces

Therefore, it seems that from all the eight forces appearing in the model, we only know one of them (F_{trac} which is equal to 1000 Newtons) and the rest of them need to be found. Nonetheless, it has not been yet applied the equilibrium conditions in the rivets, which are also elements of the model. Since each of the rivets transfer two opposing forces, and no energy is created in the rivet since no external forces are applied in each of them, we can arrive to the following conclusions:

$$F_{r1p1} = F_{r1p2} \tag{3.12}$$

$$F_{r2p1} = F_{r2p2} \tag{3.13}$$

$$F_{r3p1} = F_{r3p2} \tag{3.14}$$

Of course these equations are module related, because from figure 3.36 it is trivial that vectors do not point in the same direction, and that is the reason why none of them have an arrow on it. Therefore, with these relations we have reduced the unknown variables from 7 to 4. Although this number can be reduced even more since, when applying the overall equilibrium condition in the x-axis, all the reaction forces in the rivets are vanished, and so we are left with the condition stated hereafter:

$$F_{reac} = F_{trac} = 1000N \tag{3.15}$$

Therefore, from here we can already make the following statements:

- Rivets are the only means of transfer load on this direction between the plates, as expected.
- There are three unknowns in this equilibrium analysis and those are the shear loads corresponding to each of the three rivets.

Of course, in this analysis there is a missing force which is the friction force between the plates. However, this magnitude is totally negligible compared to the other five values given the fact that the movement between the plates is almost null.

Hence, after this explanation, let's proceed with the way these shear loads can be extracted. We will take the upper plate, and perform free body cuts in order to check which is the force in each of these instances of interest. Consequently, only the forces applied to the upper plate will be involved. Since there are four elements which applies a force to the upper plate (three rivets and the clamped support), there are four different regions identified, which are the ones shown in figure 3.37.

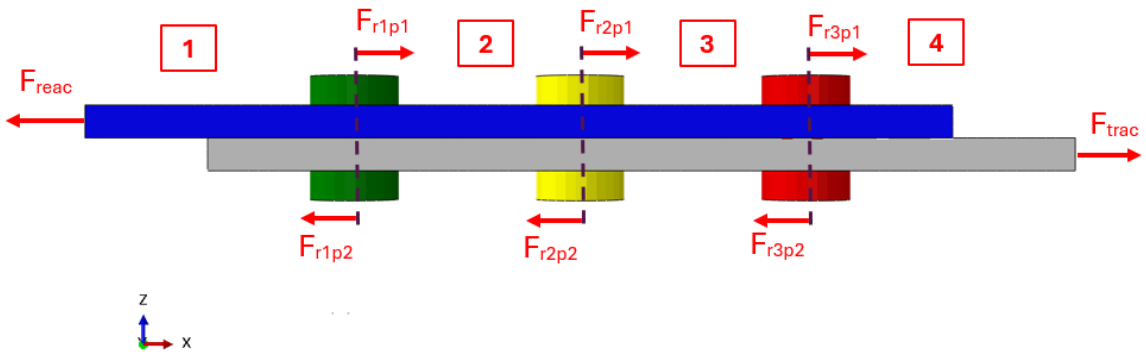


Figure 3.37: Regions where cross section force will be analyzed

These sections are divided by each of the elements which carry loads, which means the internal force will be the same in every section contained within the same region. In order to identify each of the following sections, it will be used the nomenclature $F_{sec(i)}$ being "i" the number of the region where analysis is performed. Starting with the region 1, which is the closest one to the clamped support, the following equilibrium diagram can be drawn, represented in figure 3.38.

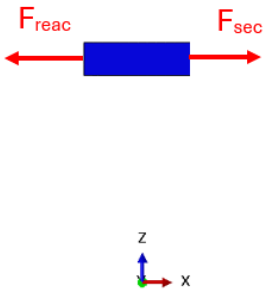


Figure 3.38: Free-body cut in region 1

For this first region, the equation is trivial, since only one support is applying a force to the plate, so the equation for region 1 is:

$$F_{sec(1)} = F_{reac} = 1000N \tag{3.16}$$

When jumping to the second region, the inner rivet already applies a certain load to the plate, so the free-body cut would be represented as in figure 3.39.

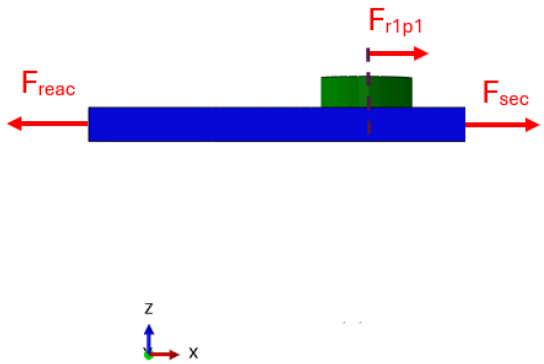


Figure 3.39: Free-body cut in region 2

which leads to the following equilibrium equation for forces in the x-axis:

$$F_{sec(2)} + F_{r1p1} = F_{reac} = 1000N \tag{3.17}$$

Therefore, as expected the force carries by the plate will go down as the x-coordinate moves towards the free edge. Something similar happens with the third region, but this time there are two rivets transferring load, which leads to the diagram represented in figure 3.40.

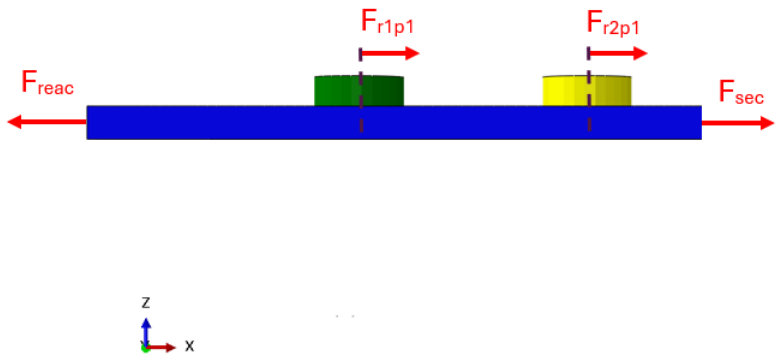


Figure 3.40: Free-body cut in region 3

In turn the equilibrium equation for forces in the x-axis can be written as follows:

$$F_{sec(3)} + F_{r1p1} + F_{r2p1} = F_{reac} = 1000N \tag{3.18}$$

Lastly, in region 4 all the fasteners have already carried some load to the upper plate, as seen in figure 3.41.

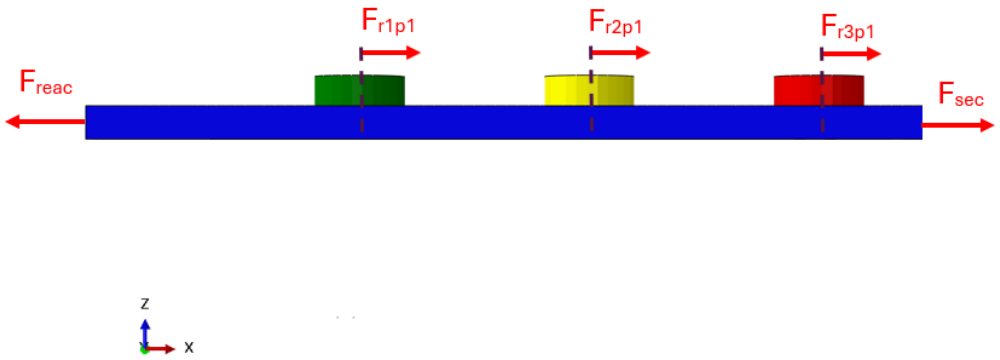


Figure 3.41: Free-body cut in region 4

Proceeding as in previous sections, the equation which comes up is the following:

$$F_{sec(4)} + F_{r1p1} + F_{r2p1} + F_{r3p1} = F_{reac} = 1000N \tag{3.19}$$

However, since there is not and end support in the upper plate but there is a free end, the boundary condition is that the load carried by the upper plate in that region is null. Considering this, the final equilibrium equation in region 4 would be equation 3.20.

$$F_{r1p1} + F_{r2p1} + F_{r3p1} = F_{reac} = 1000N \tag{3.20}$$

As expected the load carried by the plate will be reduced from the 1000 newtons at the clamped section to the zero charge as we increase the x-coordinate. But what is this useful for then?

This analysis is extremely important since from out Abaqus model the cross-section forces of the plate ($F_{sec(i)}$) can be extracted easily. This can be done by using the free-body cut tool available in the visualization module. Remember the objective was to retrieve the shear loads of the rivets, which at the end of the day are the F_{ripj} loads. If we are able to extract the section forces with the free-body cut tool, using equations 3.17, 3.18 and 3.20, these shear loads can be easily found.

Actually from these equations we can find the three actual relations which will provide the shear loads. From equation 3.17 we can state the following:

$$F_{r1p1} = 1000 - F_{sec(2)} \tag{3.21}$$

Likewise, from equations 3.18 and 3.20 the other shear loads can be defined as:

$$F_{r2p1} = 1000 - F_{sec(3)} - F_{r1p1} = F_{sec(2)} - F_{sec(3)} \tag{3.22}$$

$$F_{r3p1} = 1000 - F_{r1p1} - F_{r2p1} = F_{sec(3)} \tag{3.23}$$

Starting from model 1 (whose properties can be reviewed in table 3.2), it can be performed the obtention of these two necessary values, which are $F_{sec(2)}$ and $F_{sec(3)}$.

Checking the first region, as it was expected, the force carried by the plate is the same as the external force, as seen in figure 3.42.

Figure 3.42: Free-body cut for model 1 in region 1

Jumping to the second region, past the inner rivet, we can get a value of 327 N as seen in figure 3.43.

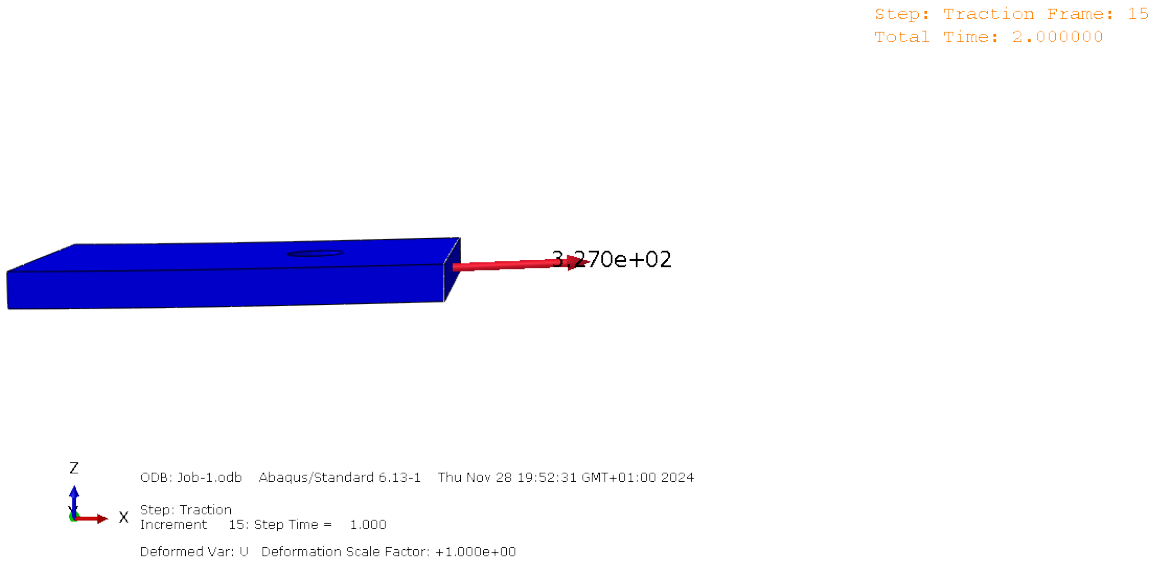


Figure 3.43: Free-body cut for model 1 in region 2

This implies thus that for model 1, the inner rivet shear load can be found:

$$F_{r1p1} = 1000 - F_{sec(2)} = 1000 - 327 = 673N \tag{3.24}$$

This is more than half of the load introduced in the model. Obtaining this value is coherent with the test model performed at the beginning of this chapter, where it could be seen that most of the stress was concentrated in the inner rivet (figure 3.16).

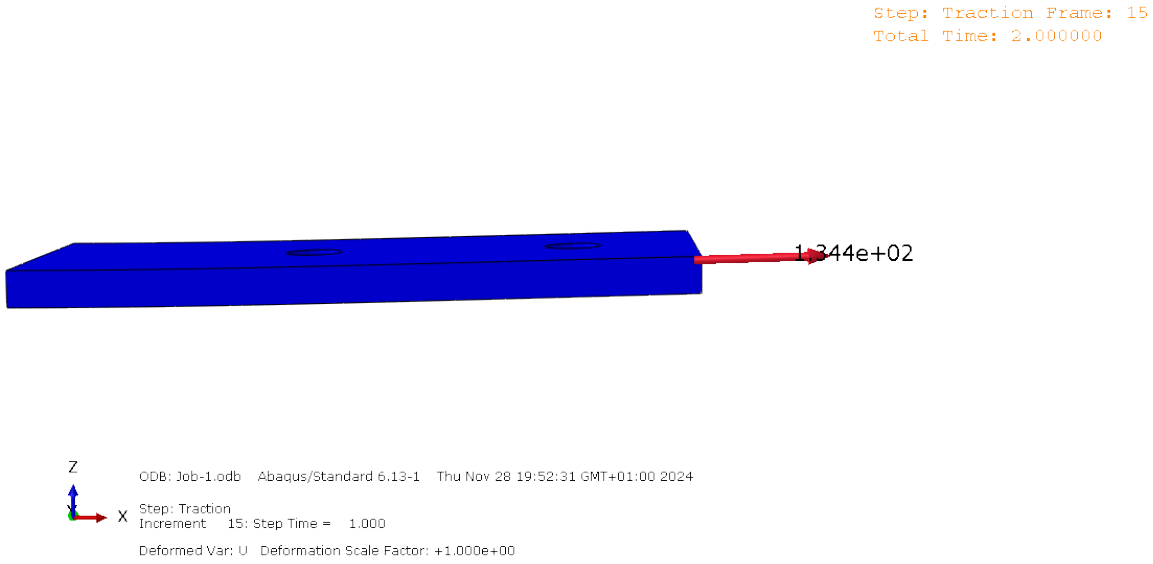


Figure 3.44: Free-body cut for model 1 in region 3

Jumping to the third region, the value for the force carried by the plate is 134.4 newtons. With this value it can be thus found the remaining shear loads:

$$F_{r2p1} = 1000 - F_{sec(3)} - F_{r1p1} = F_{sec(2)} - F_{sec(3)} = 327 - 134.4 = 192.6N \tag{3.25}$$

$$F_{r3p1} = 1000 - F_{r1p1} - F_{r2p1} = F_{sec(3)} = 134.4N \tag{3.26}$$

As expected, the other two rivets are not even close as loaded as the inner rivet is, which once again is coherent with results. In order to check the approach used is valid, the free-body cut for region 4 of model 1 is presented. The value extracted from the software is 0.07083 N, which is a very small value. This value is not zero because of the previously mentioned friction that exists between the mating plates. Nonetheless, it can be seen that our previous statement was correct since this value is four orders of magnitude lower compared to the transferring loads of the rivets. Therefore, this method of obtaining the rivet shear loads has been verified correctly and all boundary and equilibrium conditions were satisfied.

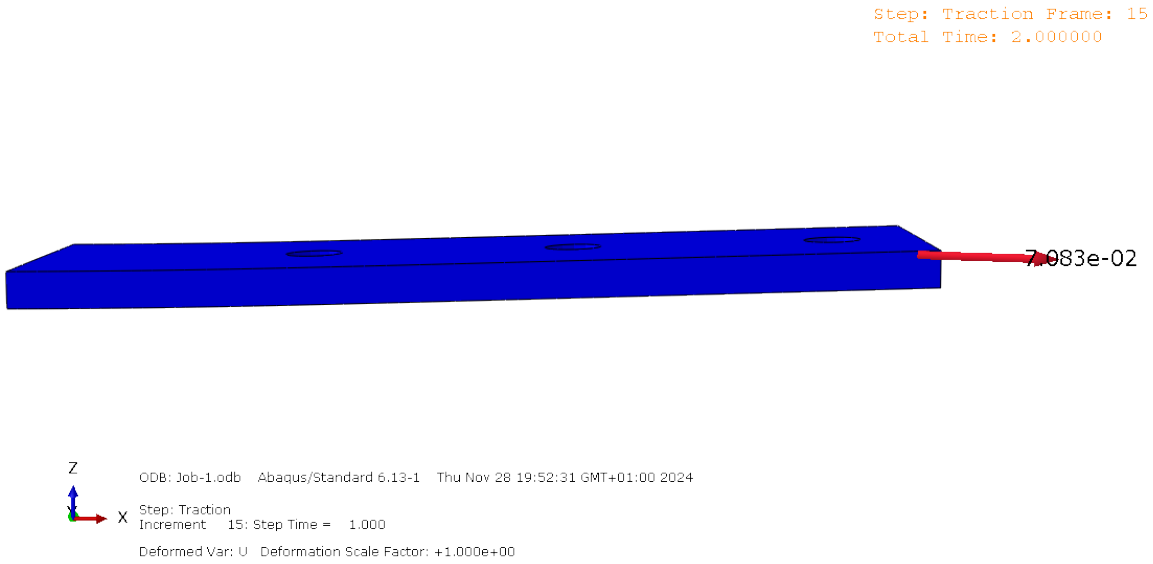


Figure 3.45: Free-body cut for model 1 in region 4

Notice that we could have performed the same procedure by picking the lower plate. Doing so, the same analysis would have been done but instead picking the forces which act in the lower plate. These forces are F_{r1p2}, F_{r2p2} and F_{r3p2}. Repeating the same body cuts shown in figures 3.38, 3.39, 3.40 and 3.41 but for the lower part of the assembly the rivet shear loading equations can be expressed as:

$$F_{r1p2} = F_{sec(2)}* \tag{3.27}$$

$$F_{r2p2} = F_{sec(3)}* - F_{sec(2)}* \tag{3.28}$$

$$F_{r3p2} = 1000 - F_{sec(3)}* \tag{3.29}$$

For this equations, since they are referred to the lower plate, the section forces are followed by the symbol (*). The beam cut exercise required to obtain this equation would be started from the free end, where the boundary condition and force is already known. But apart from that difference, the basis is the same.

Therefore, for this first model, we should find the same values using whether the upper or the lower plate. Making a free body cut in the second section, for the lower plate we obtain the force shown in figure 3.46

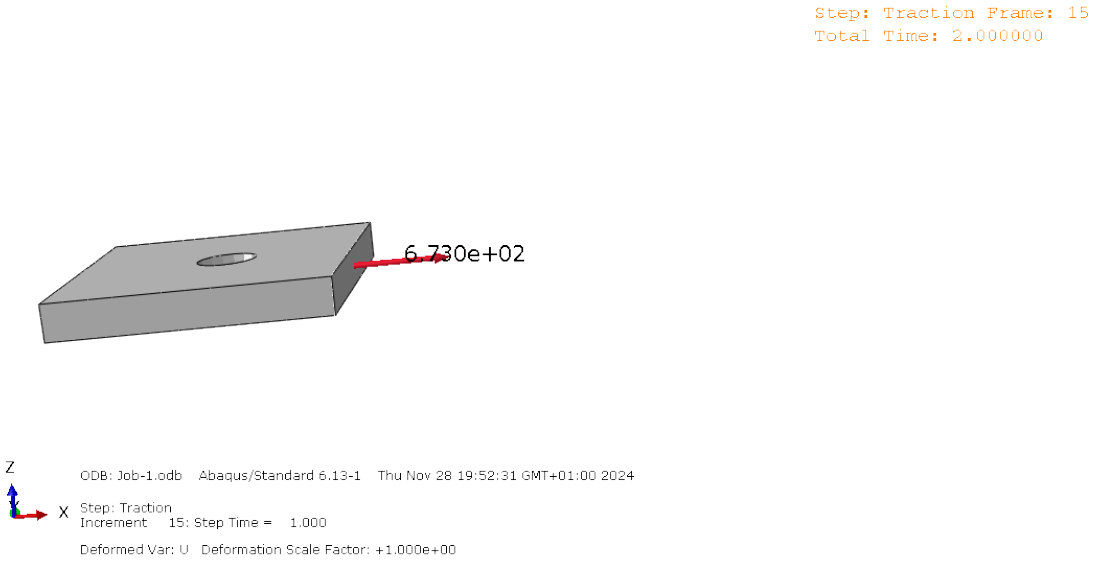


Figure 3.46: Free-body cut for model 1 in region 2 for the lower plate

Thus recalling to the equation 3.27:

$$F_{r1p2} = F_{sec(2)}* = 673N \tag{3.30}$$

The second body cut made in the lower plate, performed this time in section 3, reveals a resultant force of 865.6 newtons, as per figure 3.47:

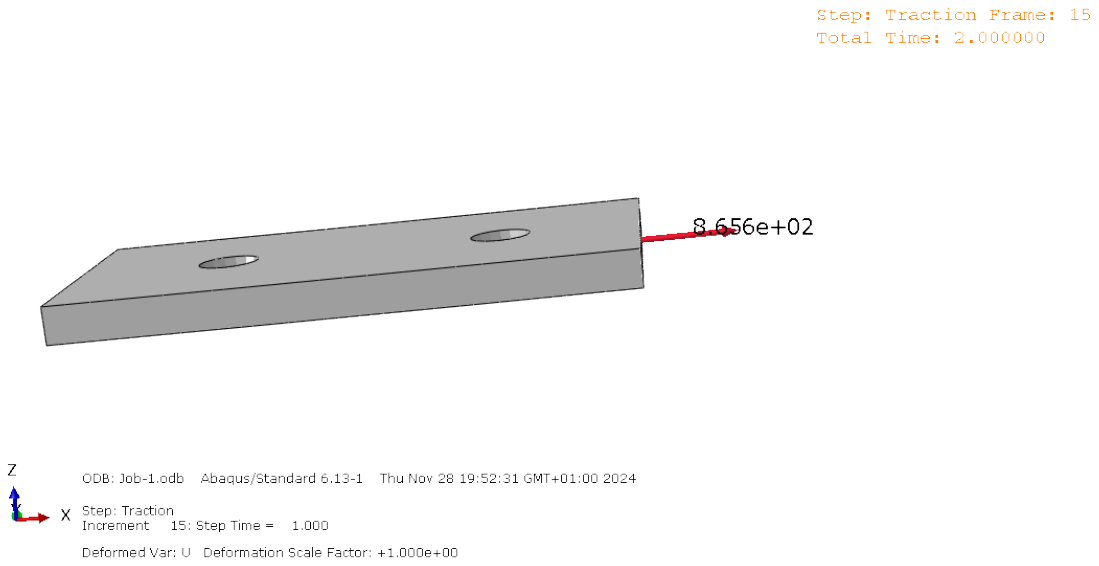


Figure 3.47: Free-body cut for model 1 in region 3 for the lower plate

This force can be introduced in equations 3.28 and 3.29 to come up with the following resultant shear forces:

$$F_{r2p2} = F_{sec(3)} * -F_{sec(2)} * = 865.6 - 673 = 192.6N \tag{3.31}$$

$$F_{r3p2} = 1000 - F_{sec(3)} * = 1000 - 865.6 = 134.4N \tag{3.32}$$

As expected the results are the same following both approximations, which once again proves this way of proceeding is correct. This exercise shown can be applied for each of the 15 models generated, and therefore the objective is to collect all these shear forces and group them in a table to check what are the results and the influence of each of the parameters in these shear force study.

In order to present the results of the analysis, the following equalities will be made:

$$F_{r1p1} = F_{r1p2} = Fshear_{inner} \tag{3.33}$$

$$F_{r2p1} = F_{r2p2} = Fshear_{center} \tag{3.34}$$

$$F_{r3p1} = F_{r3p2} = Fshear_{outer} \tag{3.35}$$

Hence, performing the same exercise with all the 15 models, a table can be made in order to sum up all the shear loads.

Rivet Transfer Shear Load (N)			
Model	$Fshear_{inner}$	$Fshear_{center}$	$Fshear_{outer}$
1	673.0	192.6	134.4
2	527.5	310.8	161.7
3	661.8	203.0	135.2
4	655.3	197.1	147.6
5	553.8	288.3	157.9
6	658.3	205.1	136.6
7	580.9	269.1	150.0
8	648.5	211.6	139.9
9	533.2	305.5	161.3
10	535.0	305.7	159.3
11	533.4	303.6	163.0
12	655.9	202.2	141.9
13	525.2	310.4	164.4
14	679.8	188.0	132.2
15	527.7	307.6	164.7

Table 3.3: Results for the shear load of each model

3.4. FEM vs Analytic model results

Once all the results are summarized in table 3.3, we can go back to section 3.1 where the analytic shear distribution formulas are presented for a cluster of rivets. Recalling this section, the shear load of a certain rivet could be expressed as:

$$P_i = P_{s,i} + P_{m,i} \tag{3.36}$$

where

$$P_{s,i} = P \cdot \left(\frac{A_i}{\sum_{j=1}^{n} A_j} \right) \tag{3.37}$$

and

$$P_{m,i} = M \cdot \left(\frac{A_i \cdot d_i}{\sum_{j=1}^{n} A_j \cdot d_j^2} \right) \tag{3.38}$$

However, as mentioned in section 3.1, in the case of this project, $P_{m,i}$ is null since the line of the external force coincides with the one joining the centers of the fasteners. As well, the model does not account for fastener clearance, so the models with this phenomenon will be left out of this analysis. Thus, with this equation 3.37 it can be found the fastener shear load distribution for models 1, 3, 4, 6, 7, 8, 12 and 14.

In table 3.2 it can be found the values of the diameters for all these eight models, therefore it is trivial to find the shear load distribution for each of the cases. For instance, taking model 1 as an example, we can see that:

$$P_1 = P_{s,1} = \frac{P \cdot A_1}{A_1 + A_2 + A_3} = \frac{1000 \cdot \pi \cdot 2.4^2}{3 \cdot \pi \cdot 2.4^2} = \frac{1000}{3} = 333.3N \tag{3.39}$$

And as A_1 is the same as A_2 and A_3 for model 1, it can be seen that:

$$P_1 = P_2 = P_3 \tag{3.40}$$

following the nomenclature explained hereafter:

- P_1 accounts for the shear load transferred by the inner rivet, the closest to the clamped section.
- P_2 accounts for the shear load transferred by the center rivet.
- P_3 accounts for the shear load transferred by the outer rivet, the furthest to the clamped section.

If we repeat the same procedure for the other seven models without clearance, the following results can be obtained, summarized in table 3.4.

Rivet Transfer Shear Load (N)			
Model	$Fshear_{inner}$	$Fshear_{center}$	$Fshear_{outer}$
1	333.3	333.3	333.3
3	333.3	333.3	333.3
4	333.3	333.3	333.3
6	391.2	304.4	304.4
7	304.4	391.2	304.4
8	304.4	304.4	391.2
12	391.2	304.4	304.4
14	333.3	333.3	333.3

Table 3.4: Results obtained following analytic solution approach

In order to make it clearer, the following tables show the results comparing both FEM and analytic approach for all the eight models.

Results for model 1 (N)			
Method	$Fshear_{inner}$	$Fshear_{center}$	$Fshear_{outer}$
Analytic solution	333.3	333.3	333.3
FEM solution	673.0	192.6	134.4

Table 3.5: Results comparison for model 1

Results for model 3 (N)			
Method	$Fshear_{inner}$	$Fshear_{center}$	$Fshear_{outer}$
Analytic solution	333.3	333.3	333.3
FEM solution	661.8	203.0	135.2

Table 3.6: Results comparison for model 3

Results for model 4 (N)			
Method	$Fshear_{inner}$	$Fshear_{center}$	$Fshear_{outer}$
Analytic solution	333.3	333.3	333.3
FEM solution	655.3	197.1	147.6

Table 3.7: Results comparison for model 4

Results for model 6 (N)			
Method	$Fshear_{inner}$	$Fshear_{center}$	$Fshear_{outer}$
Analytic solution	391.2	304.4	304.4
FEM solution	658.3	205.1	136.6

Table 3.8: Results comparison for model 6

Results for model 7 (N)			
Method	$Fshear_{inner}$	$Fshear_{center}$	$Fshear_{outer}$
Analytic solution	304.4	391.2	304.4
FEM solution	580.9	269.1	150.0

Table 3.9: Results comparison for model 7

Results for model 8 (N)			
Method	$Fshear_{inner}$	$Fshear_{center}$	$Fshear_{outer}$
Analytic solution	304.4	304.4	391.2
FEM solution	648.5	211.6	139.9

Table 3.10: Results comparison for model 8

Results for model 12 (N)			
Method	$Fshear_{inner}$	$Fshear_{center}$	$Fshear_{outer}$
Analytic solution	391.2	304.4	304.4
FEM solution	655.9	202.2	141.9

Table 3.11: Results comparison for model 12

Results for model 14 (N)			
Method	$Fshear_{inner}$	$Fshear_{center}$	$Fshear_{outer}$
Analytic solution	333.3	333.3	333.3
FEM solution	679.8	188.0	132.2

Table 3.12: Results comparison for model 14

As it can be seen, for all the models, the differences between FEM and analytic solutions are quite large. As anticipated in section 3.1, the analytic approach does not consider the out of plane effects, which in this case play a huge role. This occurs because from the analytic solution perspective, no external momentum is exerted to the fasteners since the line of the external force goes through the cluster c.g. Nonetheless, when we analyze the problem in 3D, this does not occur since the cluster c.g. is no longer an axis but a point, which in this case is not contained in the plane where the external force is applied.

Consequently, as anticipated, there will be a momentum making the plates to rotate. Therefore, this is one of the reasons why this project is performed using FEA, since if done analytically the results would not be accurate because this external momentum contribution is very difficult to be calculated by hand. By intuition, looking at the assembly, it could be anticipated that the inner fastener would be more loaded than the rest, but this increment, as mentioned before, it is very difficult to calculate.

To sum up, as expected, the analytic approach does not match the FEM results since it does not take into account 3D effects. Notice that only models without clearance have been considered in this section and even these ones which are easier to analyze do not comply with the required accuracy using an analytic approach. This is used once again as an argument to support the relevance of these 3D models, since from them a lot of important information can be retrieved and further used to extract useful conclusions which could not be found using an analytic solution.

4

Results discussion and conclusion

After getting the results from the 15 models analyzed, the aim of this fourth chapter is to explain the factors that supports these results and to verify if the selected design variables have had an effect in these ones.

The effects pretended to be analyzed can be summarized in the listed points hereafter:

- **Hole clearance**: making the hole diameter larger than the one of the fastener inserted in that position.
- **Diameter size**: making the diameter of one or more fasteners larger.
- **Plate thickness**: varying one of both thickness to shift the pretension plane location.
- **Offset between plates**: incrementing the distance between the first rivet and the clamped section.

Notice that these effects may appear isolated or in the presence of more of the listed phenomena. Therefore, in order to quickly check which element appears in each model, the chapter starts with a table (4.1) which exclusively remarks the values which are deviated from the nominal model, considered to be the first one. Once this is made is easier to check which models need to be compared to evaluate the influence of each of the listed effects.

Starting with the hole clearance effect, which is the most important one inside this thesis, model 1 and 2 were compared since this gap between the rivets and the hole was the only difference between them. The results lead to the fact that this gap creation reduces the effective load transfer area of the rivet shank, being able to transfer less load. Therefore, the inner rivet, which is the one which will be more loaded, has a lower shear load value compared to the first model where no clearance appears. Consequently, by only comparing the first two models it was extracted that this clearance effect, as expected, reduces the shear load that can be carried. Therefore, failure occurs at less load.

Models 3 and 4 studied the influence of having two plates with different thicknesses. Each of them shifted the pretension plane upwards and to the bottom, respectively. The results showed that the stress distribution would vary, incrementing the stress concentration (with respect to the nominal case) in the rivet end (head or collar) closer to the pretension plane. Moreover, this deviation from the ideal condition where the pretension point is in half way of the shank makes the critical region to allow less load before failure. However, unlike the hole clearance effect, this one has much less influence in the overall rivet shear load distribution.

Indeed model 5 include both of these two mentioned phenomena, and it is observed that the behaviour corresponding to the hole clearance implementation accounts for almost the entire variation in shear load distribution with respect to the nominal model. This could be verified by looking at the contours presented.

Models 6-11 evaluate the effect of increasing the diameter in each of the three positions, as well as evaluating the combined effect of hole clearance and diameter sizing. By looking at models 6-8 where no gap is present, it could be checked the following points:

- When incrementing the inner rivet diameter, the circular crown area formed by the rivet head is reduced which leads into a higher concentration of stress. Therefore despite having a larger diameter which should allow the rivet to absorb more load, this rivet head effect counteracts this second one avoiding thus the shear load distribution to change considerably

- Unlike what happens for the inner rivet, the slope delta for of the plates in the second rivet is not high enough to make the stress concentrate in the head. Keeping the shank as the critical region, making its lateral surface larger we allow this second rivet to absorb more load compared to the nominal case.

Incrementing the diameter of the last rivet had a low influence in the result since this last fastener usually carries the least load since it is located the last in the row following the load transfer path between the plates.

However, once again, in models 9-11, where in presence of hole clearance, this diameter effect is completely vanished, since when incrementing the gap once again the shank region becomes more critical as the effective load transfer area reduces and, as it can be seen in table 3.3, the shear load distribution is almost the same for the three cases mentioned. Models 12 and 13 were useful to check if the thickness effect had relevance when the diameters of the rivets were different. However, it was checked that the difference between models 6 and 12, and models 9 and 13, respectively, were not sufficient to state the thickness variation has a relevant effect on the rivet shear load distribution.

Lastly, models 14 and 15 were created to evaluate the relevance of parameter o_x inside the model. When there is no clearance, the effect of increasing this parameters leads to a higher slope in the position of all the rivets. A higher slope means a stronger interaction between the head of the rivet and the plate, being now both the shank and the rivet the critical regions where failure can occur. Nonetheless, this effect did not vary that much the shear load distribution in the rivets either.

Once again, when introducing a gap between the rivets and the holes, the effect of incrementing this parameter o_x vanished, as the contours showed the same results for models 13 and 15, whose only differences were the values for this intended parameters.

After comparing all the models created in this project, the following general conclusion could be made:

- All the studied aspects had an impact in the stress distribution along the model profile.
- Among all of them, the hole clearance effect is the one which dominates, being able to reduce the allowable load that can be carried by more than 20% with respect to the case where no gap exists.
- Increasing the shank diameter will make the rivet absorb more load but as well it can make more vulnerable the region where rivet head and plate interacts. Depending on the fastener position and the amount of shank increment this can be either beneficial or harmful.
- Having different thicknesses will shift the pretension plane towards the head or the collar of the rivet, making thus stress to concentrate more in the are closer to this pretension plane.
- Plate offset increase the distance between the critical point ant the clamped region, promoting hence the rivet head - plate interaction and making both rivet head and upper part of the shank the critical regions where failure occurs.
- When all of these effects are present, hole clearance effect dominates and dictates model behaviour and both stress distribution and shear load distribution.
- This clearance will increase the stress in the region close to the edge where the fastener shank and the head are joint together.
- Since this last region is the most critical in terms of failure, clearance will make material to fail earlier.

4.1. Results analysis

In order to make a proper evaluation of the effect of each of the variables, it is important to recall table 3.2, where all the model properties are defined for each of the 15 cases. However, to make it simpler, model 1 is taken as reference and thus, table 4.1 will summarize the deviations with respect to this first model, so it will be straightforward to see what elements caused the difference in shear loads for each of the rivets.

The value "-" inside table 4.1 means that model parameter value coincides with the one of model 1, representing numerically only the values which are different.

Design variable values (mm)									
Model	dr_{s1}	dr_{s2}	dr_{s3}	d_{h1}	d_{h2}	d_{h3}	t_1	t_2	o_x
1	4.8	4.8	4.8	4.8	4.8	4.8	3.0	3.0	11.2
2	-	-	-	5.4	5.4	5.4	-	-	-
3	-	-	-	-	-	-	-	3.5	-
4	-	-	-	-	-	-	3.5	-	-
5	-	-	-	4.9	4.9	4.9	3.1	-	-
6	5.2	-	-	5.2	-	-	-	-	-
7	-	5.2	-	-	5.2	-	-	-	-
8	-	-	5.2	-	-	5.2	-	-	-
9	5.2	-	-	5.4	5.0	5.0	-	-	-
10	-	5.2	-	5.0	5.4	5.0	-	-	-
11	-	-	5.2	5.0	5.0	5.4	-	-	-
12	5.2	-	-	5.2	-	-	3.2	-	-
13	5.2	-	-	5.4	5.0	5.0	-	3.2	-
14	-	-	-	-	-	-	-	-	13.0
15	5.2	-	-	5.4	5.0	5.0	-	3.2	13.0

Table 4.1: Design of experiment

Recalling the meaning of each of the variables supported by table 4.3 and image 4.1, these variations with respect to the nominal model (model 1) can create the following phenomena:

- **Clearance effect**: Hole diameter is bigger than the rivet shank diameter.
- **Diameter effect**: Shank diameter of a certain rivet is larger than the other two.
- **Thickness effect**: Thickness of the plates are varied with respect to the original case.
- **Offset effect**: A value of o_x different from the one used in the nominal case is introduced

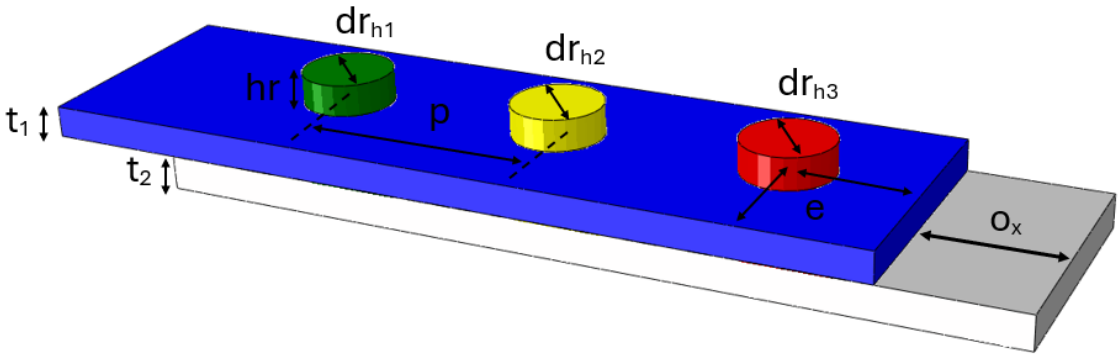
Therefore, we can determine which phenomenon will be present in each of the following models:

Phenomena appearing in the models				
Model	Clearance effect	Diameter effect	Thickness effect	Offset effect
1	No	No	No	No
2	Yes	No	No	No
3	No	No	Yes	No
4	No	No	Yes	No
5	Yes	No	Yes	No
6	No	Yes	No	No
7	No	Yes	No	No
8	No	Yes	No	No
9	Yes	Yes	No	No
10	Yes	Yes	No	No
11	Yes	Yes	No	No
12	No	Yes	Yes	No
13	Yes	Yes	Yes	No
14	No	No	No	Yes
15	Yes	Yes	Yes	Yes

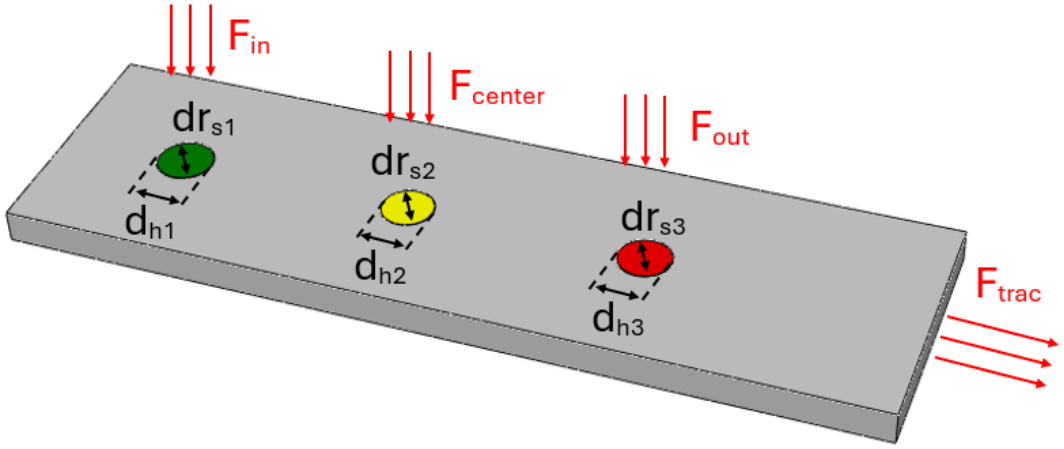
Table 4.2: Design of experiment

Design variables nomenclature	
Variable	Nomenclature
Inner rivet shank diameter	dr_{s1}
Center rivet shank diameter	dr_{s2}
Outer rivet shank diameter	dr_{s3}
Inner rivet head diameter	dr_{h1}
Center rivet head diameter	dr_{h2}
Outer rivet head diameter	dr_{h3}
Offset measurement in the x-direction	o_x

Table 4.3: Design variables considered in DOE



(a) Design parameters isometric view



(b) Design parameters middle cut

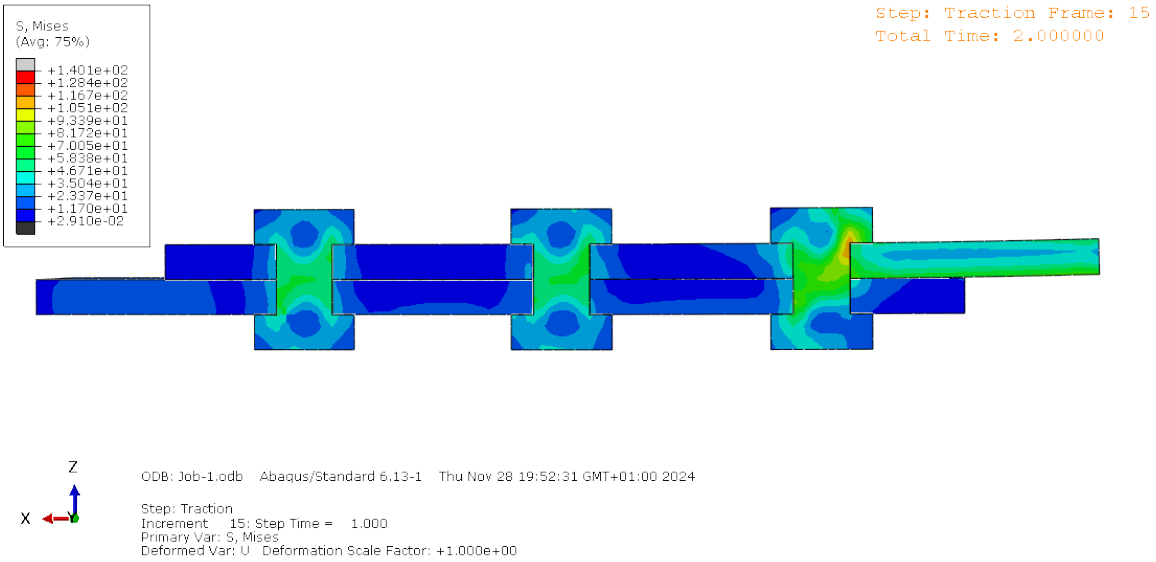
Figure 4.1: Design parameters represented in the model

By comparing the first two models, it is trivial to verify that the second model is the same as model 1 but leaving a hole clearance of 0.3 mm between the shank walls and the hole. Referring to table 3.3 it can be seen that for model 2 the highest shear force is reduced and hence the load is distributed more evenly between the three rivets.

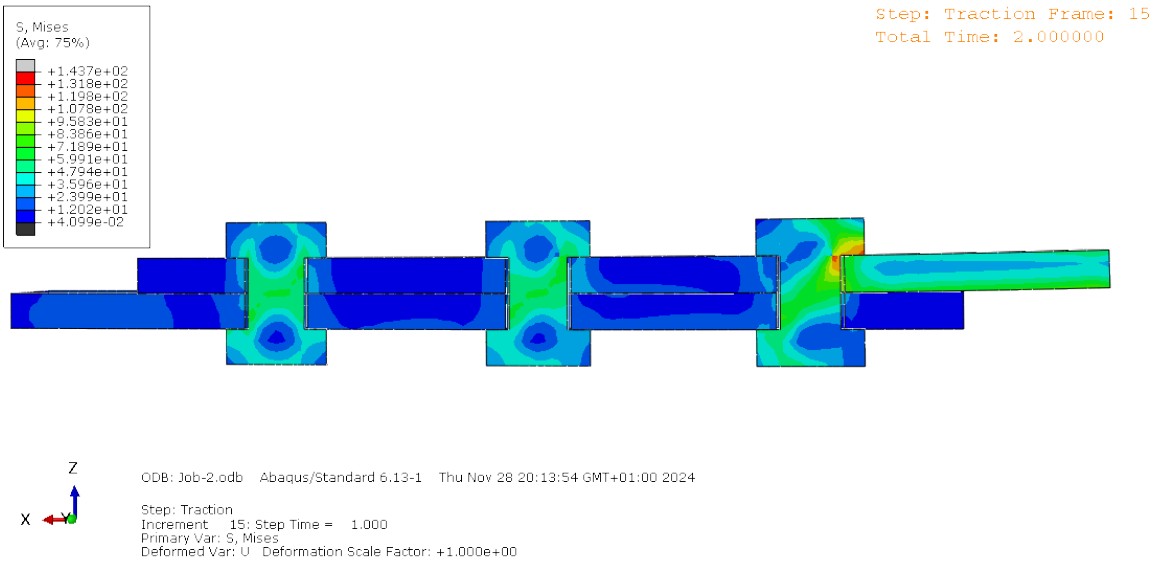
Notice that during the trial model setup it was evaluated how the model would react to the external excitation. The first rivet will be the one which carries at first all the load. When it cannot transfer any more load, the second rivet will start to receive the increment of force applied and, lastly, the last rivet will suffer the rest of the load once the previous ones have reached their limit. Therefore, as expected, by implementing a hole

clearance in the riveting process, the available loading has been reduced.

This can be explained by observing the following contours, where both models are compared. In figure 4.2(a) it can be seen a cut where the stress distribution (Von Mises stress) for model 1 is represented. Likewise, the same contour map for model 2 is provided in figure 4.2(b). Despite having the same shape both figures, the maximum stress value (located in the upper contact between inner rivet and upper plate) for model 2 is higher.



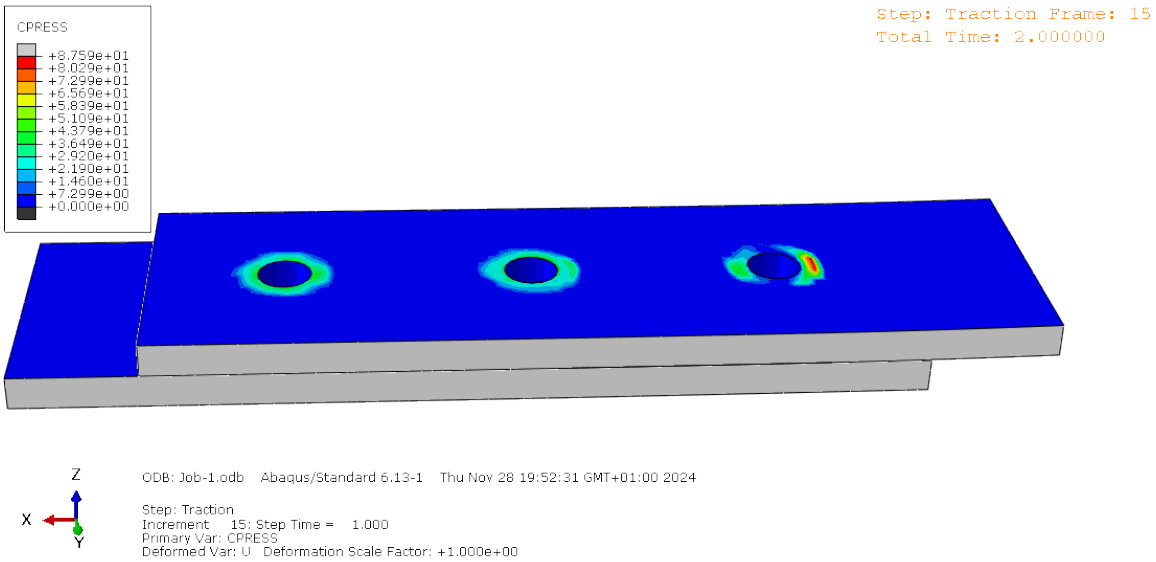
(a) Stress contour map for model 1



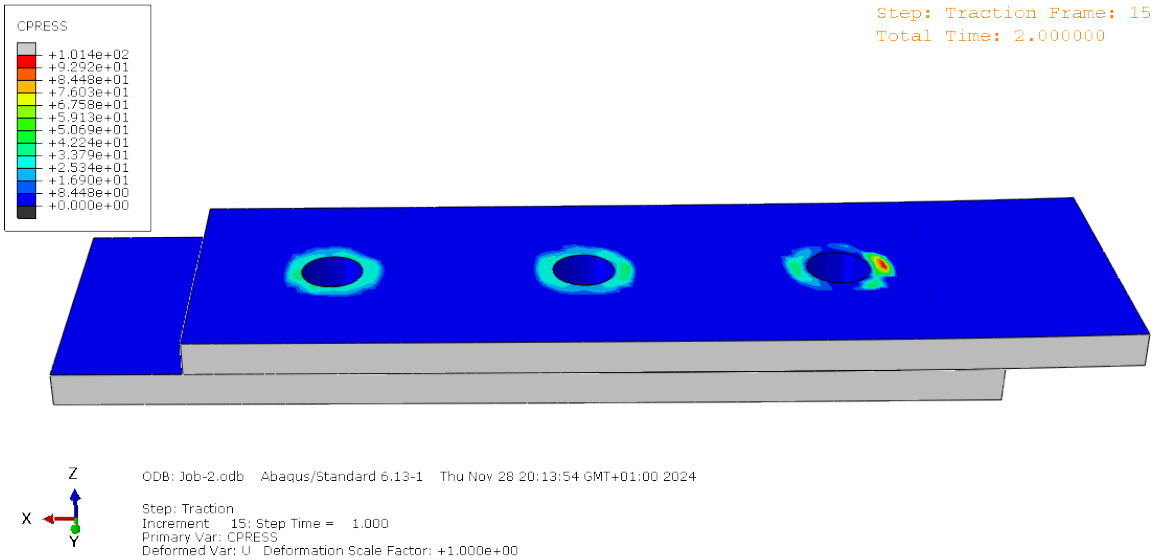
(b) Stress contour map for model 2

Figure 4.2: Stress contour map for both models

This means that, despite the fact that the inner rivet of the second model supports less load, this last is more concentrated compared to the first model. This can be also shown in figure 4.3, where the *CPRESS* loads are shown for both models. As it happened before, the contour shape is similar, but once again notice that the maximum load is presented in model 2, the one with hole clearance. This thus is coherent with the results shown in figure 4.2.



(a) CPRESS contour map for model 1



(b) CPRESS contour map for model 2

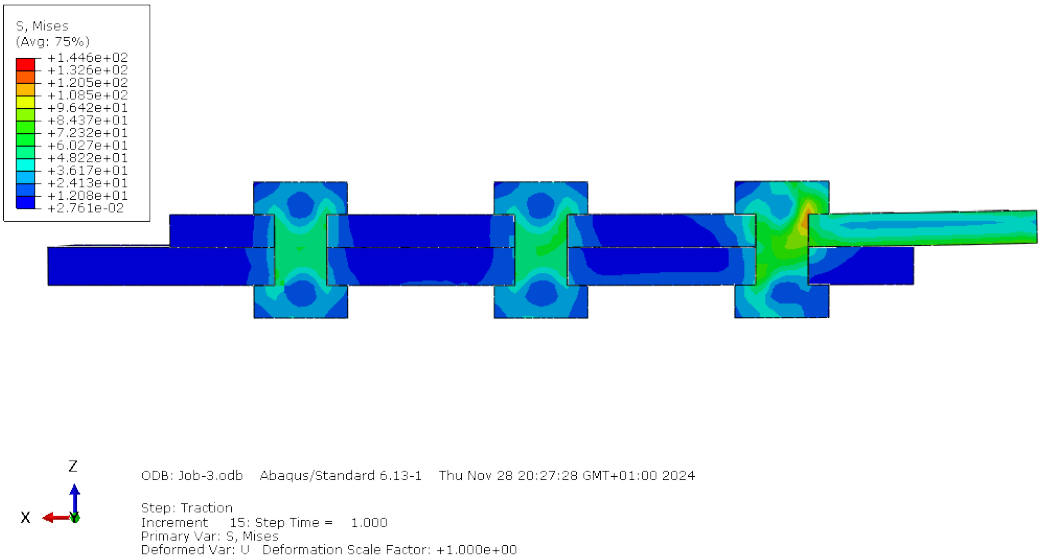
Figure 4.3: CPRESS contour map for both models

Recalling the article "*The structure of the strength of riveted joints determined in the lap joint tensile shear test*", and more precisely, figure 1.21 mentioned in the state of art previously in chapter 1, we can verify that the phenomenon occurring in model 2 is the one anticipated by this last document. When leaving clearance in between the fastener and the hole, the shank area used to transfer the shear load is reduced and thus, the stress is incremented in that region. This last aspect is coherent with the information retrieved from "*The effect of rivet gun operating pressure and hole clearance on rivet shear strength in sheet metal riveting process*", mentioned in the state of art, which indicated that the clearance would increase the stress of the critical rivet. Assuming the material is the same (which is the case for all of our models) since the area is reduced the maximum load will be hence reduced because of this reason. Actually, in figure 4.2 it can be seen that, looking at the inner rivet shank, the green area, which represents somehow the regions which transfer a considerable amount of load, is larger for the first model due to the lack of clearance.

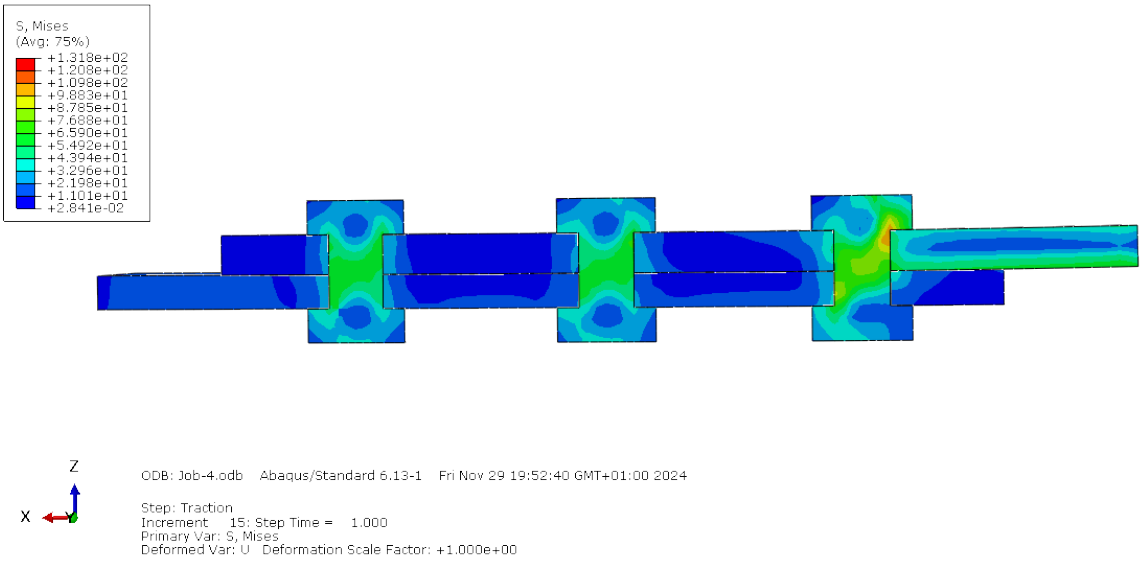
Even though this result was the expected one, since previous works on this chapter had already discussed this effect, it is important to verify that indeed the behaviour is the one expected. However, the objective of

this thesis is not only verify this hole clearance effect, but also quantify this last mentioned. So we can see that for a **hole clearance of 0.3 mm** the maximum shear load has been reduced by **145.5 N**, which accounts for a **21.62% reduction**. These 145.5 N have been distributed between the other two rivets. The middle rivet shear load has been incremented by 118.2 N, whereas the outer rivet shear load has received the rest 27.3 N, which accounts for 18.76% of the load which was transferred from inner rivet to the other two.

Model 2 allowed to evaluate the effect of hole clearance with respect to model 1. However, both model 3 and 4 do not have clearance. Instead, the aim of the next two models is to evaluate the effect of incrementing both of the plates thickness. Thus, as it can be observed in table 4.1, model 3 has a larger thickness for the upper plate, whereas for model 4, is the lower plate the one whose thickness is increased.



(a) Stress contour map for model 3

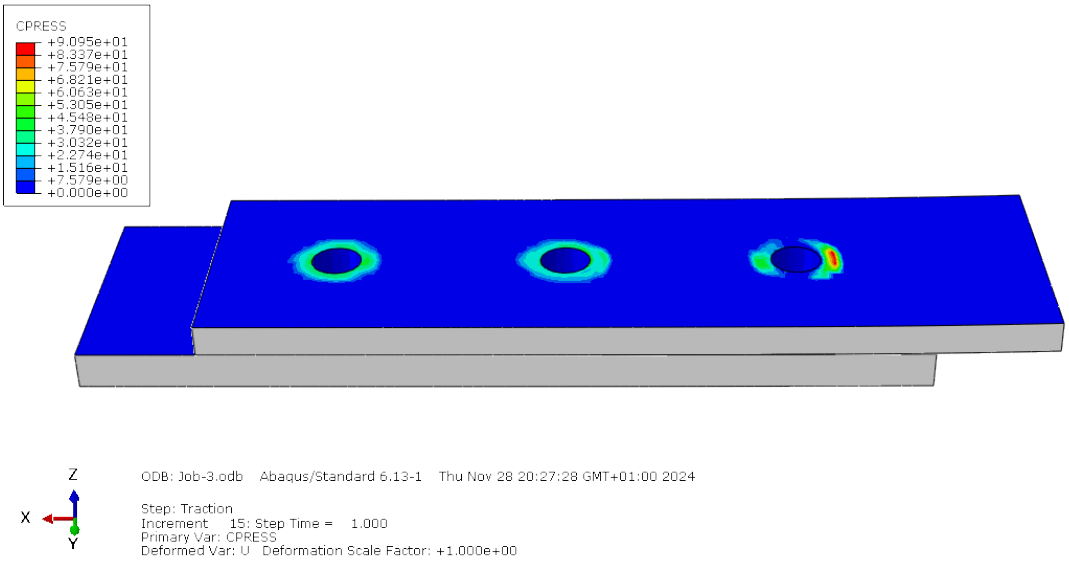


(b) Stress contour map for model 4

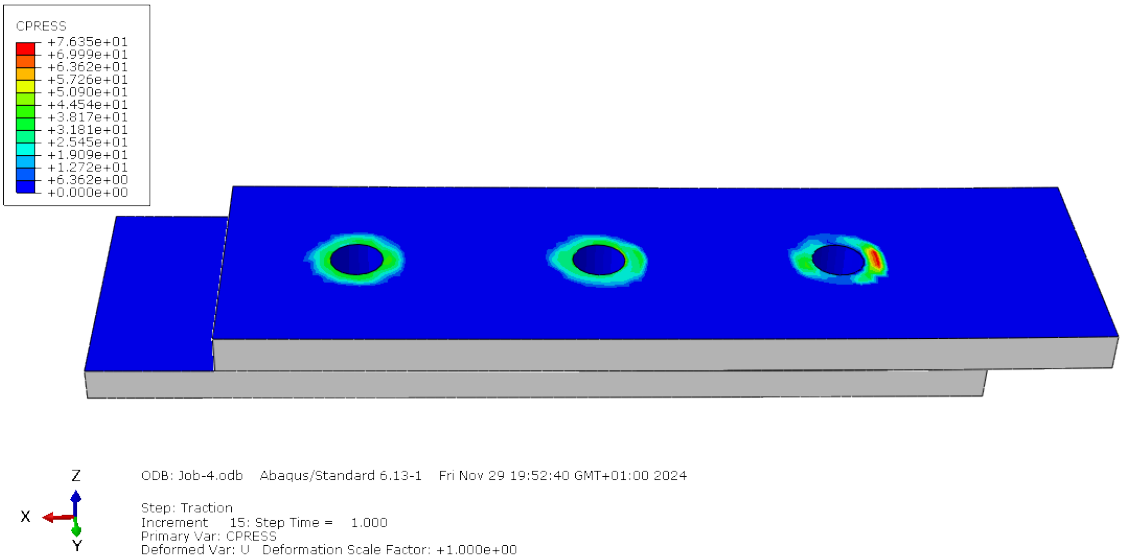
Figure 4.4: Stress contour map for both models

As it has been done with the first two models, the two contours can be shown, and these are represented in figures 4.4 and 4.5. Notice that both models have present the same shape in the contour and the differences are to small to be pointed out. They are not only similar to each other, but also to the figures presented for

model 1. In the shear force results it can be observed a small reduction in the inner rivet loading, nevertheless, this effect is way to small compared to the one produced by the hole clearance.



(a) CPRESS contour map for model 3

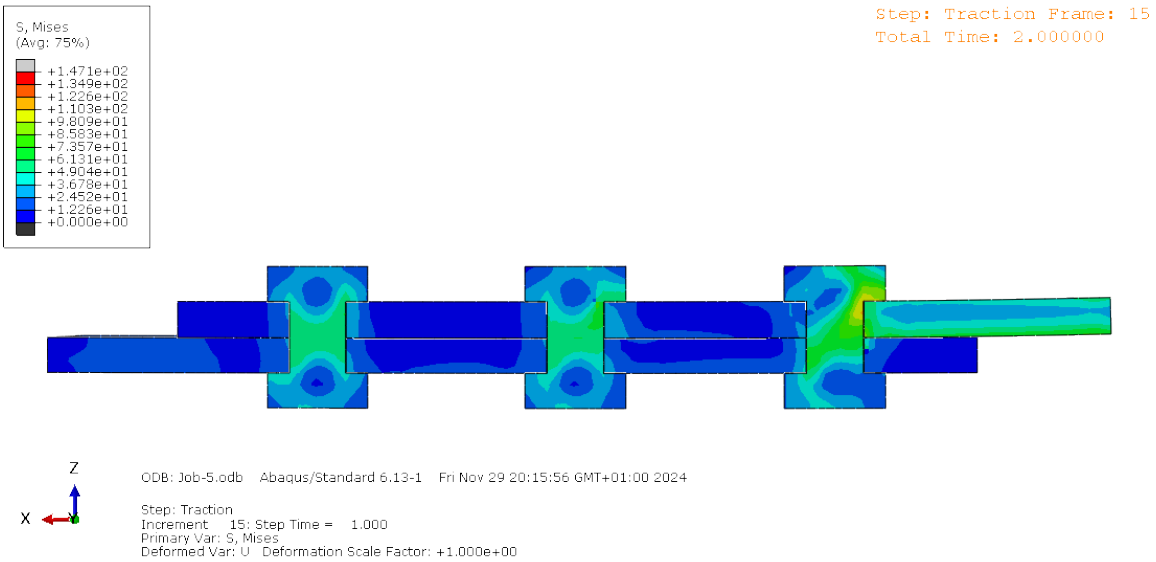


(b) CPRESS contour map for model 4

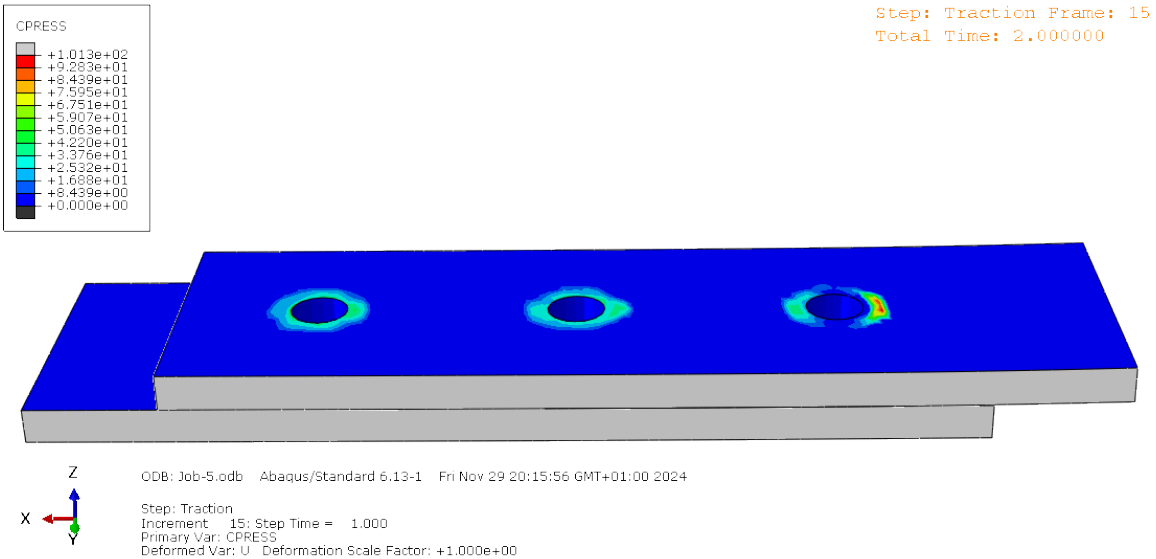
Figure 4.5: CPRESS contour map for both models

Therefore, by looking at the results of these two models, it can be determined that the difference in plate thickness does not influence that much in rivet shear loading compared to the clearance effect. Nonetheless, it is still unknown how is the relation between the clearance and the shear loading. Of course, we know from the state of art that the larger the clearance is, the less force it will be able to transfer until breaking, but we do not know still if this relation is linear, hyperbolic, or follows another trend.

Consequently, model 5 was built with a very small clearance, which was 0.05 mm, one sixth of the gap left in model 2. It was also added a small perturbation in the upper plate thickness to bring in the two effects seen in the previous models. However, as it was mentioned before, the leading cause of the shear load change will be the clearance effect. As it was done with the previous four models, the two contour maps are provided.



(a) CPRESS contour map for model 5



(b) CPRESS contour map for model 5

Figure 4.6: Contour maps for model 5

Following the same reasoning of the model 2 discussion, once again it can be observed in figure 4.6(a) the reduction of the load path inside inner rivet, since the green area is smaller and has a different shape compared to the other two rivets. Unlike it happens with models 1,3 and 4 where the stress contour has somehow a similar shape in the three rivets, when gap is created this loading zone tends to shrink, reducing the contact area and increasing therefore the stress in the critical regions, as stated in the article "*The effect of rivet gun operating pressure and hole clearance on rivet shear strength in sheet metal riveting process*".

Notice that the maximum values in the scales shown in figure 4.6 are very similar to the ones in figures 4.2(b) and 4.3(b), which indeed leads to a shear load distribution close to the one represented in model 2. This time, maximum load is reduced by **17.71%** (recall for model 2 the reduction was a 21.62%), being the highest value **119,2 N lower** than the one found in model 1. These are the results of leaving a **clearance of 0.05 mm**.

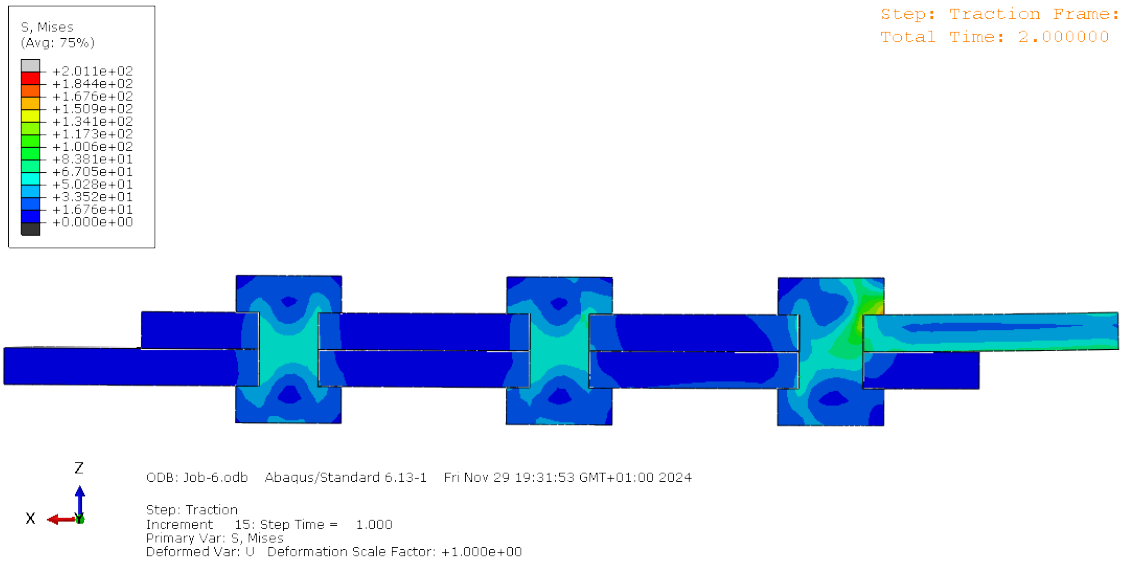
As soon as the clearance exists, the force of the inner rivet will be reduced drastically and it will continue doing so as this gap keeps getting larger. Nevertheless, the rate of reduction is reduced as the clearance

increases, that is the information we can retrieve from our models. In fact going back to the article "*The effect of rivet gun operating pressure and hole clearance on rivet shear strength in sheet metal riveting process*", for low external excitation we can see the same behaviour represented in the blue line of figure 1.18, so the results obtained are coherent with the previous works developed regarding this topic. This was the reason why model 5 was so necessary, since now we can have an idea on how clearance and inner rivet shear force are related.

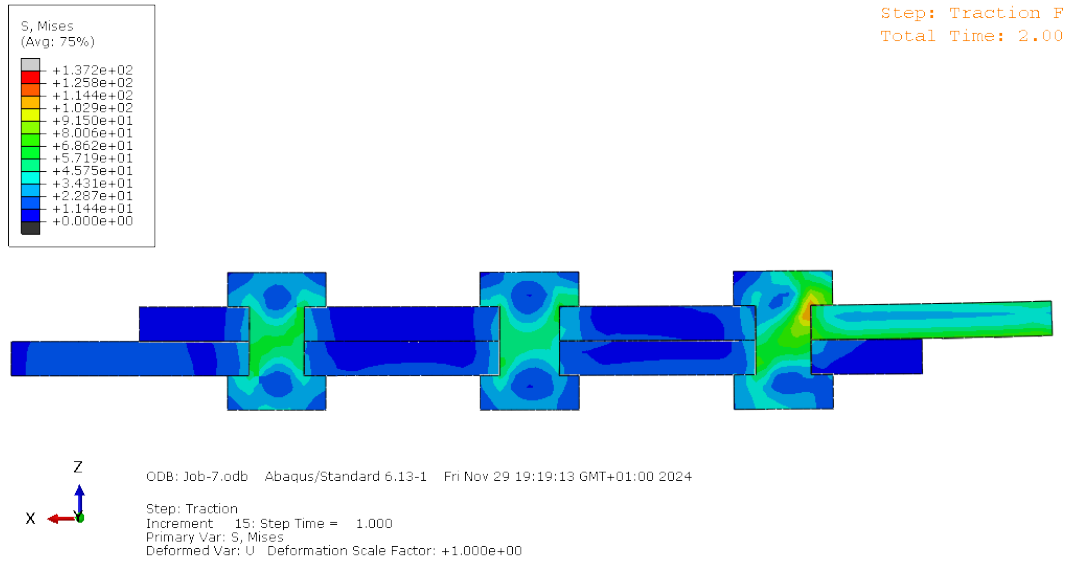
After focusing almost exclusively on studying the effect of the gap between the drilled element and the rivet shank, the next element analyzed is the influence of the rivet diameter. Models 6-8 will study the influence of incrementing this magnitude without the presence of hole clearance, whereas models 9-11 will do the same but introducing this mentioned gap. We can synthetize the delta expressed in each of the mentioned models as follows:

- **Model 6**: Inner rivet shank diameter is increased. No gaps present.
- **Model 7**: Center rivet shank diameter is increased. No gaps present.
- **Model 8**: Outer rivet shank diameter is increased. No gaps present.
- **Model 9**: Inner rivet shank diameter is increased. There is hole clearance.
- **Model 10**: Center rivet shank diameter is increased. There is hole clearance.
- **Model 11**: Outer rivet shank diameter is increased. There is hole clearance.

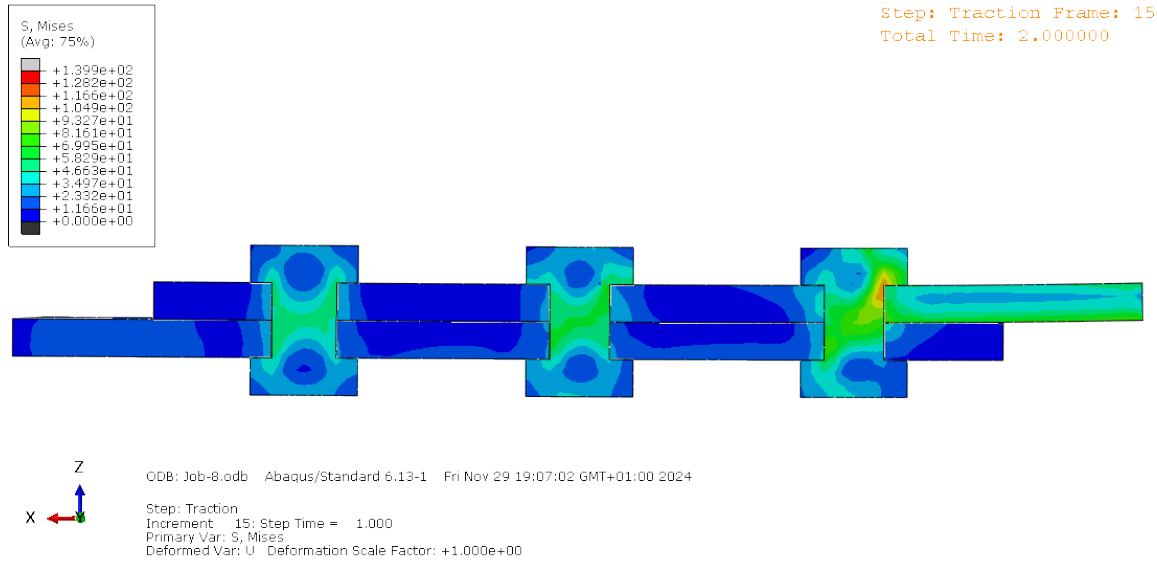
Starting with the models without clearance, as always, the contours for the stress distribution and the *CPRESS* values are presented in order to analyze the differences between them. But before starting the results discussion, the theory tell us that the bigger the shank diameter is, the more shear load it will carry.



(a) Stress contour map for model 6

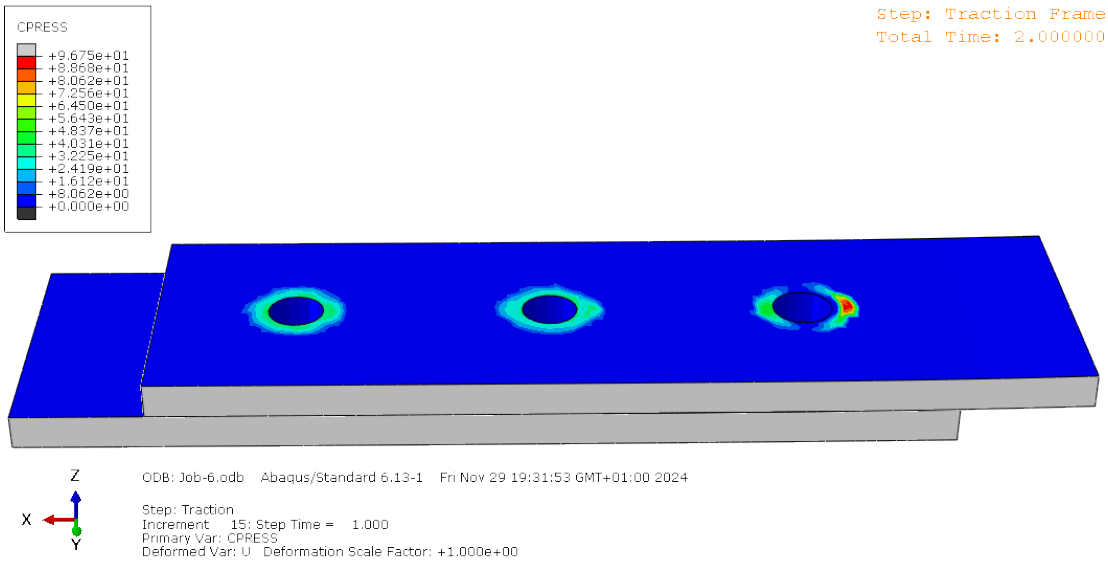


(b) Stress contour map for model 7

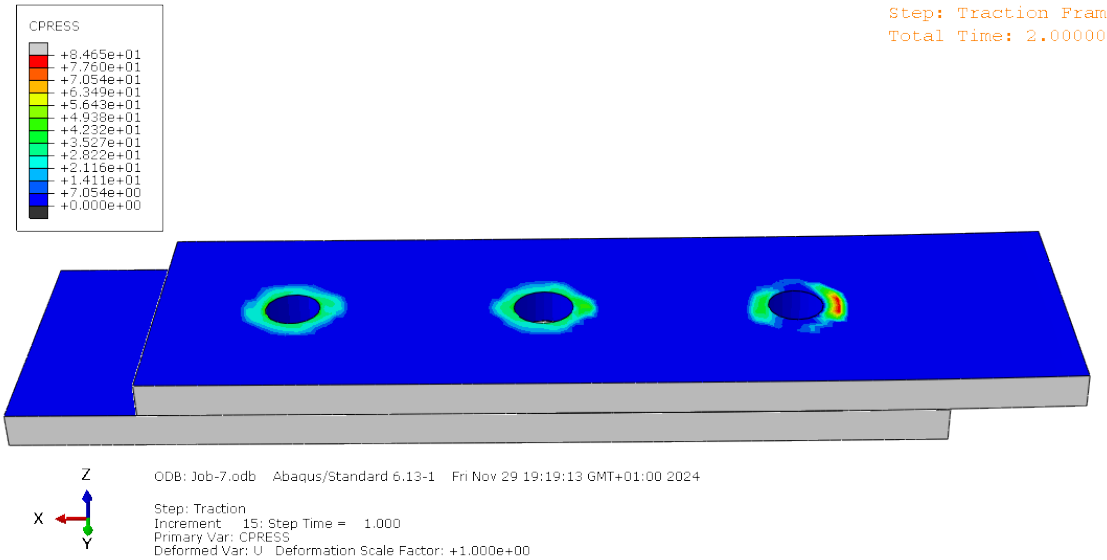


(c) Stress contour map for model 8

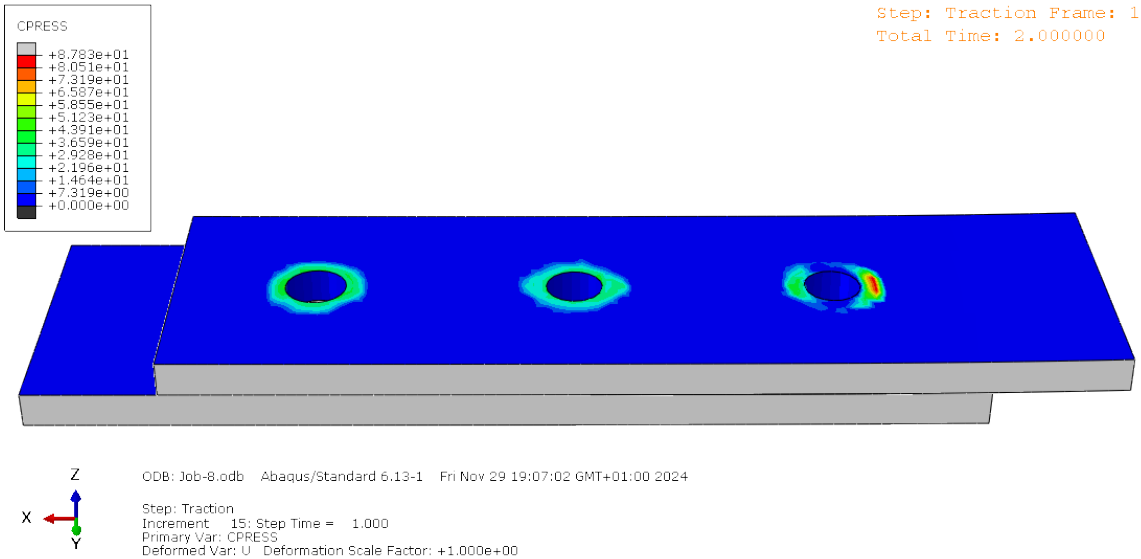
Figure 4.7: Stress contour map for models 6-8



(a) CPRESS contour map for model 6



(b) CPRESS contour map for model 7

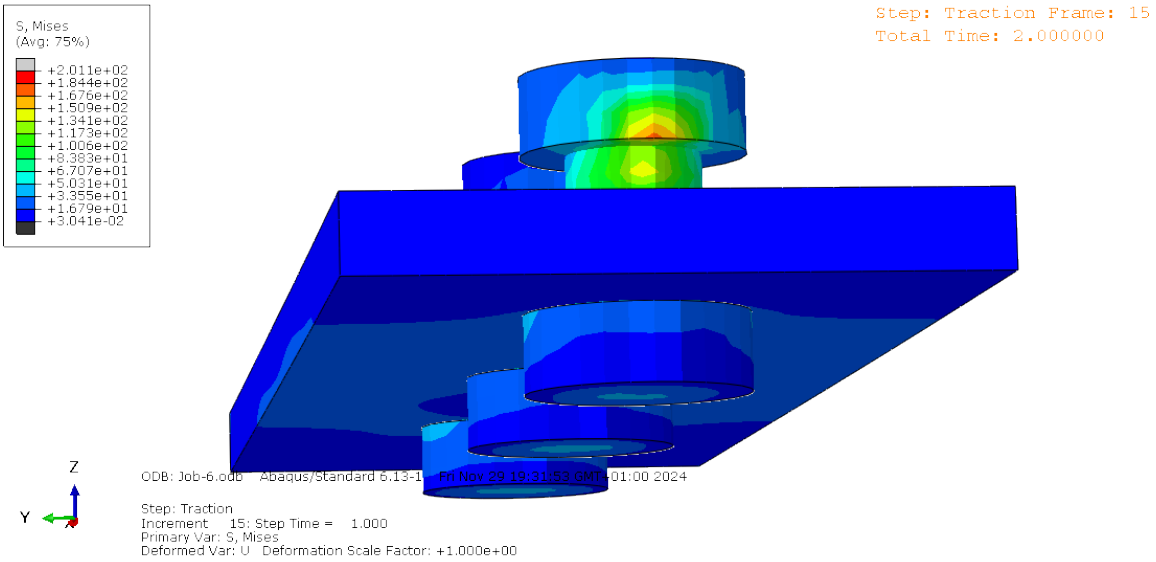


(c) CPRESS contour map for model 8

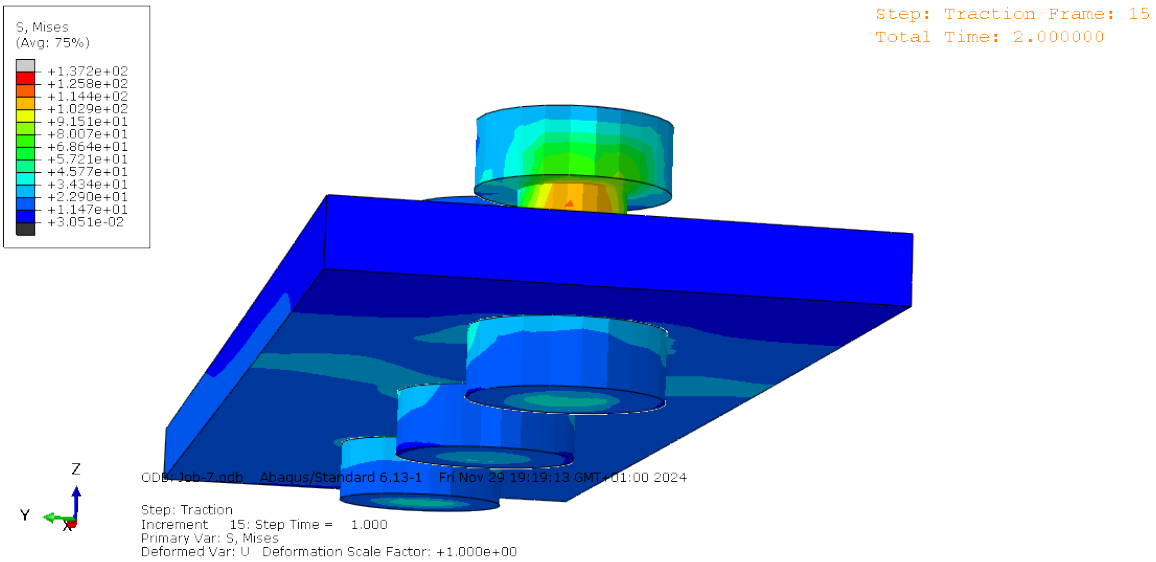
Figure 4.8: CPRESS contour map for models 6-8

First thing that comes up when comparing the models in figure 4.7 is that the range of stresses for model 6 is different with respect to the other two. In fact, the maximum stress in model 6 (whose region is not visible in figure 4.7(a)) is much higher compared to the other two. The theory, as mentioned previously, tell us that the biggest shank should carry the most shear load under this conditions. Nonetheless, we check that maximum stress presented in the inner rivet is increased instead of decreased, which would be the expected result following this reasoning. So, why this happens?

In order to figure it out, we need to take a closer look at figure 4.9. Recall the DOE changes the rivet shank diameter but not its head diameter. This fact is relevant to make it simple to understand. As we increase the shank diameter, the gap between both diameters is decreased, and therefore, the circular crown formed by at the head base is reduced. Therefore, as a direct cause, since the contact area between the rivet head and the plate is reduced, the stress will increase. We can see that for model 7 (figure 4.9(b)) the region where the stress is the highest does not even belong to the rivet head, whereas in model 6 (figure 4.9(a)) this does happen.



(a) Rivet head detail for model 6



(b) Rivet head detail for model 7

Figure 4.9: rivet head detail for stress concentration inspection

Having less contact area between the head and the plate reduces the maximum shear load, but as mentioned before, increasing the shank diameter makes the rivet absorb more shear load. Thus, both effects somehow counteract each other, leaving a shear load distribution similar to the ones of models 1, 3 or 4.

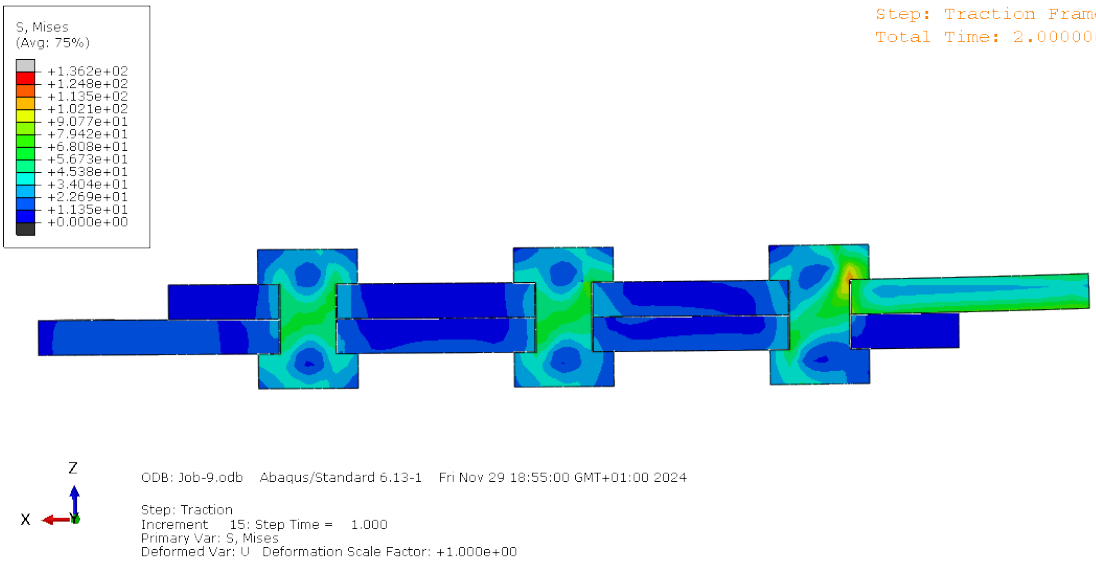
Now, jumping to model 7, we would have the same situation as in model 6 but with the middle rivet. Increasing the shank diameter will make the center rivet to absorb more shear load, as it is reflected in the model results. However, unlike in model 6, the rivet head - upper plate interaction is not that relevant. This can be easily checked looking at figure 4.8, where it can been that for the three models, the important part occurs close to the inner rivet. Therefore, in this case, this last effect plays a smaller role in the analysis, making hence the distribution more even due to the fact that the middle rivet absorbs more shear load thanks to its bigger diameter. Notice that, as the middle rivet is carrying more load, because of the assembly mechanism, the outer rivet, consequently, will transfer more shear with respect to the original case.

Lastly, in model 8 we check that, despite the outer rivet has a larger diameter, the loading does not change that much compared to the initial case. The same way in model 7, the rivet head effect was not that important, in this last model, both rivet head interaction and diameter size are not relevant to the global model. This happens as a consequence of the assembly mechanism. The load has to go through the first two rivets after reaching to this outer fastener. Therefore, the amount of load which can be attracted by increasing the diameter is very low. Actually, the shear load for this rivet is increased, but as said, it does not represent a significant change due to the load path determined by the mechanism itself.

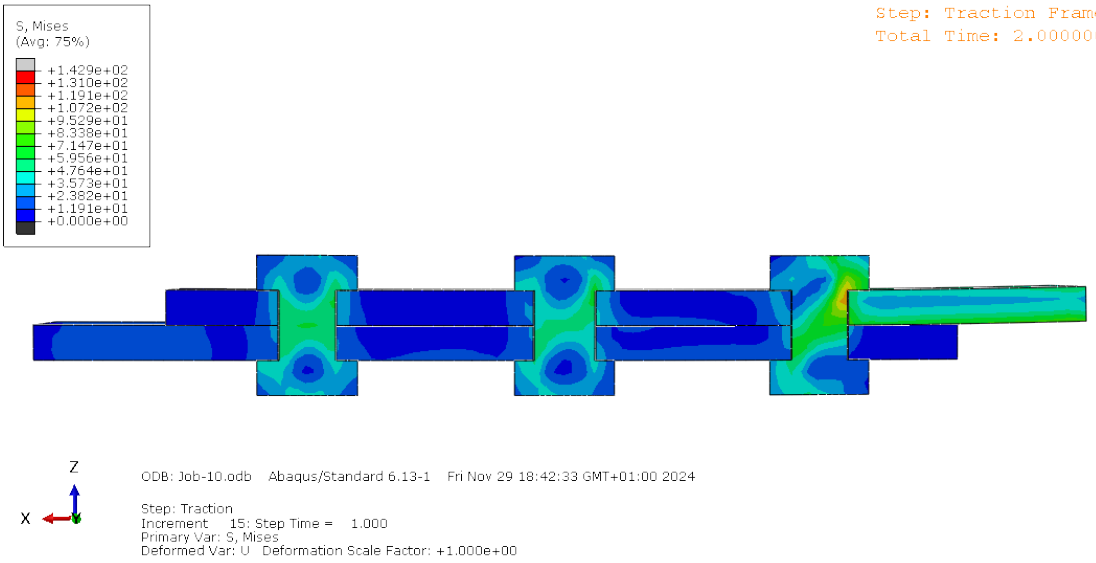
Once these three models have been analyzed, we can jump to models 9-11 where in addition to a shank diameter increase, there is the presence of hole clearance. As always the two contours are shown for the three models since those are the ones which keep track of the most important regions in the model, showing the stress distribution along the shank and the contact pressure contained in the top plate close to the inner rivet.

From previous results, due to the presence of a gap between the rivet and the hole it could be seen that the stress in the inner shank was distributed in such a way the maximum stress was forming a diagonal lines which connects the two critical points of the rivet when interacting with the plate. In figure 4.10 it can be seen that indeed, this behaviour is repeated as a consequence of leaving that clearance. This diagonal shape, once again, reduces the transfer load area between the shank and the plates, thus incrementing the local maximum stress and reducing the amount of load which can be carried by the first rivet. As a result, this leads to a lower highest value in terms of rivet shear loads, which indeed is coherent once again with the reasoning applied and both previous results and the state of art.

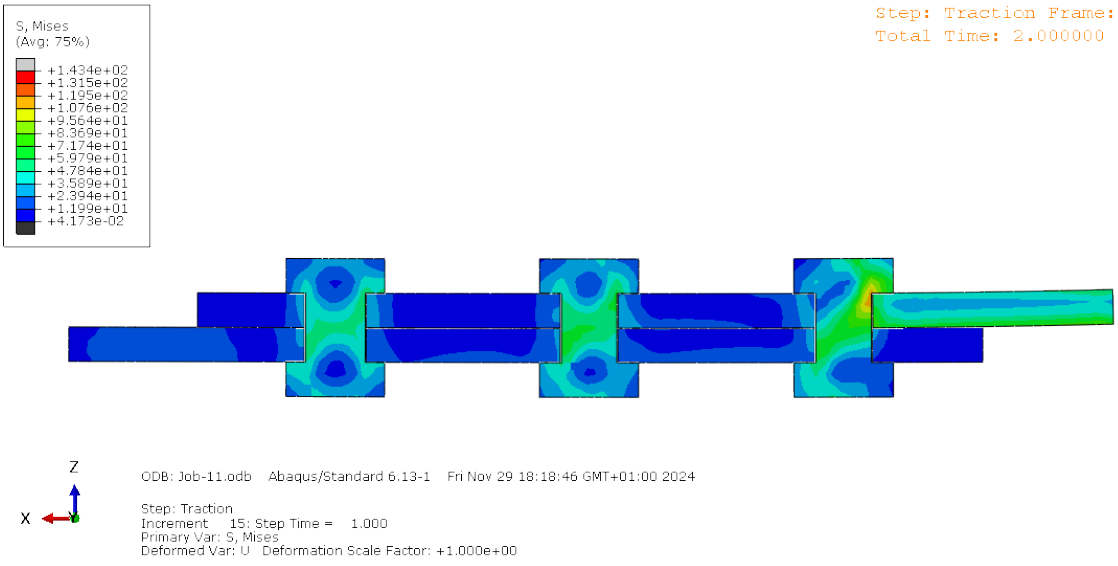
However, unlike model 2, these models apart from the clearance effect, they had diameter differences. Nonetheless, as anticipated before, the diameter effect has a lower impact compared to the one of hole clearance. Going back to the results discussion of model 6, the reason why inner rivet did not carry more load was because the head diameter was not increased, and thus, the critical region where stresses were concentrated was not the rivet shank but the rivet head. Since this area was reduced, the stress increase in that region and the capability of absorbing load is reduced.



(a) Stress contour map for model 9



(b) Stress contour map for model 10



(c) Stress contour map for model 11

Figure 4.10: Stress contour map for models 9-11

Nonetheless, when the clearance appears, the region were most of the load is transferred is no longer the rivet head but once again the upper part of the shank. In fact, in figure 4.11 it is shown a representation similar to the one of figure 4.9, corresponding to models 6 and 7. As seen the stress contour indicates that the shank is subjected to larger stresses (this figure is more similar to figure 4.9(b) rather than figure 4.9(a)), and this image is the key to understand why the clearance effect has dominance over the diameter increase, because as soon as this clearance appears, the diameter head is not loaded that much and therefore all the shear load reduction is carried by the presence of the gap, being the contour of figure 4.11 very similar to the sketch 1.21 contained in article "*The structure of the strength of riveted joints determined in the lap joint tensile shear test*", referenced in the state of art.

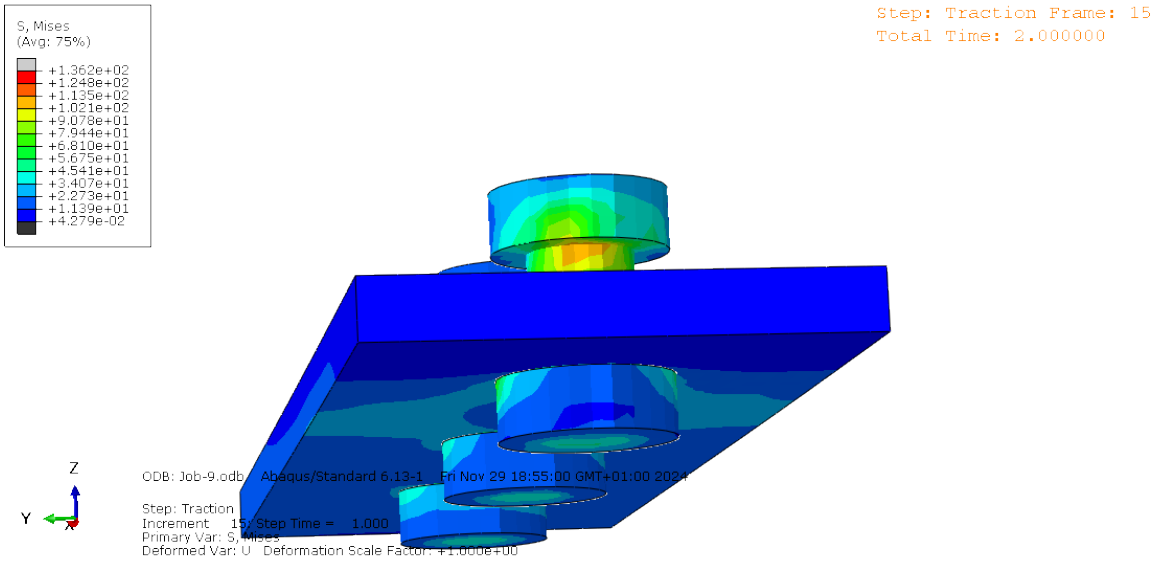
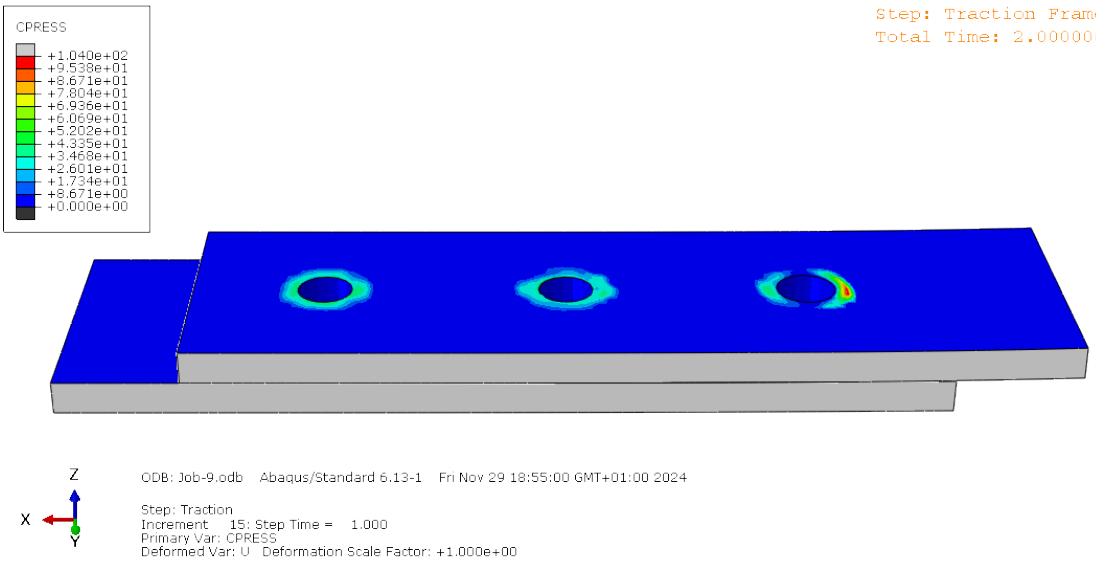


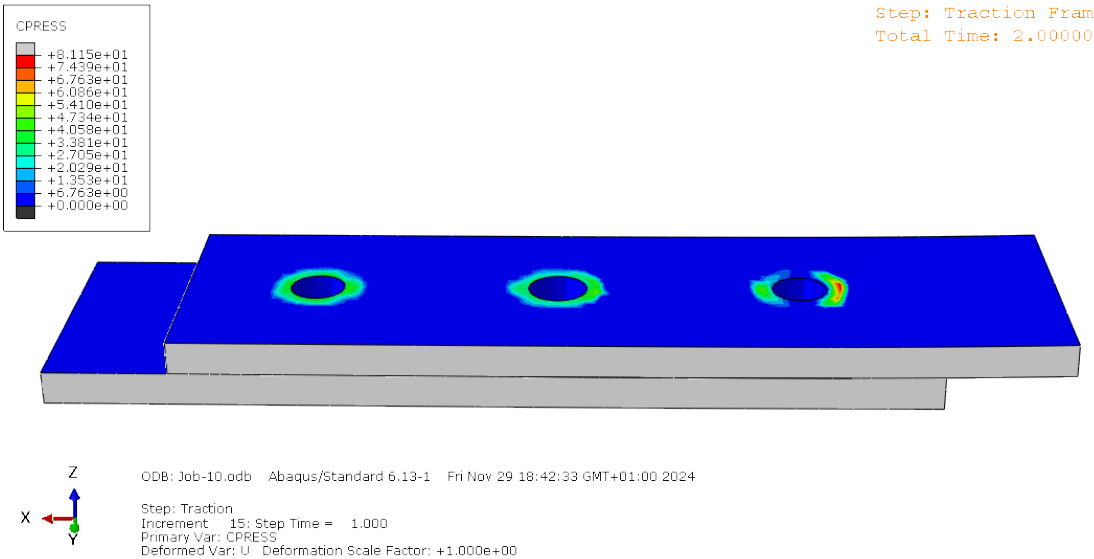
Figure 4.11: Rivet head detail for model 9

This fact explains why the results do not differ that much with respect to the ones of model 2, where all the rivet diameters were identical. Notice that the presence of clearance does not eliminate the effect of the diameter change. In terms of shear loading, it may not have to much influence with respect to the other mentioned variable, but still these last three models present some differences between themselves, caused by the different position where diameter is incremented. in figure 4.12 it can be checked that the maximum pressure is higher in model 9 compared to the other two models which have similar scale values. This is due to the increment in diameter, since, as it happened with model 6, the contact are between the rivet head and the plate is reduced and so the stress in that region will be higher. In fact, check that for figures 4.12(b) and 4.12(c) the red regions are larger, which makes sense.

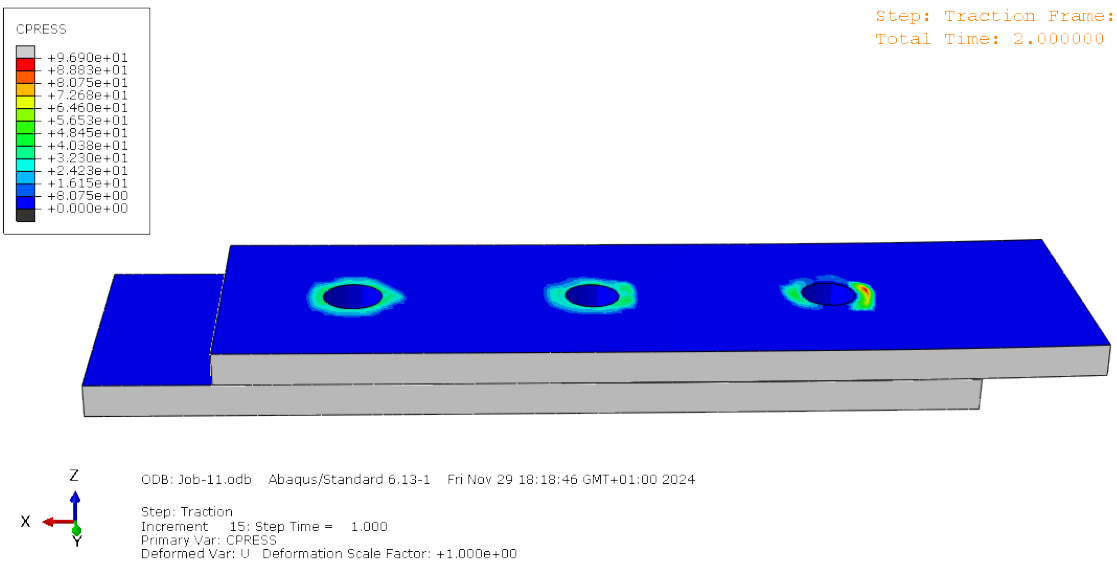
Another effect caused by changing the diameter is seen in figure 4.10, more specifically in 4.10(b) and 4.10(c), where it can be appreciated that the stress values for the rivets which have larger diameter are lowered (in the contours the colours tend to be more blue than green). If there were not clearance, this of course would have an impact in the shear loading distribution. However, because of the reasons mentioned before regarding what happens at the first rivet, these changes do not imply relevant deviations in rivet shear loads since the loading transfer between plates starts in the inner fastener.



(a) CPRESS contour map for model 9



(b) CPRESS contour map for model 10

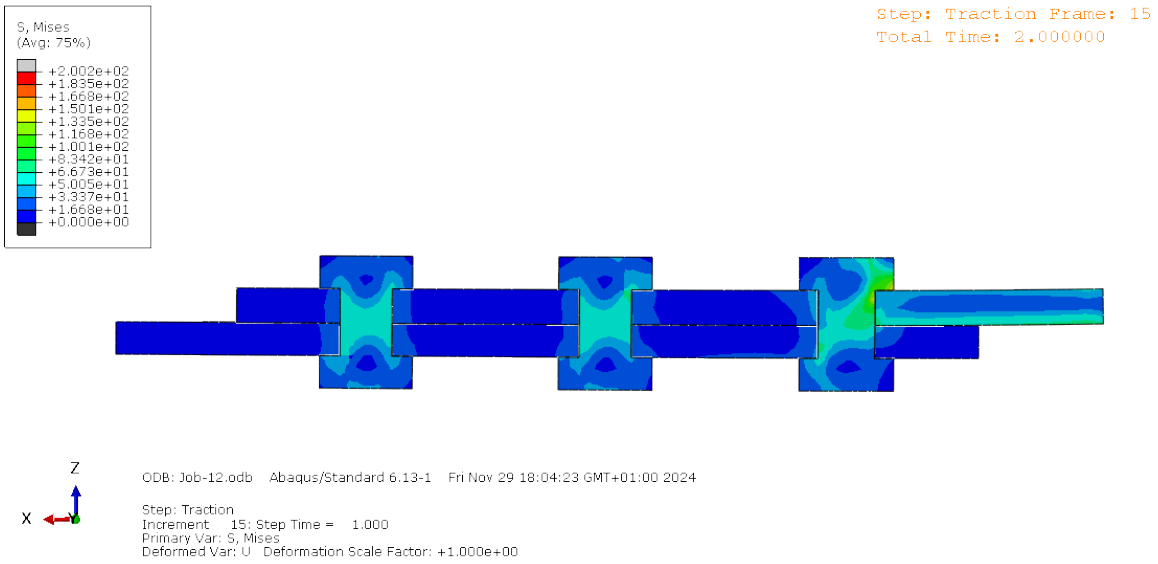


(c) CPRESS contour map for model 11

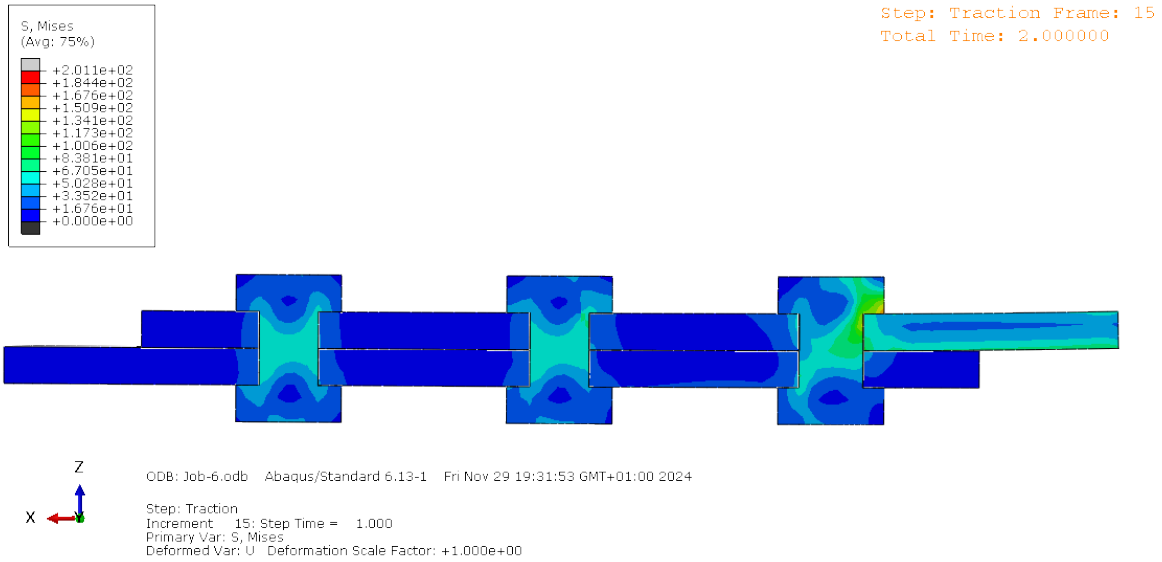
Figure 4.12: CPRESS contour map for models 9-11

Continuing with the results discussion, we are left with the last four models. First two evaluate the thickness effect combining both clearance and diameter differences, whereas the last two focus on determining the effect of the offset (defined as o_x) in two different cases.

Let's start first with the models 12 and 13. Taking a look back at table 4.1, notice that model 12 is exactly the same as model 6 but with an increment in thickness for the upper plate. Something similar happens with model 13, which is almost identical to model 9. Therefore, in order to understand an evaluate the isolated effect of increasing the thickness in only one plate, the contours of model 12 will be presented right next to the ones of model 6, and so will happen with models 13 and 9, respectively.



(a) Stress contour map for model 12



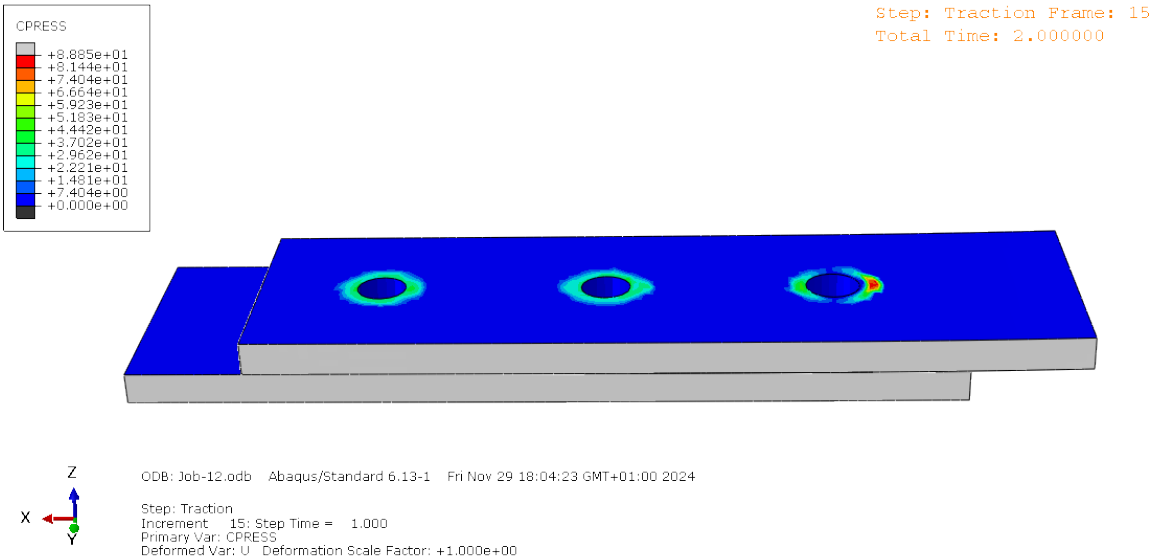
(b) Stress contour map for model 6

Figure 4.13: Stress contour map for models 12 and 6

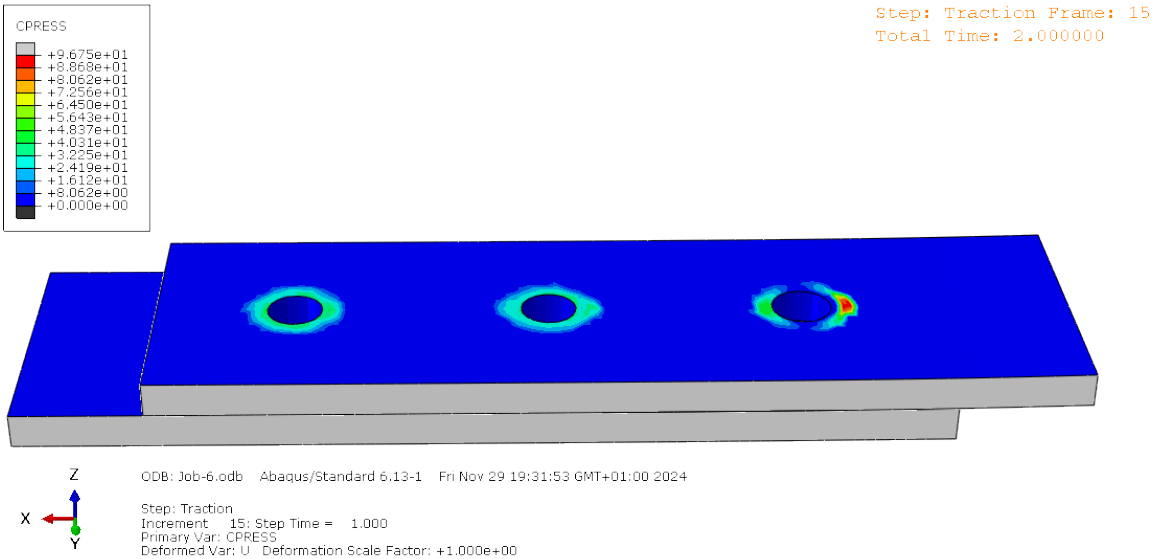
Starting with figures 4.13 and 4.14, it can be seen that the contours are almost identical. For model 12 the upper plate is 0.2 mm thicker but apart from that, recall the rest of the model parameters are the same. Despite the contours being very similar, notice that this slight deviation in the thickness caused the maximum stress to be reduced, but what is the reason behind that?

The pretension excitation is enforced in the shank cross-section corresponding to the plane belonging to the mating surface of both plates. Thus, when creating a thicker upper plate, we are moving the pretension point (in z-direction) towards the collar with respect to the model centroid (once again, this refers to the z-coordinate). Therefore, since the critical region for our mechanism is the interaction zone between the upper plate, the shank and the rivet head, for this model this region is further from the pretension plane compared to the case of model 6, which indeed leads to a lower maximum stress.

This happens for both the rivet shank profile (figure 4.13) and the rivet head interaction with the top plate (figure 4.14). Nevertheless, these differences are very small comparing the results from both model 6 and 12, and therefore, the rivet shear load distribution along the model is almost the same, as seen in table 3.3.



(a) CPRESS contour map for model 12

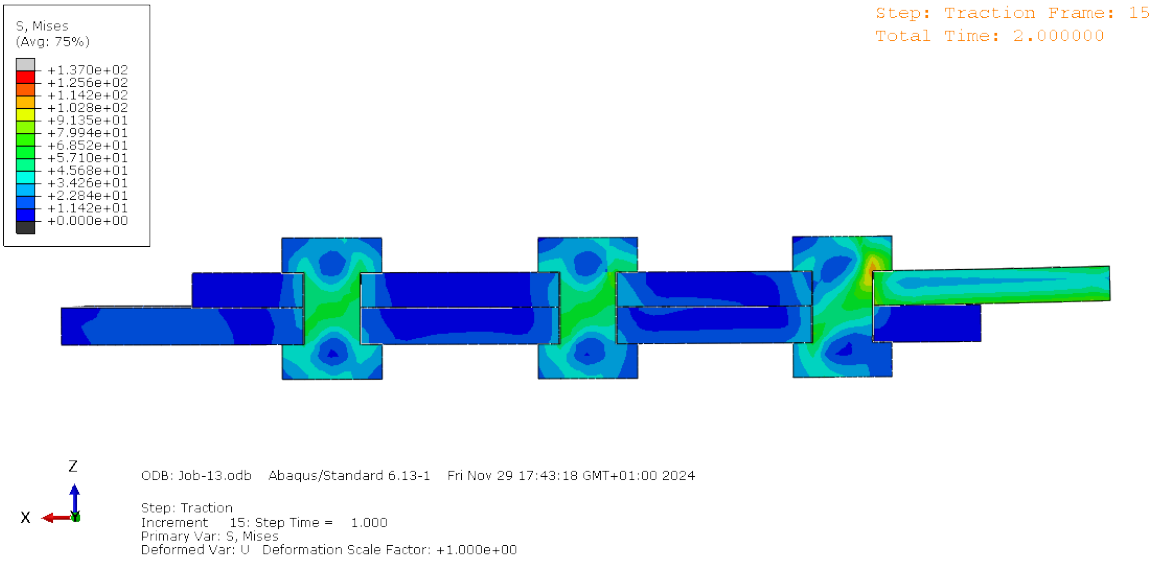


(b) CPRESS contour map for model 6

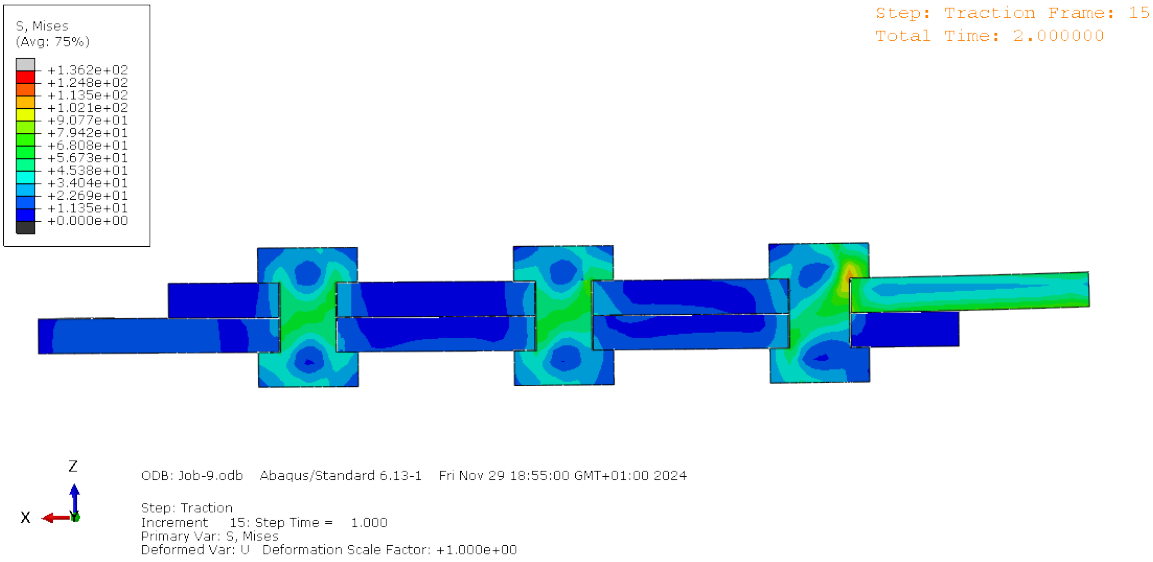
Figure 4.14: CPRESS contour map for models 12 and 6

Unlike in model 12, in model 13 the lower plate which is subjected to the traction force is the one whose thickness is being increased. Recall that the pretension force is applied in the shank in the specific surface where both plates interact.

Since this time the lower plate is thicker, the pretension point is moved towards the head with respect to the rivet centroid, which will cause a small increase in the stress in the head region, which is the most critical of our assembly. This, consequently, leads to an increase in the maximum stress captured by both contour maps. Once again, the differences between contour maps of model 9 (figures 4.15(b) and 4.16(b)) and model 13 (figures 4.15(a) and 4.16(a)) are very small, which makes the rivet shear loading very similar in both models.



(a) Stress contour map for model 13



(b) Stress contour map for model 9

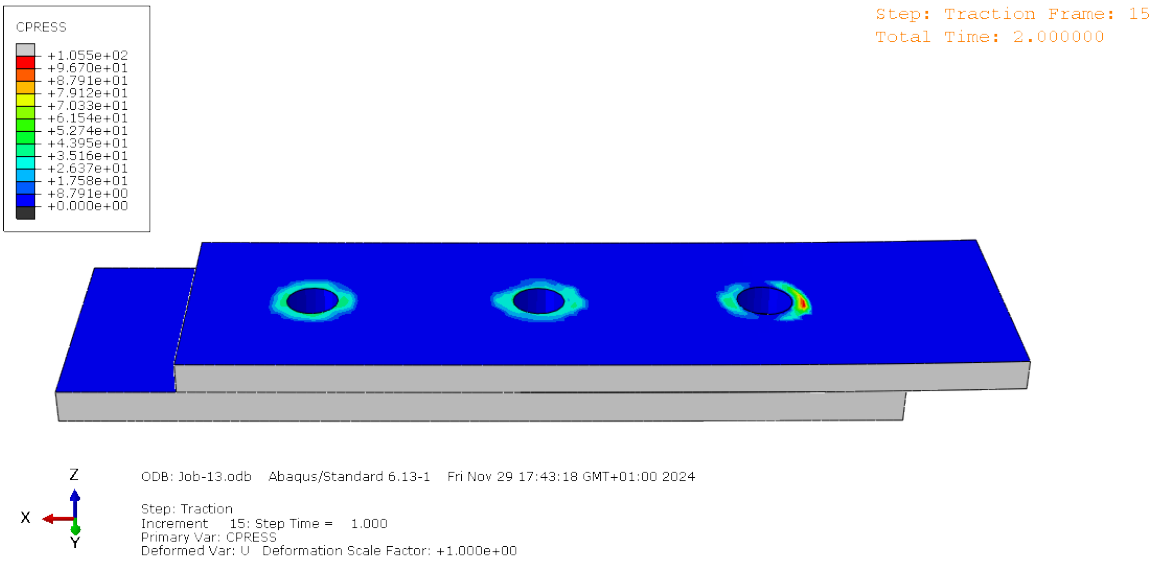
Figure 4.15: Stress contour map for models 13 and 9

However, no matter which plate thickness is increased, the maximum rivet shear load is decreased. By increasing the thickness of the plates, the rivet shank is also increased, and so the distance between the head and the collar. As a consequence, if the pretension force applied is the same (which is the case) the capability to transfer shear in the first rivet is reduced. So in order to summarize the effect of thickness in stress distribution, we can make the following points:

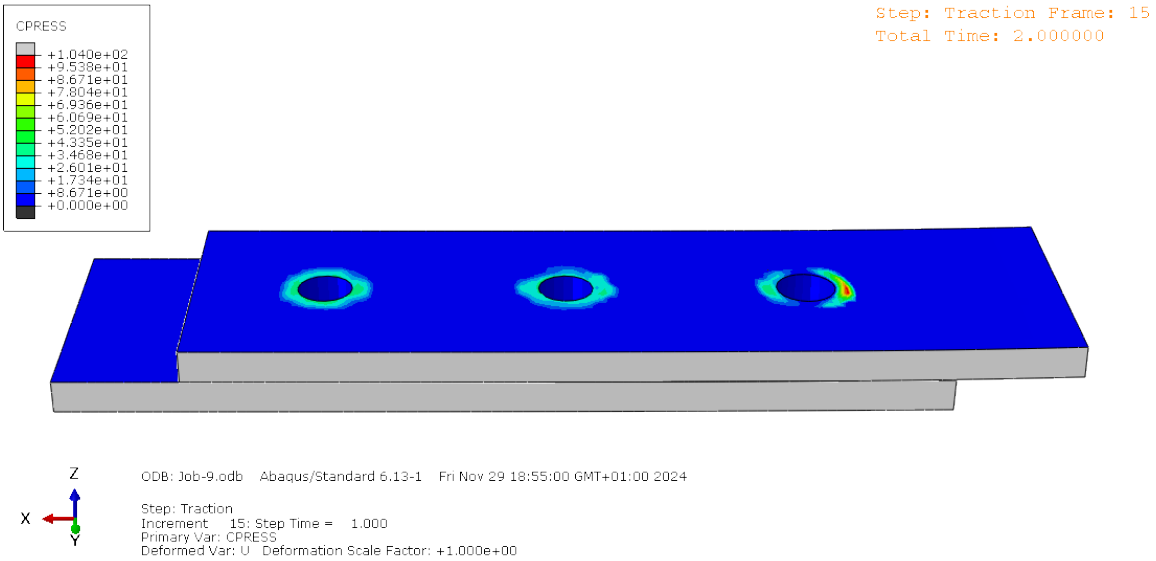
- When increasing one plate thickness, the local stress in that region will be reduced, since it will be

further from the pretension point.

- No matter which plate thickness is increased, the shear load transferred by the critical fastener will be reduced.
- This thickness variation has still a lower influence in the rivet shear load distribution compared to the hole clearance effect.



(a) CPRESS contour map for model 13

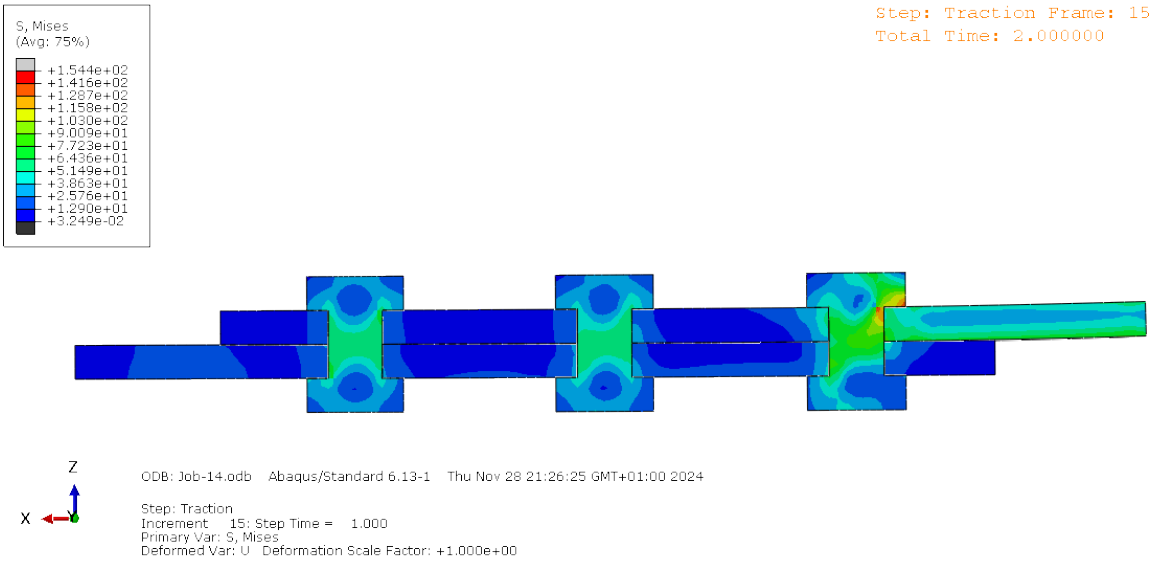


(b) CPRESS contour map for model 9

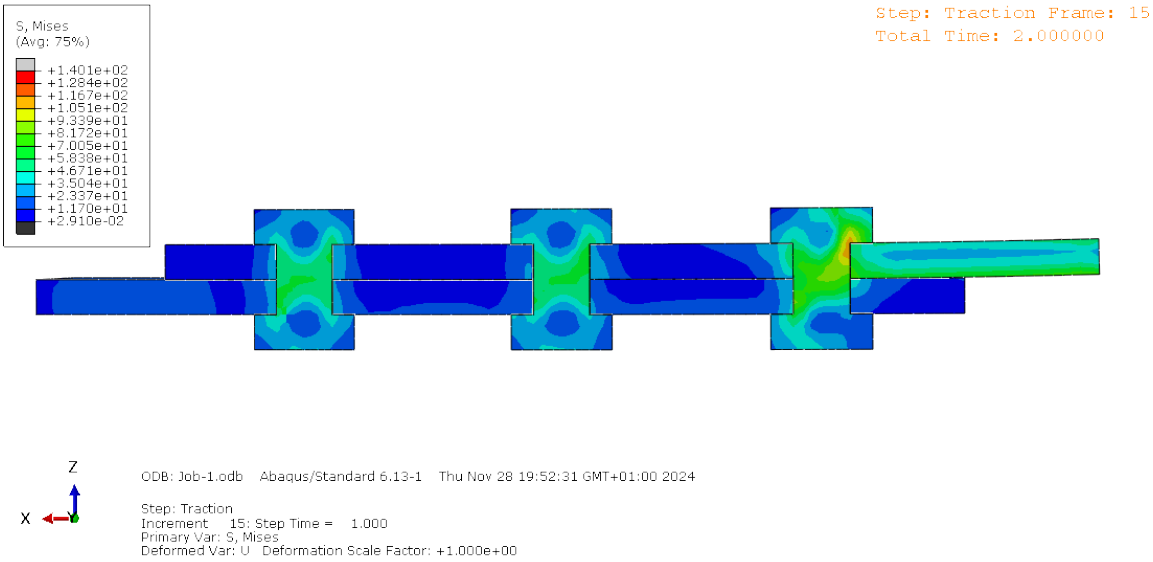
Figure 4.16: CPRESS contour map for models 13 and 9

Lastly, the remaining two models to be commented are model 14 and 15. These two have the intend to evaluate the effect of the parameter o_x. In the case of model 14, there is no hole clearance nor diameter difference between the rivets, and both plates have the same thickness. Indeed, model 1 is the same as model number 14, but this last one has a larger o_x. Therefore, as it have been done with models 12 and 13, contours for models 1 and 14 will be presented together in order to evaluate the effect of this parameter o_x.

Notice that, for the case of model 15, we do have both hole clearance and a diameter difference between the rivets, which makes this model almost identical to model 13. Like it happened before, the only difference between them is the parameter o_x, and therefore, the contours of both model 13 and 15 will be presented together to check the influence of the desired parameter when hole clearance, diameter difference and thickness difference is present.



(a) Stress contour map for model 14

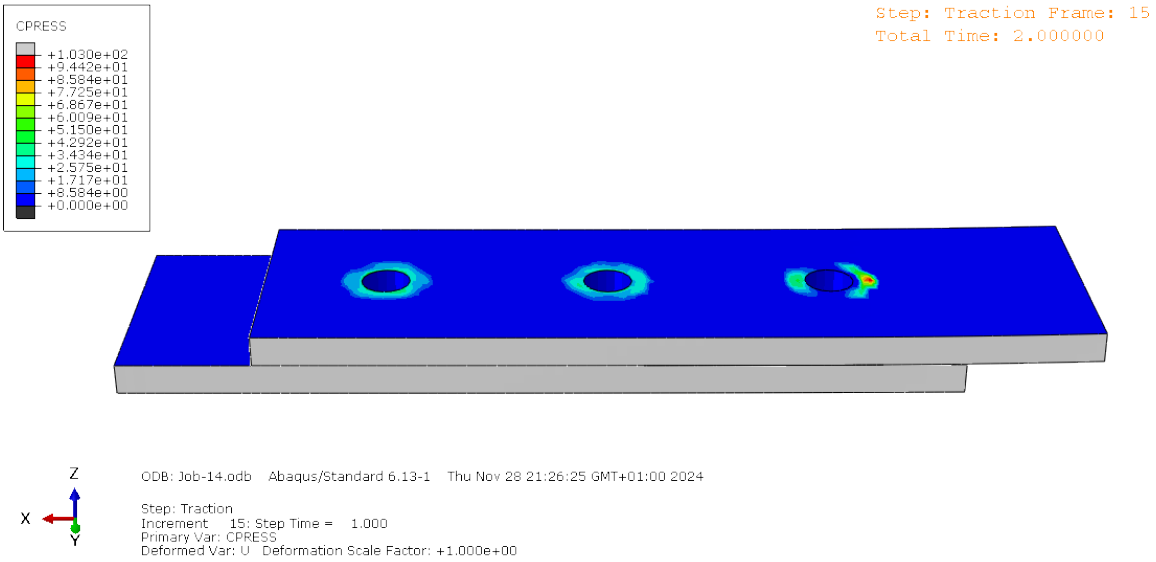


(b) Stress contour map for model 1

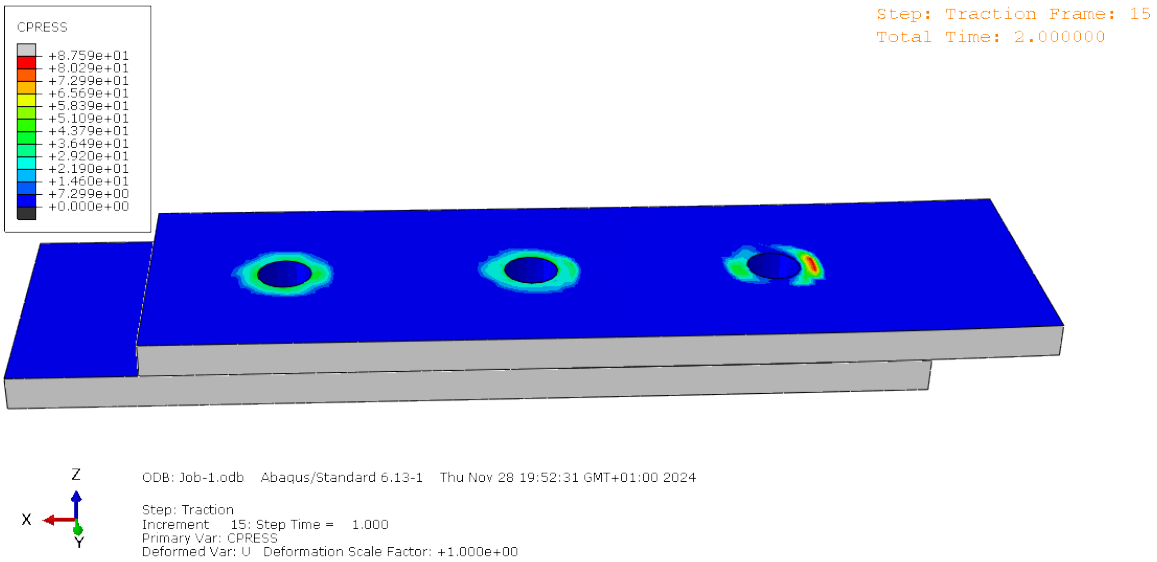
Figure 4.17: Stress contour map for models 14 and 1

Let's start with model 14, when no clearance is present. Notice that when since there is not an applied force in the region between the clamped section and the inner rivet position, the angle of the beam should be the same for all the cross-sections inside that region. Therefore, if the slope of the beam is close to be constant in that region, when incrementing the distance between the clamped section and the first fastener we will be thus increasing the vertical displacement of the beam Consequently, this will cause a stronger collision between the rivet head and the upper plate, and therefore, more stress will be concentrated in that rivet head area. This fact can be seen in both figure 4.17 and 4.18, where maximum stress values are registered

for the model with larger value of o_x.



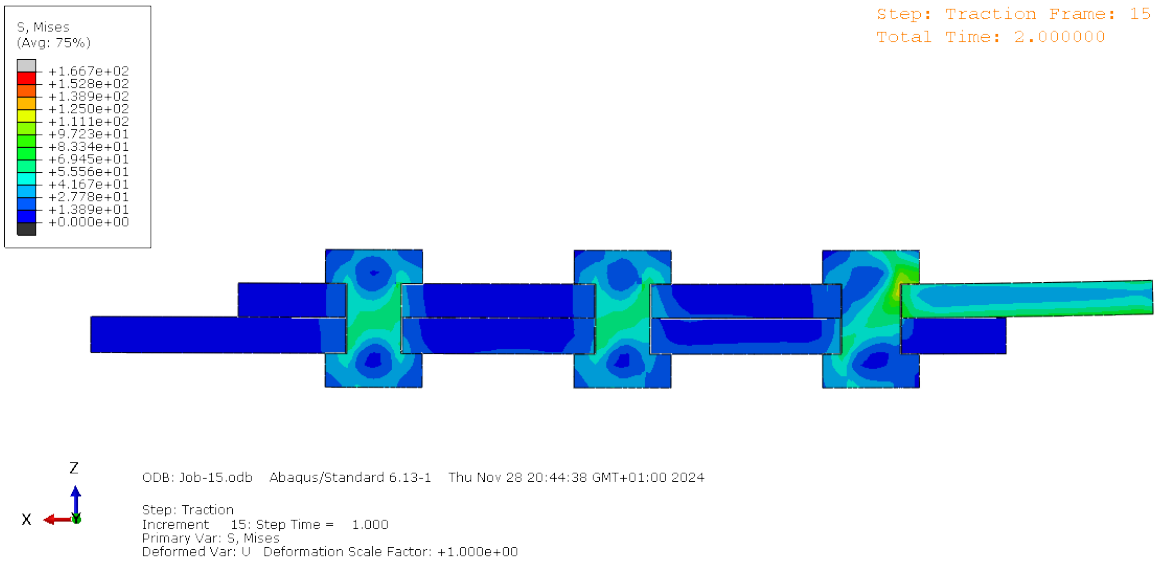
(a) CPRESS contour map for model 14



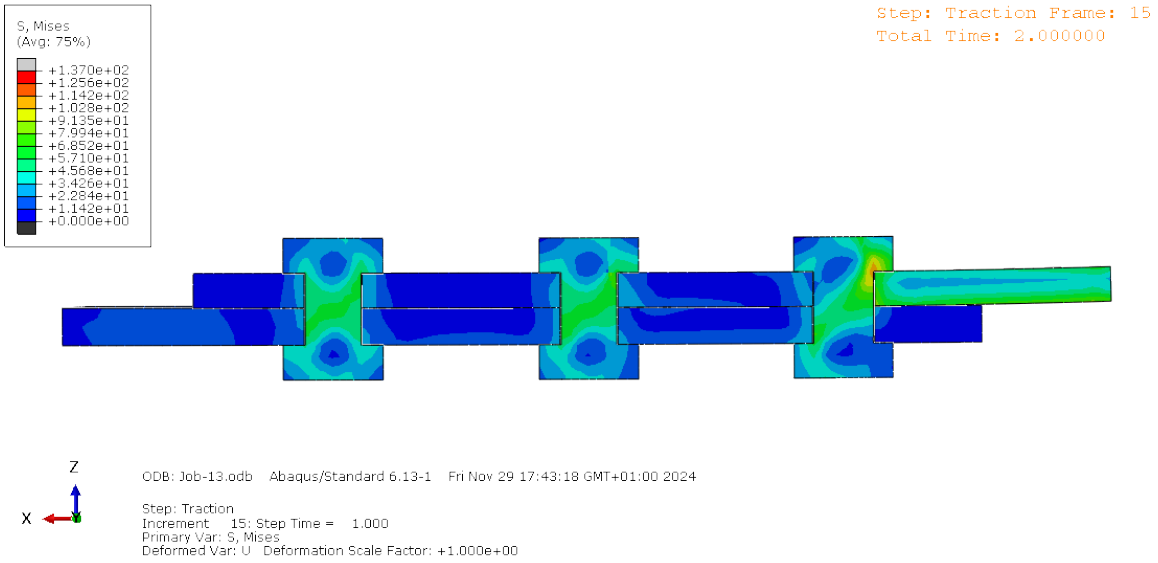
(b) CPRESS contour map for model 1

Figure 4.18: CPRESS contour map for models 14 and 1

However, in model 15 we have the rest of the effects previously analyzed present. Among these parameters, the hole clearance is the one which makes different model 14 and 15 in terms of stress distribution. Whereas for model 14 we can see a clear increase in stress concentration in the rivet head, for model 15 this does not happen because the gap between the hole and the fastener makes the stress to concentrate in the upper part of the shank. Therefore, like it happened before with the diameter effect, when clearance exists the effect of the parameter o_x is counteracted. This can be proven by looking at the contours 4.19 and 4.20, where when looking at the legend, it can be checked that the stress values for all the instances are very similar between these two models.



(a) Stress contour map for model 15

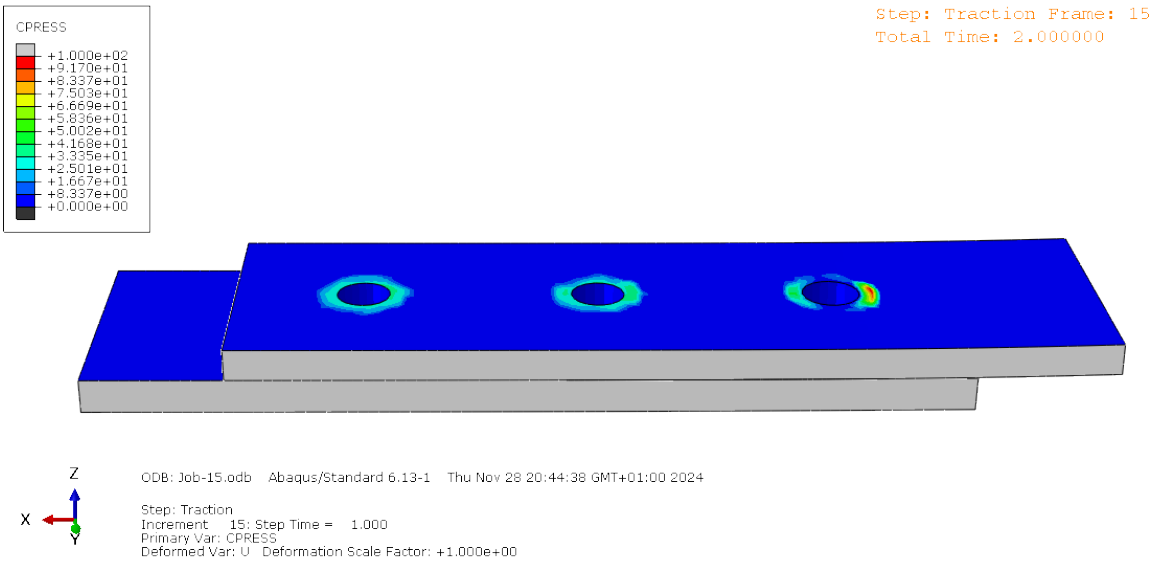


(b) Stress contour map for model 13

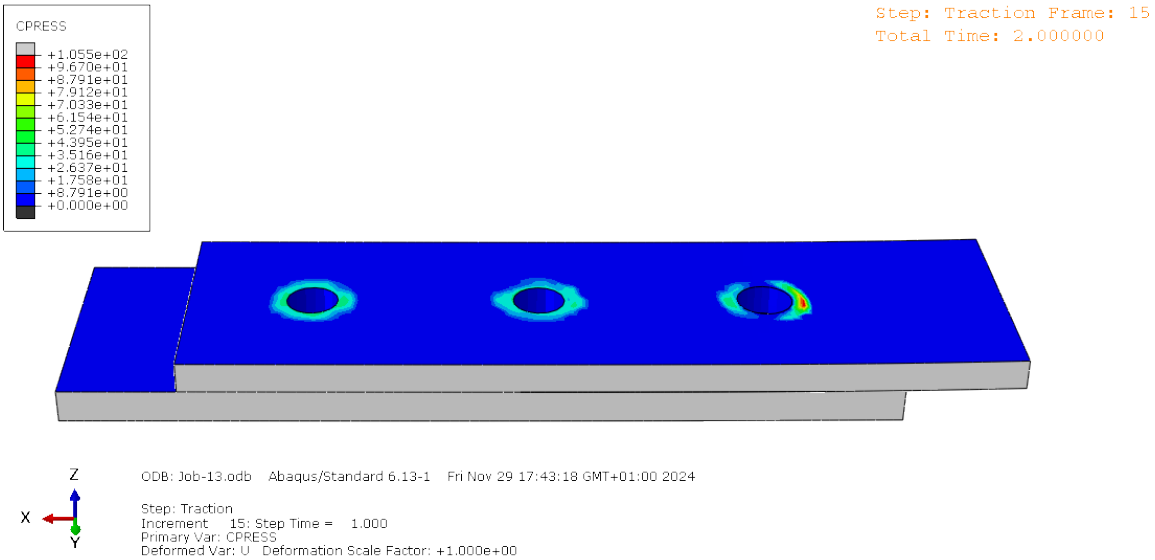
Figure 4.19: Stress contour map for models 15 and 13

Therefore, in order to sum up the general effect of the parameter o_x, we can state that, by incrementing its value we will increase the vertical position of the fasteners and so favoring the head - plate interaction. Nevertheless, when a clearance appears, the angle between the plate and the rivet are no longer coupled and the critical region becomes the upper part of the rivet neck, as all the cases with clearance.

Despite the effects caused by changing o_x, the influence of this variable is once again very small compared to the gap effect in terms of rivet shear loading, as it can be seen in the results shown in table 3.3.



(a) CPRESS contour map for model 15



(b) CPRESS contour map for model 13

Figure 4.20: CPRESS contour map for models 15 and 13

With this we would end up with the discussion about the rivets shear load distribution. However, another important magnitude in this project that can be analized is the Von-Mises stress for each of the rivets. In fact we are not interested in the Von-Mises stress of all the rivet nodes but most importantly, we will focus on the highest value. This is because this stress has a very important implication regarding possible fracture and material failure. In this project no damage tolerance analysis is applied but if we select the highest Von-Mises stress suffered in each one of the rivets, some information regarding its behaviour under fatigue can be also extracted.

These values can be extracted directly from Abaqus contours. In fact, notice that one of the contours previously extracted in Abaqus (the one with the cross-section cut) represents the Von-Mises stress contour. This, apart of being useful to explain the rivet shear load distribution it is also important to provide information regarding failure.

Starting with model 1, it can be seen that the highest Von-Mises stress is, as expected, located at the top part of the shank which belongs to the inner rivet, as shown in figure 4.21.

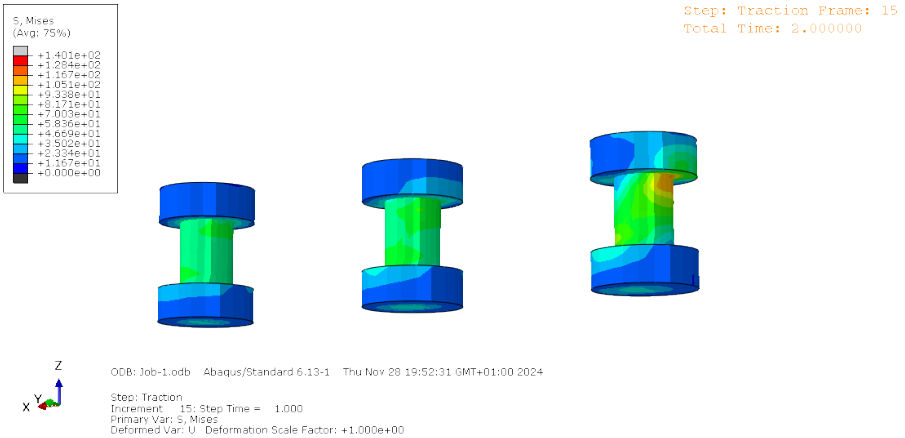
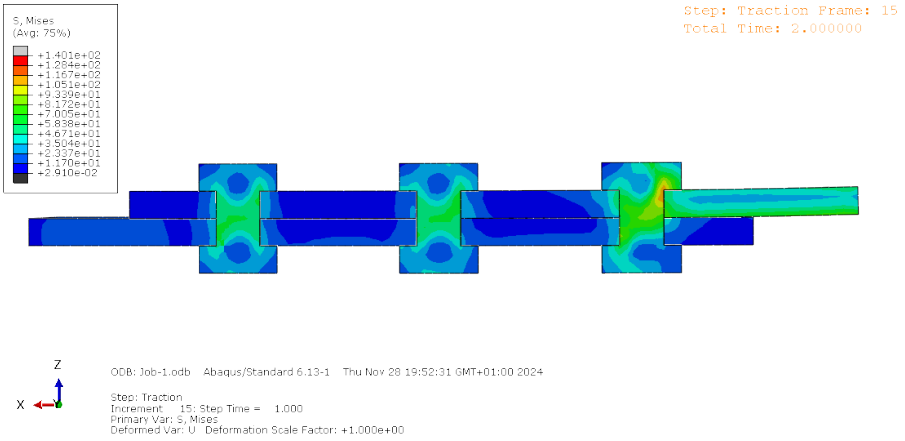
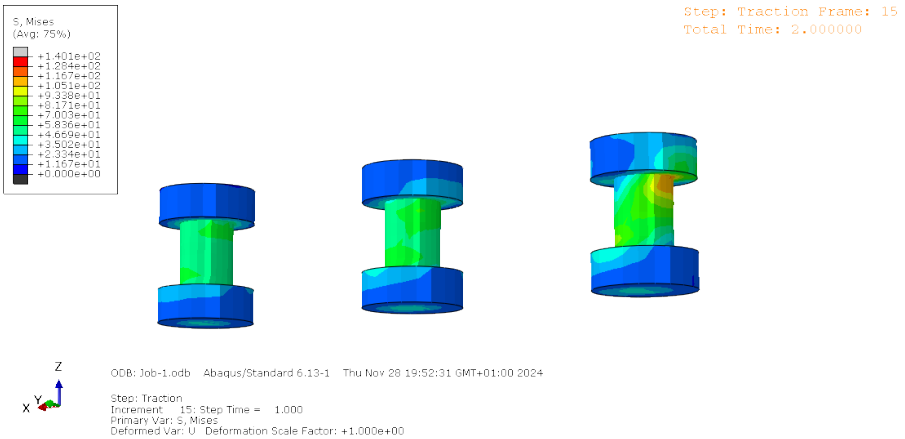


Figure 4.21: Von-Mises stress distribution for model 1

In addition, because of the symmetry of the problem, the highest Von-Mises stress will be located not only in the inner rivet, but also in the plane $y = 0$. This means that the highest value of the Von-Mises stress for each model should be captured inside the contours provided along this section. We can see this for instance in model 1, where this last image and figure 4.22(a) provides the same result.



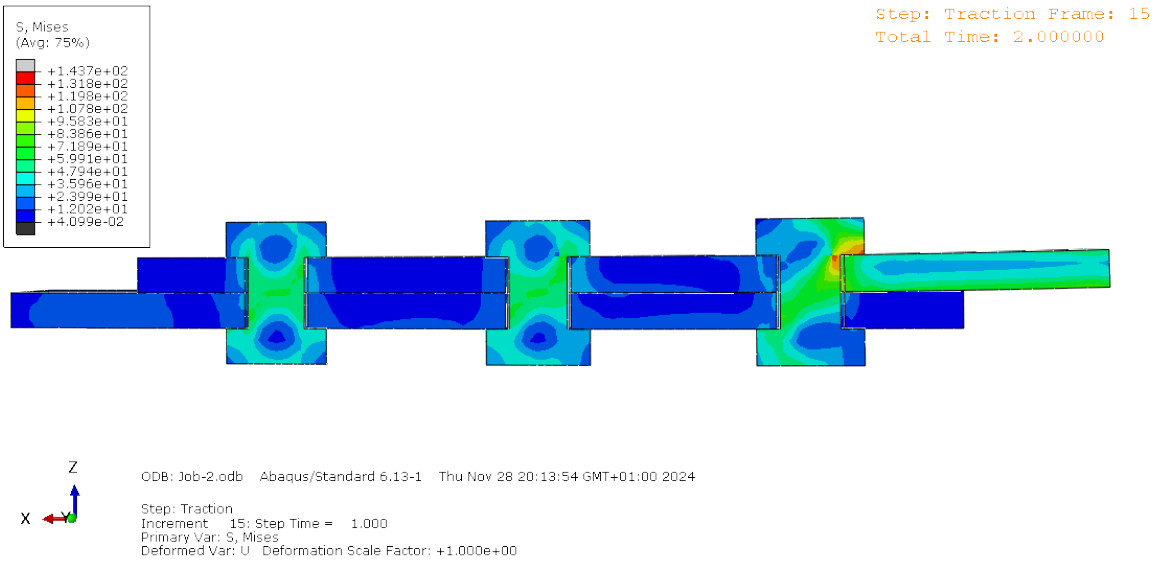
(a) Von-Mises stress distribution contour at y=0



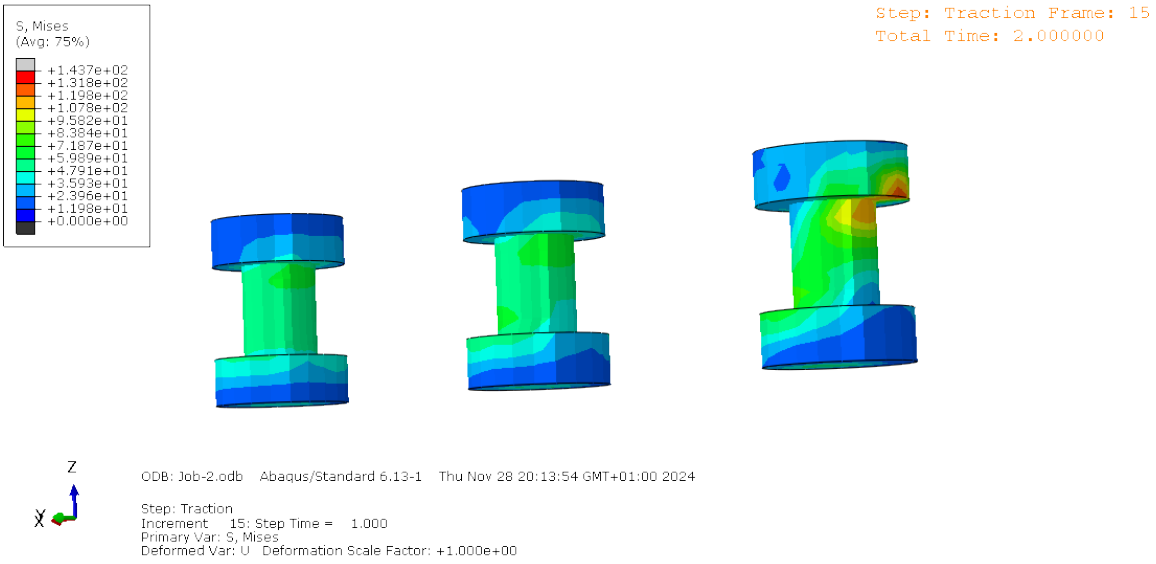
(b) General Von-Mises stress distribution contour

Figure 4.22: Von-Mises stress distribution contour for model 1

The same thing happens with models with clearance, such as model 2. This is observable in figure 4.23, since both contours give the same highest Von-Mises stress.



(a) Von-Mises stress distribution contour at y=0



(b) General Von-Mises stress distribution contour

Figure 4.23: Von-Mises stress distribution contour for model 2

Therefore, we can make a table collecting the highest Von-Mises stress for each of the models, in order to afterwards compare which is the effect of clearance in rivet stress concentration. These results are collected in table 4.4, where each model appears with each corresponding peak in Von-Mises stress.

Highest Von-Mises stress for each model (MPa)	
Model	Maximum value
1	140.1
2	143.7
3	144.6
4	131.8
5	147.1
6	201.1
7	137.2
8	139.9
9	136.2
10	142.91
11	143.4
12	200.2
13	137.0
14	154.4
15	166.7

Table 4.4: Design variables nomenclature

In order to understand the effect of clearance in stress concentration, the models will be compared in such a way the only difference between them is the fact that clearance exists. This way we can make the following groups:

- **Model 1 and 2**: model 2 is the same as model 1 but this last does not have clearance. This gives information about the clearance effect when all diameters are the same.

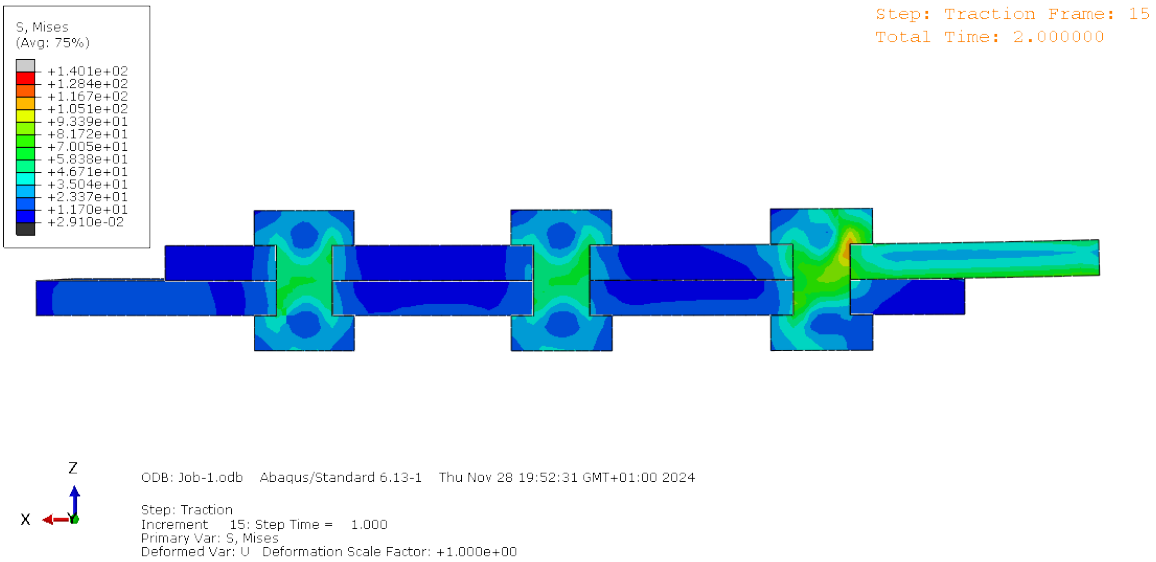
- **Model 6 and 9**: model 9 is the same as model 6 but this last does not have clearance. This gives information about the clearance effect when inner rivet is bigger than the rest of them.

- **Model 7 and 10**: model 10 is the same as model 7 but this last does not have clearance. This gives information about the clearance effect when middle rivet is bigger than the rest of them.

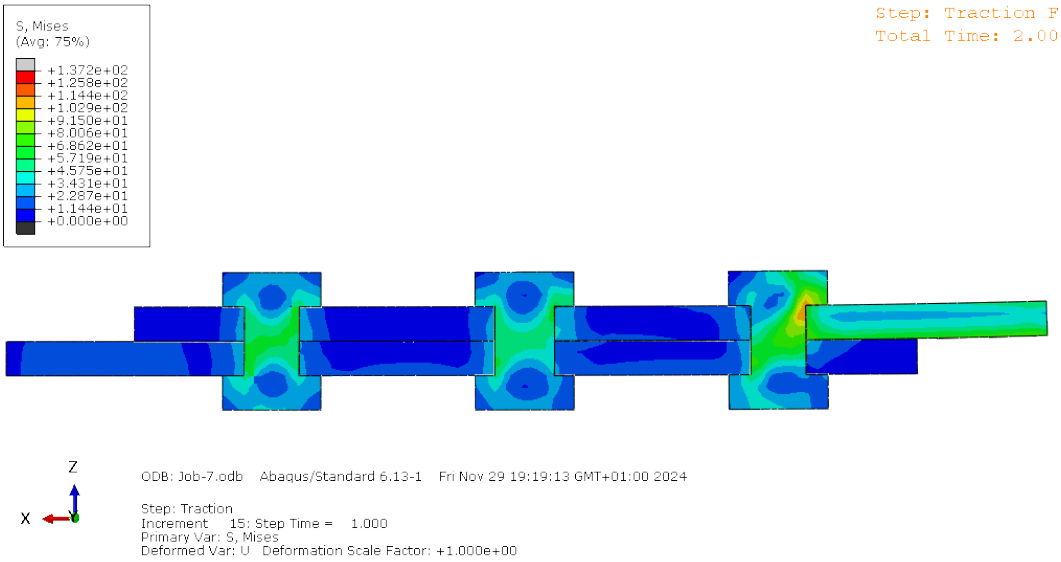
- **Model 8 and 11**: model 11 is the same as model 8 but this last does not have clearance. This gives information about the clearance effect when outer rivet is bigger than the rest of them.

We can check that, when all diameters are the same, the Von-Mises stress is larger for the case with clearance. As well this happens when the center or the outer rivet is the one which is larger. However, when the inner rivet is the one which is larger, the clearance will reduce the maximum stress.

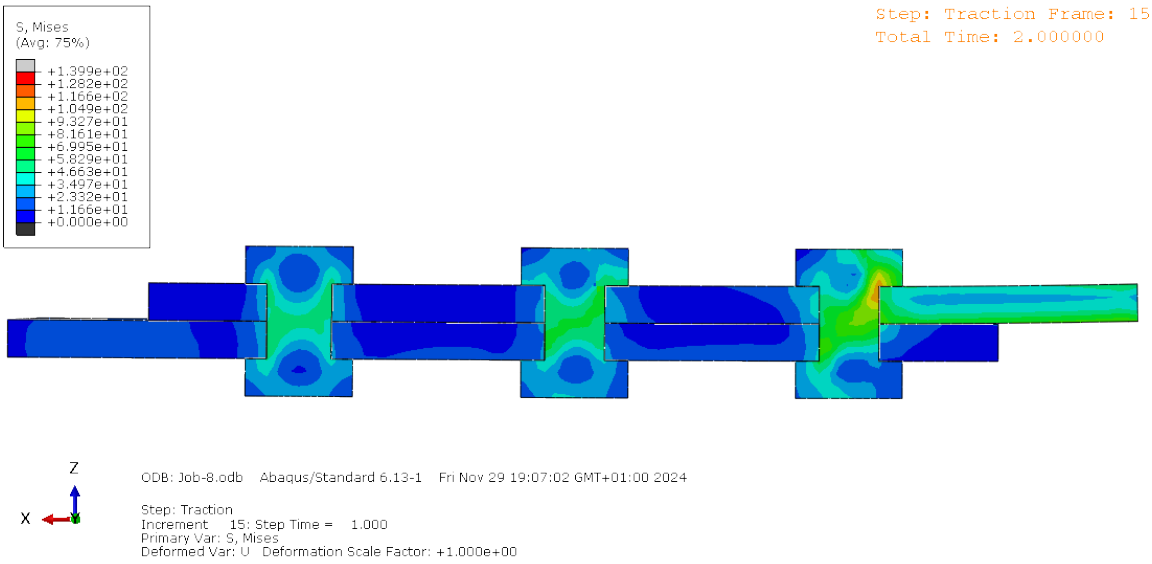
To understand why this happens, it is very relevant the information provided by the contours. For model 1, 7 and 8 (all of them models without clearance), as seen in figure 4.24, the highest stress is located in the region of the shank, close to the head, but still in the part colliding with the hole. For these three cases, we see that implementing a clearance will lead to a higher Von-Mises stress.



(a) Von-Mises stress distribution for model 1



(b) Von-Mises stress distribution for model 7

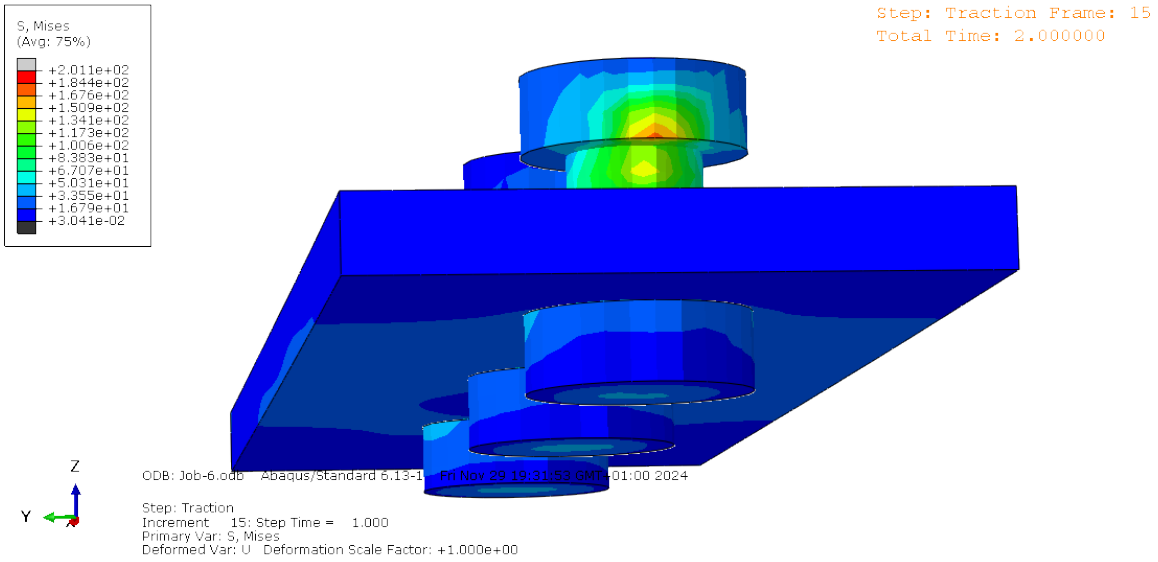


(c) Von-Mises stress distribution for model 8

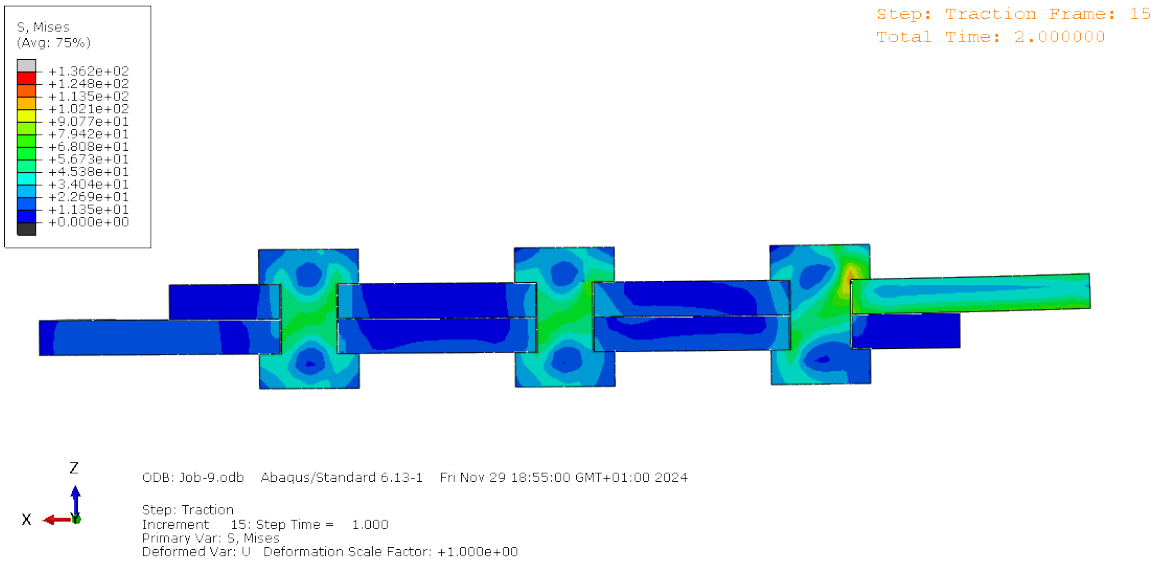
Figure 4.24: Von-Mises stress distribution contours

This occurs because the clearance, as stated in article "*The structure of the strength of riveted joints determined in the lap joint tensile shear test*", increases the stress in the shank region close to the head, meaning that the if that region turns to be the one with the highest Von-Mises stress, this maximum value will therefore be increased. In fact, this result is the one retrieved also from article "*The effect of rivet gun operating pressure and hole clearance on rivet shear strength in sheet metal riveting process*" where, as it happens in these cases, the maximum stress is increased.

On the other hand, when implementing a clearance in model 6, the maximum stress is reduced. Once again. in the contour shown in figure 4.25, it can be seen that before clearance (figure 4.25(a)), the critical region is not the shank anymore but the rivet head. Therefore, since the clearance makes the previously mentioned region of the shank to be more loaded, this makes other regions such as the rivet head be more relieved, reducing thus the maximum stress value.



(a) Von-Mises stress distribution for model 6



(b) Von-Mises stress distribution for model 9

Figure 4.25: Von-Mises stress distribution contours

Therefore, we can state that clearance makes the stress to be more concentrated in the region where rivet

head and shank are joint together. **For most of the cases**, where the angle difference between the rivet and the plate is almost null, the critical region will be the one mentioned before and thus, **clearance will increase the maximum Von-Mises stress**. As a consequence, material failure will occur earlier, as anticipated in article "*Effect of the rivet-hole tolerance on the stress-severity factor*", mentioned in the state of art.

However, for certain cases that cause the rivet shank to not be the most critical region, this stress concentration effect caused by clearance can deviate the stress from the zone where originally material was more loaded, and therefore causing a reduction in the maximum Von-Mises stress. Nonetheless, notice that these cases are residual and are much less common.

4.2. Project conclusions

The aim of this project, as specified in chapter 1, was to analyze what were the consequences of leaving a gap between the rivet shank and the hole drilled, since this was beneficial for the manufacturing process of the fasteners installation but it may had some harmful effects in terms of performance. Consequently, the 15 models developed during these project covered up this hole clearance effect for several scenarios where conditions were different to check whether the relevance of this effect was dependent on some other factors.

After analyzing the 15 single shear models, the following conclusions can be made:

- **All the studied aspects had an impact in the stress distribution along the model profile**. However, the relevance of each of these parameters may vary depending on the case.
- **Among all of them, the hole clearance effect is the one which dominates**, being able to reduce the allowable load that can be carried by more than 20% with respect to the case where no gap exists. As seen in the contours, when a gap is added the stress contours follow a determined distribution, and that same pattern is observed in all cases with hole clearance despite the fact that other variables may appear or not, and that is the reason why this effect is dominant compared to the rest.
- **Increasing the shank diameter will make the rivet absorb more load but as well it can make more vulnerable the region where rivet head and plate interacts**. Depending on the fastener position and the amount of shank increment this can be either beneficial or harmful. Notice however that during these analysis the rivet head diameter, which as discussed in the results analysis, has influence in this kind of cases, has not been changed. Therefore, as another outcome of this project, for future lines of investigation, the rivet head diameter could be considered as a design variable. This way the exercise could be improved and more valuable information could be extracted.
- **Having different thicknesses will shift the pretension plane towards the head or the collar of the rivet, making thus stress to concentrate more in the are closer to this pretension plane**. This change in the stress distribution will make it to deviate from the ideal case where pretension plane is half-way of the shank, thus making the maximum stress until failure lower, whether this pretension point is moved forward or backwards along the shank.
- **Plate offset increase the distance between the critical point ant the clamped region, promoting hence the rivet head - plate interaction**. This makes both rivet head and upper part of the shank the critical regions where failure occurs, being the regions where stress concentration is higher.
- **When all of these effects are present, hole clearance effect dominates and dictates model behaviour and both stress distribution and shear load distribution**. All phenomena have been represented in our models for several scenarios, but each one of them have been presented with clearance and without it. This has been done on purpose so it is clearly reflected that the dominant effect is the hole clearance.
- **Clearance increases the stress in the region where rivet shank and head are joint together**: In most of the cases, since this corresponds to the critical region where failure occurs, clearance increase the maximum Von-Mises stress of the model which leads to an earlier failure.

Talking more specifically about the hole clearance effect, this project was not able only to understand its consequences, but as well, in this paper it could be found a first estimation of the amount of shear load capacity can be lost in the first rivet which is the critical. Actually, based on the results of this projects it indicates that the hole clearance relation with the shear load reduction is not any close to linear. This comes from the result that when creating a small clearance, the shear load reduction is very large, but when incrementing this initial gap times 6, the maximum shear load is reduced by a much smaller amount of force.

As future lines of investigation, since it has been proved that the hole clearance effect is dominant with respect to the other ones studied, this exercise can be repeated considering the rest of the design variables left out of the DOE and perform the same analysis. As well, other interesting possibility could be understanding more the shape of this hole clearance - maximum shear load relation by studying more points along this curve for many different conditions to cover a matrix of cases which can lead to relevant conclusions. One way or another, this work can be set as a basis for future projects in order to work deeper in one of the multiple points covered in this document.

A

General steps followed to create a model in Abaqus in detail

Abaqus CAE is structured in such a way it leads the user through the main steps that are required to be done in order to build a model.

When opening the program, it can be found a module window which indicates the subsequent fields that need to be completed by the user in order to build the model.

These modules can be seen in figure A.1 and for more clarity are listed below:

1. **Part**: First step to create the model is defining the geometry of the different elements that conform the assembly. Thus, different tools are available for the user to create a wide variety of shapes in order to conform the necessary parts.

2. **Property**: Once the geometry of the parts has been set, the next thing to do is to assign the properties to these parts which will describe their behaviour during the execution. This aspect is highly related to the material used and the sections created with them inside this module.

3. **Assembly**: Third step is to assemble the parts conforming the model, either by rotating the different items or translating them. In this module it is possible to declare different instances inside the assembly which may be convenient for future steps.

4. **Step**: This module is used to divide the multiple events inside our project into time steps. As well, each time step is assigned with solving properties such as maximum number of iterations per attempt or the range of increments which can be made.

5. **Interaction**: Along our assembly there will exist certain interfaces which will play an important role and hence, it is important to define their properties so the software knows how to evaluate them. Most common interaction is the contact, but as well, inside each kind of interaction, there exist many parameters that can be assigned starting from trivial parameters reaching up to more complex ones.

6. **Load**: Once the different time steps have been created, the loads may be defined in any of these steps. There exists a wide variety of loads, and this module allows not only to define the loads but as well indicate how this load will be propagated along time.

7. **Mesh**: Defines the mesh strategy, as well as the size of the mesh among other parameters. It is highly relevant to make a proper balance between accuracy and computational cost in order to finally arrive to both model convergence and coherent results.

8. **Job**: Having set the prior modules, the last step is to let the software calculate the behaviour of our system based on the data we entered previously. Inside this environment, user is provided with a monitor which displays a table containing important data after every calculation attempt, which indeed is useful to identify where could be a problem in case the model does not come to a successful ending.

9. **Visualization**: If user manages to reach a successful calculation by the software, this tool will show the different outputs defined previously by the user, being able to handle colour contours, simulation velocity or amplification factors to appreciate small deviations. This one is useful to corroborate that the results indeed make sense.

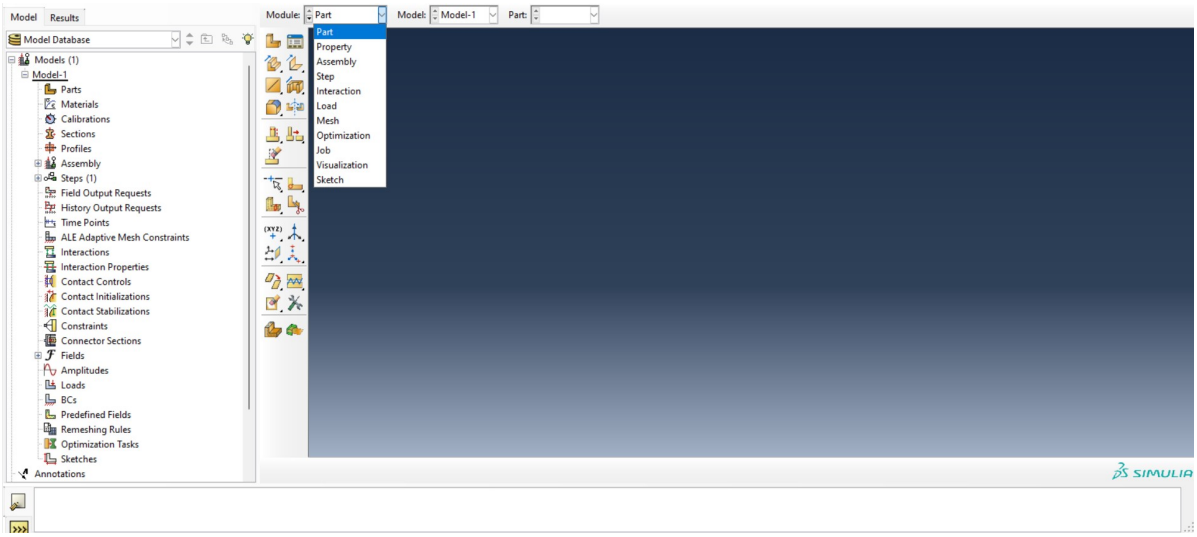


Figure A.1: Modules inside Abaqus CAE

After briefly describing all the modules which take part in the model construction, next task was to construct a sample model that somehow represented a similar load case to the one desired to be parametrized.

This descriptive appendix will go through the building process following the path through the modules listed above. But firstly it is convenient to create a sketch showing the different instances that will conform the model, which can be found in figure A.2, represented hereafter.

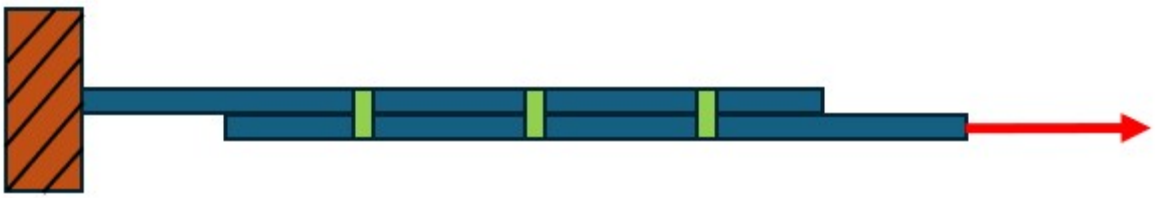


Figure A.2: Model sketch

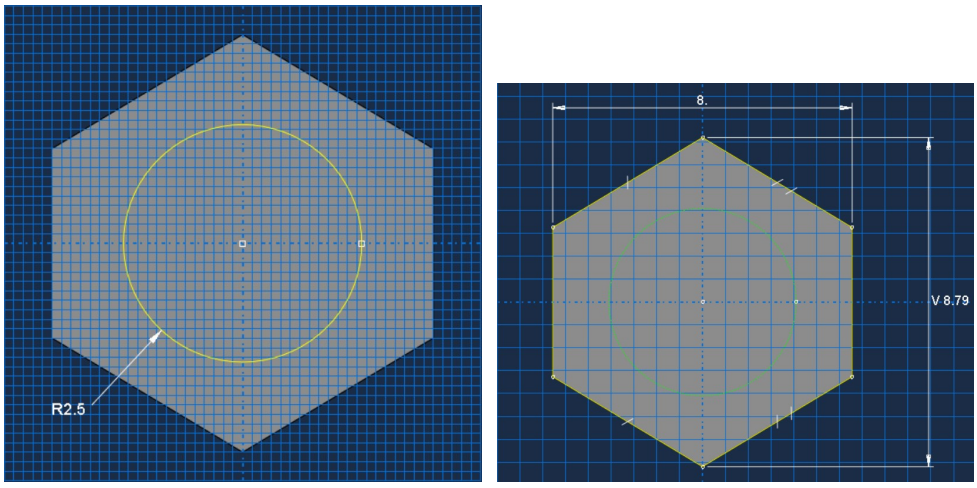
The top plate will be clamped to the wall at one of its edges, while the bottom plate will be joined to the first one by using three rivets, represented in green. These last will be installed by interference, thus making the shank diameter and the one of the threaded holes coincide. On the other end of the bottom plate, a certain traction load will be applied, trying to separate the metal plates. Therefore, first thing that would be made is the rivets pretension to assemble the two plates and, afterwards that force would be applied to the bottom plate, which will try to expand towards the direction of the force.

After giving a short description of the model, it was time to start creating the model itself. As mentioned previously, the project was constructed filling the data in the corresponding modules following the order described above:

1. **Part**: There are two different parts that need to be created to build the case, which are the rivet and the plate.

   Starting with the rivet, it is composed by three different extrusions, the first one conforms the shank, whereas the other correspond to both the head and the collar of the rivet. For simplicity, both the head

and the collar were modelled the same way and consisted on an hexagonal head prism. The sketches for the three extrusion can be found in the following figure A.3:



(a) Shank sketch (b) Head and collar sketch

Figure A.3: Sketches that conform the rivet

For the case of the plate it was easier to build it, since only one extrusion was necessary. It was only required to dimension the width and length of the plate, as well as placing the three holes where rivets where going to be placed.

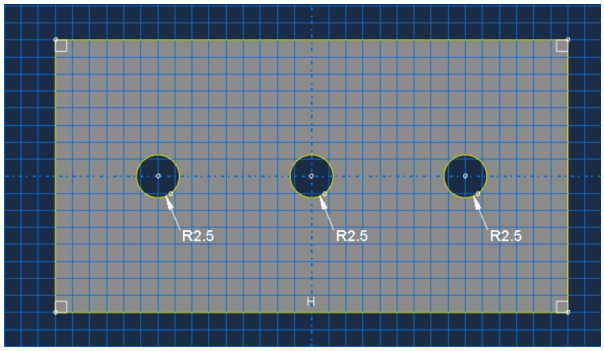
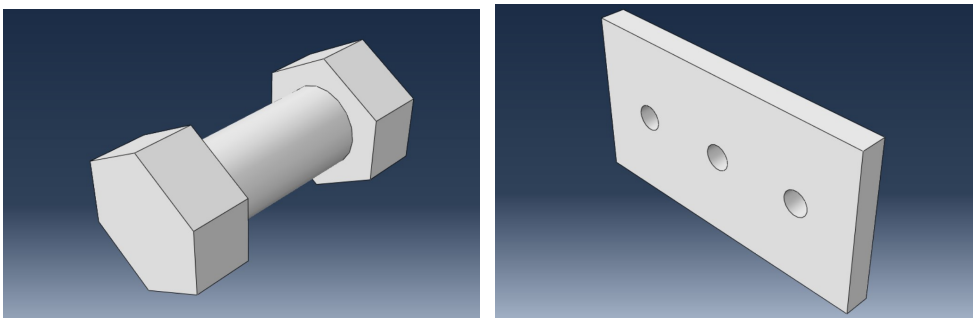


Figure A.4: Plate sketch

Performing the extrusion of the sketches shown before, the two parts are obtained:



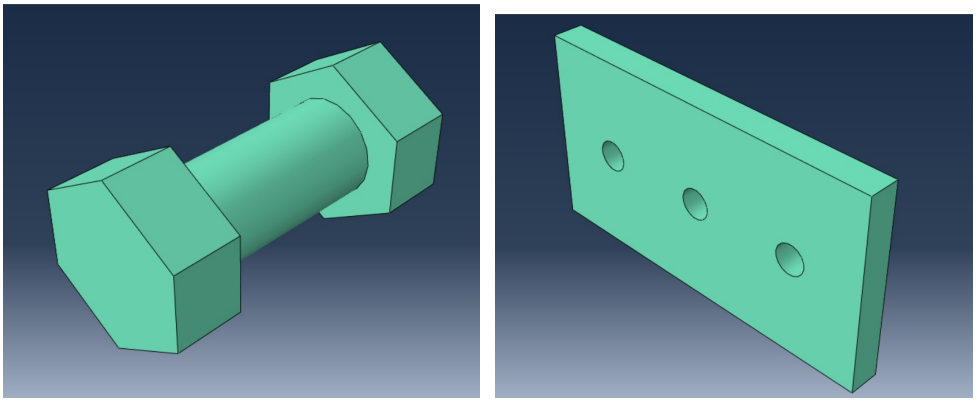
(a) Rivet (b) Plate

Figure A.5: Parts of the model

2. **Property**: After creating the parts shown in figure A.5, next step is to assign the material and, consequently the section to these elements. For this particular case, the properties assigned to the parts were [23]:

   - Density of 7900 kg/m^3
   - Young Modulus of 198.5 GPa
   - Poisson Ratio of 0.3

   By introducing these amounts the material created was assigned with elastic behaviour, but not plastic. Since it was the first sample model, for terms of convergence only these values were defined. After creating that material, a section was linked to this material properties so afterwards, this section could be added to the two different parts. If done correctly, the parts should appear with a green colour on their surface as shown in figure A.6



(a) Rivet (b) Plate

Figure A.6: Parts of the model with assigned section

3. **Assembly**: Next step is to create the assembly of the model. Firstly, the three rivets and the two plates are added as independent parts, and once these have been added they can be translated by moving them to certain specific locations, making the two plates collide by a mating surface and making the three rivets coincide with the three holes of the plates, respectively. By default, the software allows the user to utilize the center points of the circles of the corner points of the prisms to rotate and move the different parts.

   As mentioned previously, it is interesting to divide each part into instances since each of these instances will have a different type of contact and hence, it is interesting to differentiate them. In the case treated

in this paper, the rivets were divided into four different instances which are the head, the collar and the shank, this last divided in two making the total of four. This last division is not that useful for the interaction definition, but for the pretension load creation.

After performing these tasks one can arrive to the following assembly, shown in figure A.7.

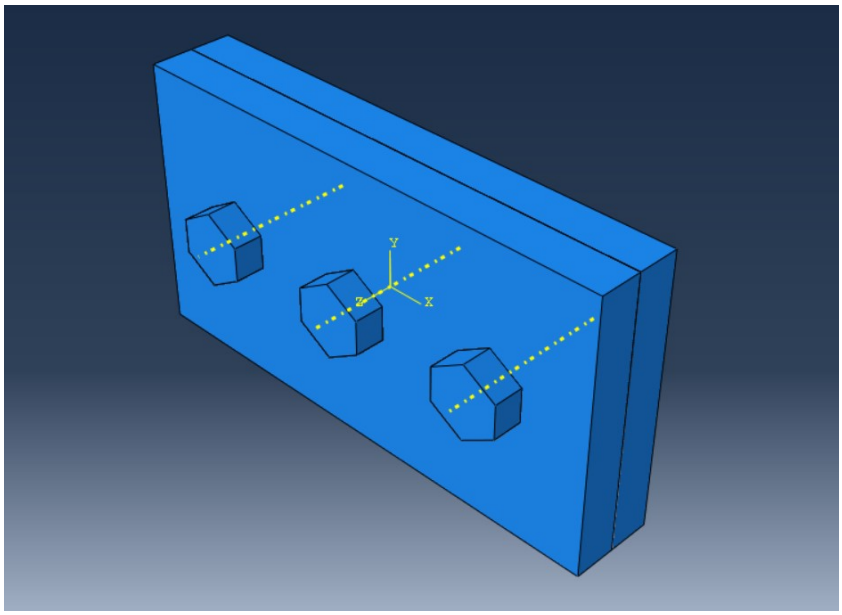


Figure A.7: Model assembly

4. **Step**: In this job there exist three steps. The first one it is created by default since the system requires it to run the model and it is the initial one. After it, the other two steps are added. The step following the initial one is called pretension and will include the pretension loads applied in the rivet shank cross-section, which will simulate the riveting of the three positions. After this step ends, meaning that the plates have been joined by this mechanical force, the traction step will start applying the force in the bottom plate edge.

   The last two steps are static steps which have the following properties:

   - Maximum number of increments: 10^6
   - Initial increment size: 0.01
   - Increment size range:$(10^{-4} - 0.1)$

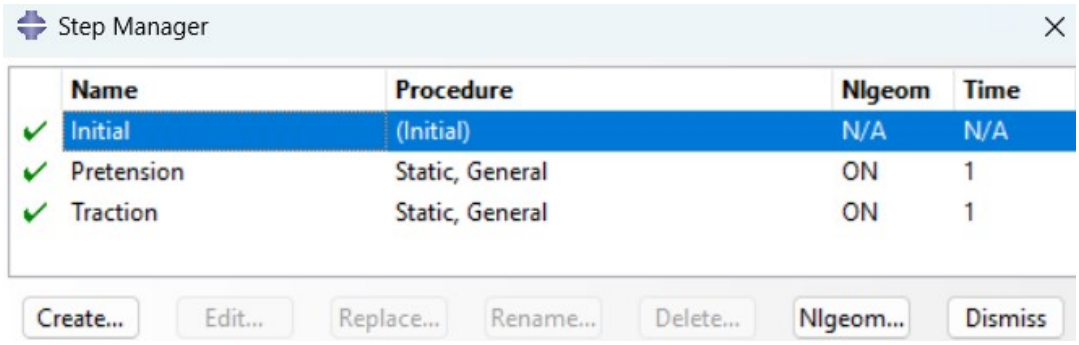


Figure A.8: Step sequence

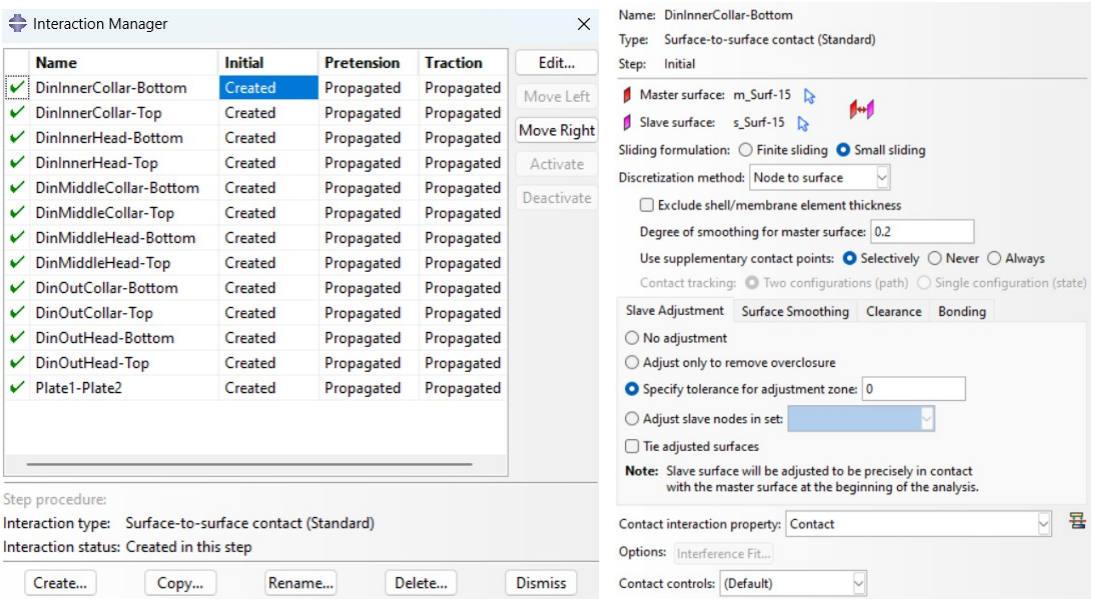
5. **Interaction**: Inside this module, first thing to do is to define the kind of interactions that exist in the model. In this particular case, there is only one which is the physical contact between the mating surfaces. This contact interaction eventually can be completed with many types of data, but for simplicity,

only tangential and normal behaviour were added, introducing among other factors the friction coefficient.

With the contact property already defined, this property has to be assigned wherever it applies. In the model shown in figure A.7 there exist 13 interactions:

- 6 of them are the contact between the plates and the head/collar of the rivet.
- Other 6 constitute the contact between the shank and the holes of the plates.
- The last one is the contact between the two plates.

For each of these interactions it is required to define the master and slave surfaces, the adjustment parameters and the clearance settings, among others.



(a) List of 13 interactions

(b) Settings for each interaction

Figure A.9: Interaction module

6. **Load**: In this module the loads for pretension and traction are created in its corresponding step. Since these are two different types of load, they are created differently:

- Pretension: Inside Abaqus CAE the pretension can be simulated by a bolt load. For this bolt load to be defined the program needs a cross-section of the rivet to apply the force, the magnitude of that pretension force and lastly the axis of the rivet. After giving these three parameters the load can be easily created, as shown in the figure A.10

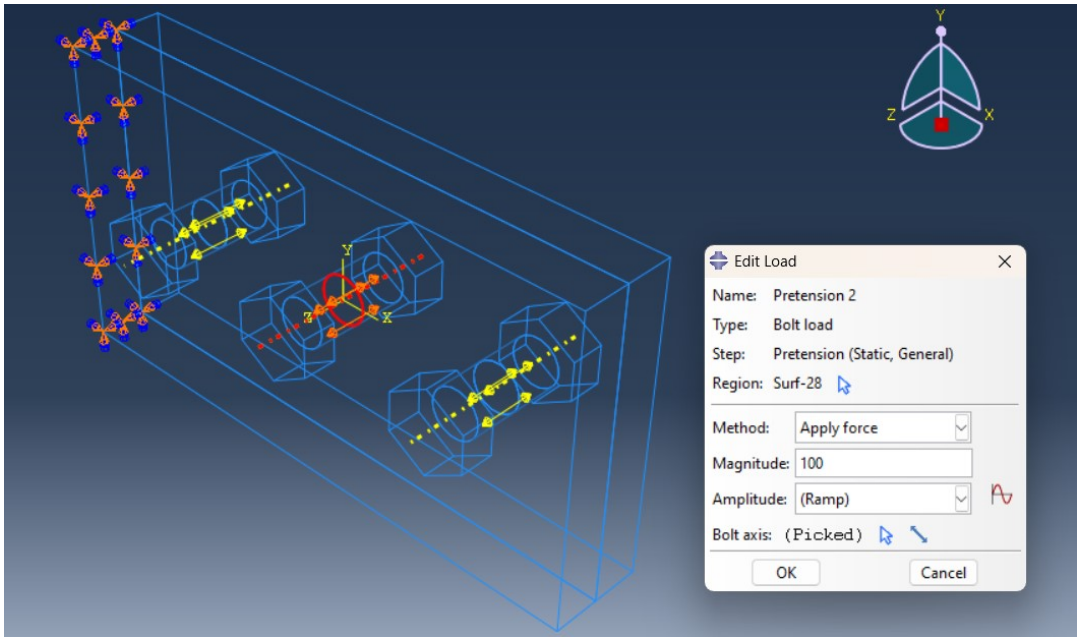


Figure A.10: Magitude, surface and axis that define the bolt load

- Traction: In order to create a traction force it is firstly required to create a reference point from which the pull will come. After adding a kinematic coupling between the end face of the bottom plate and the reference point, situated in this last mentioned face, a magnitude and a direction can be given to create the traction force.

Lastly, in this module the boundary conditions are also assigned. As it can be seen in figure A.10, the clamped edge is represented by the blue and orange triangles which prevent that face of either rotate or translate.

7. **Mesh**: Given the fact that there exist axial symmetry in the model, a good strategy to place the seeds in the parts is by sweep around the axis of symmetry, which can be selected in the window shown in figure A.11.

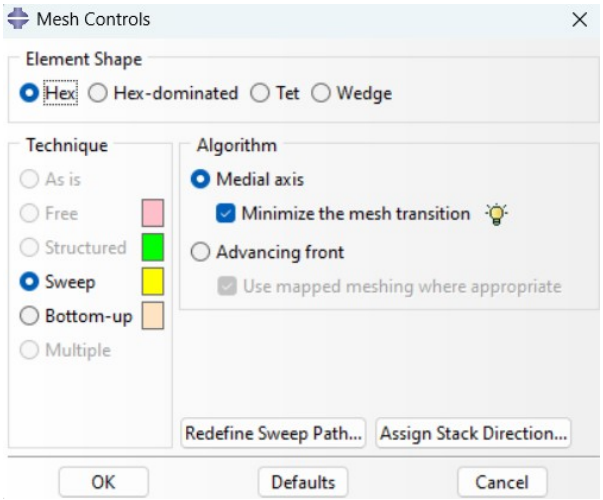


Figure A.11: Mesh strategy definition

After selecting the meshing strategy, next thing is to define the size of the meshing elements. For convinience, it was used a global size for the entire model to create a uniform mesh. Once this has been set the user can proceed to mesh the instances.

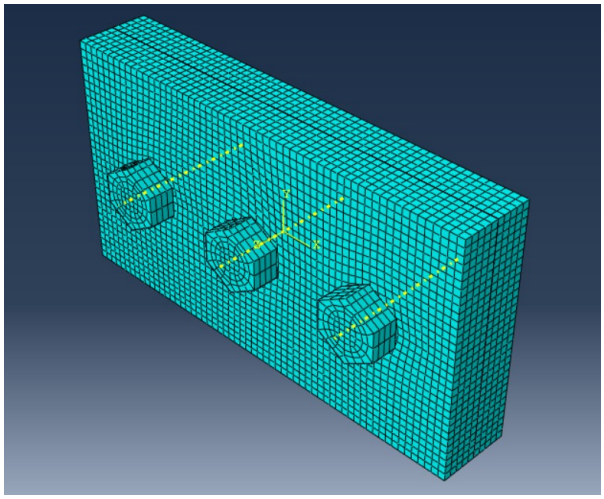


Figure A.12: Mesh of the model

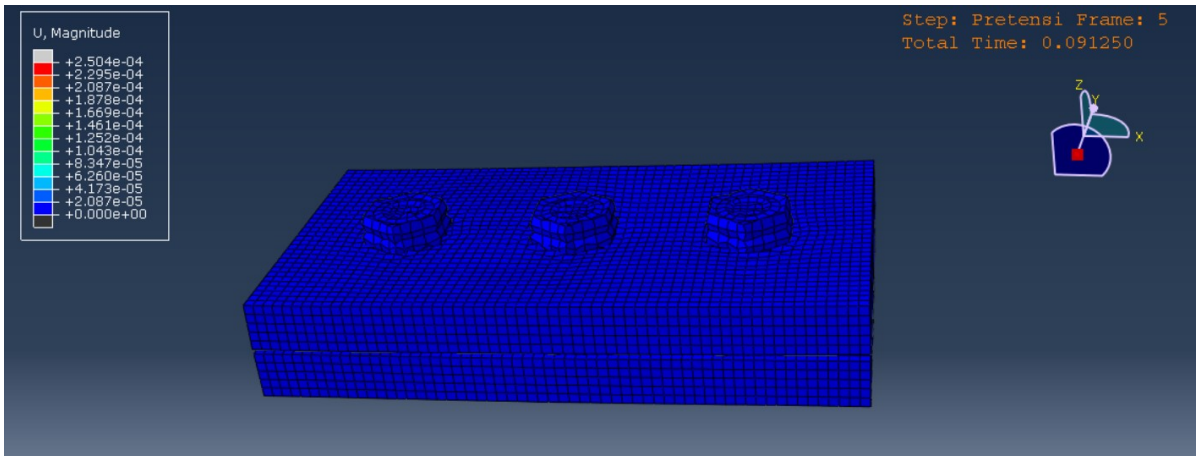
8. **Job**: The user uploads the model for analysis so that the software can calculate the stresses and displacements during the entire simulation. As mentioned previously, the monitor shows a table with information regarding convergence and stability of the interactions.

Job: Job-1 Status: Completed

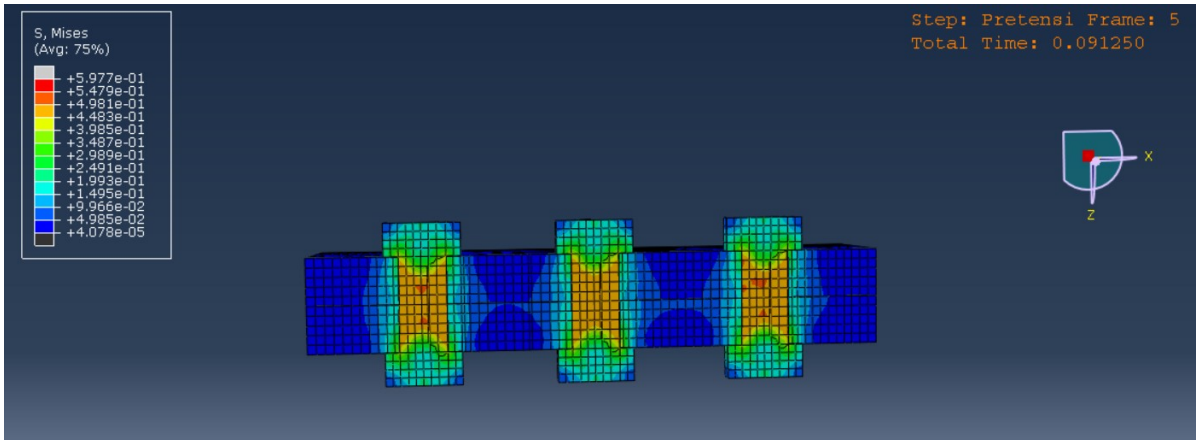
Step	Increment	Att	Severe Discon Iter	Equil Iter	Total Iter	Total Time/Freq	Step Time/LPF	Time/LPF Inc
1	1	1	4	1	5	0.01	0.01	0.01
1	2	1	2	1	3	0.02	0.02	0.01
1	3	1	0	1	1	0.035	0.035	0.015
1	4	1	1	0	1	0.0575	0.0575	0.0225
1	5	1	1	1	2	0.09125	0.09125	0.03375

Figure A.13: Job monitor

9. **Visualization** Last part is to check and verify the contours of the model. From this module many pictures and videos can be extracted which are useful to help the user understand the reason behind the model behaviour.



(a) Augmented deformation of the rivet head



(b) Cross-section stress distribution

Figure A.14: Model results

B

Script used to generate the rpy files

In this appendix it can be found the main script used during this thesis. The script is presented in such a way that it only requires to be copied and pasted in Python to be tested. In order to separate the entire script by its lines, some adjustments which are defined in the script are required to be done before running it. This script contains the information needed to create the rpy files, each of them according to the DOE table, which is shown in figure 3.30. To make the script work, it is only required to create an excel file called "Cases.xlsx" in the same folder as the script file and then all the files shall be generated.

Having this said, please find hereafter the code used to create the model files according to the table mentioned in previous paragraph:

#**Design variables**
#inner rivet shank diameter (green)
dr_s_1= 6.0
#middle rivet shank diameter (yellow)
dr_s_2= 6.0
#outer rivet shank diameter (red)
dr_s_3= 6.0
#inner rivet hole diameter
d_h_1= 6.0
#middle rivet hole diameter
d_h_2= 6.0
#outer rivet hole diameter
d_h_3= 6.0
#inner rivet head diameter
dr_h_1= dr_s_1+2.0
#middle rivet head diameter
dr_h_2= dr_s_2+2.0
#outer rivet head diameter
dr_h_3= dr_s_3+2.0
#upper plate thickness (blue plate which is clamped)
t_1= 2.0
#lower plate thickness (gray plate which is pulled)
t_2= 2.5
#offset in x-direction between plates
o_x= 11.2
#inner rivet pretension force
f_in= 1000.0
#middle rivet pretension force
f_cent= 1000.0
#outer rivet pretension force
f_out= 1000.0
#traction force

```
    f_trac= 10000.0
    #rivet head height
    hr= 2.0
    #mesh size
    m_size= 0.9
    #edge distance
    e= 2.0*dr_s_1
    #fastener pitch
    p= 4.0*dr_s_1



    def RpyFileCreator(dr_s_1, dr_s_2, dr_s_3, dr_h_1, dr_h_2, dr_h_3, hr, d_h_1, d_h_2, d_h_3, t_1, t_2, o_x,
e, p, f_in, f_cent, f_out, f_trac, m_size, iter):



    # With this line, the function begins.
    #Thus, when copying the following lines of the code, until it is specified otherwise,
    #it is required to tabulate to the right the following coding lines.

    #dummy variable used to replace the dots by comas
    #in during instance naming in order to avoid errors produced by Abaqus
    diameter = str(dr_s_1).replace('.', ',')
    #radius
    radius1=dr_s_1/2.0
    radius2=dr_s_2/2.0
    radius3=dr_s_3/2.0
    radius_hole1= d_h_1/2.0
    radius_hole2= d_h_2/2.0
    radius_hole3= d_h_3/2.0
    r_outer1= dr_h_1/2.0
    r_outer2= dr_h_2/2.0
    r_outer3= dr_h_3/2.0
    #z-coordiante in absolute value where traction force is applied
    z_point_force= t_2/2.0
    #cdistances used to introduce the parts when making the model
    #fasteners 2 y 3 (middle and outer)
    d2= 3.0*dr_s_1
    d3= 6.0*dr_s_1
    #plates
    d4=20.0*dr_s_1
    d5=40.0*dr_s_1
    #shank length
    shank_number= t_1+t_2
    #dummy variable used to replace the dots by comas in during instance naming in order to avoid errors
produced by Abaqus
    shank= str(shank_number).replace('.', ',')
    #coordinates used to create the plates
    plate_left_x= -(p + e)
    plate_right_x= (p + e) + o_x
    plate_top_y= e
    plate_bottom_y=-e
    #coordinates used to make the holes in the plates hole_1_x= -p hole_2_x= 0.0 hole_3_x= p
    #coordinates used to determine the radius of the holes.
    #Plate 1
```

```
    hole_1_d= -(p + radius_hole1)
    hole_2_d= radius_hole2
    hole_3_d= (p + radius_hole3)
    #Plate 2
    hole_13_d= -(p + radius_hole3)
    hole_31_d= (p + radius_hole1)
    #Other variables used to construct the assembly
    d4_d2=d4-d2
    d4_d3plushole3x= d4-d3+hole_3_x
    d4_hole3x= d4-hole_3_x
    d5_d4= d5-d4
    #iteration to name the files
    valor_fil = str(iter)

    filename = 'modelo_' + valor_fil + '.rpy'
    file = open(filename,'w')
    file.write('from abaqus import *'+ '\n')
    file.write('from abaqusConstants import *'+ '\n')
    file.write("""session.Viewport(name='Viewport: 1', origin=(0.0, 0.0), width=95.8541641235352, height=134.273147583008)"""
+ "\n")
    file.write("session.viewports['Viewport: 1'].makeCurrent()" + "\n")
    file.write("session.viewports['Viewport: 1'].maximize()"+"\n")
    file.write('from caeModules import *'+'\n')
    file.write("from driverUtils import executeOnCaeStartup" + "\n")
    file.write("executeOnCaeStartup()" + '\n')
    file.write("""session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues( referenceRepresen-
tation=ON)""" + '\n')

    file.write("session.viewports['Viewport: 1'].setValues(displayedObject=None)" + '\n')
    file.write("""session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues( referenceRepresen-
tation=ON)""" + '\n')
    file.write("s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=10.0)" + '\n')
    file.write('g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints' + '\n')
    file.write('s.setPrimaryObject(option=STANDALONE)' + '\n')
    file.write('s.CircleByCenterPerimeter(center=(0.0, 0.0), point1=(' + str(radius1) + ', 0.0))'+ '\n')
    file.write("p = mdb.models['Model-1'].Part(name='DIN660-" + diameter + '-' + shank + """-1', dimension-
ality=THREE_D, type=DEFORMABLE_BODY)""" + '\n')
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank +"-1']" + "\n")
    file.write("p.BaseSolidExtrude(sketch=s, depth=" + str(shank_number) + ")" + '\n')
    file.write('s.unsetPrimaryObject()' + '\n')
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-1']" + "\n")
    file.write("session.viewports['Viewport: 1'].setValues(displayedObject=p)" + "\n")
    file.write("del mdb.models['Model-1'].sketches['__profile__']" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-1']" + "\n")
    file.write("f, e = p.faces, p.edges" + "\n")
    file.write("""t = p.MakeSketchTransform(sketchPlane=f[1], sketchUpEdge=e[0], sketchPlaneSide=SIDE1,
sketchOrientation=RIGHT, origin=(0.0, 0.0, """ + str(shank_number) + "))" + "\n")
    file.write("""s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=15.75, gridSpac-
ing=0.39, transform=t)""" + "\n")
    file.write("g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints" + "\n") file.write("s1.setPrimaryObject(option=SUP
+ "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-1']" + "\n")
    file.write("p.projectReferencesOntoSketch(sketch=s1, filter=COPLANAR_EDGES)" + "\n")
    file.write("s1.CircleByCenterPerimeter(center=(0.0, 0.0), point1=(" + str(r_outer1) + ", 0.0))" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-1']" + "\n")
    file.write("f1, e1 = p.faces, p.edges" + "\n")
```

```
    file.write("""p.SolidExtrude(sketchPlane=f1[1], sketchUpEdge=e1[0], sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, sketch=s1, depth=""" + str(hr) + ", flipExtrudeDirection=OFF)" + "\n")
    file.write("s1.unsetPrimaryObject()" + "\n")
    file.write("del mdb.models['Model-1'].sketches['__profile__']" + "\n")
    file.write("session.viewports['Viewport: 1'].view.setValues(session.views['Back'])" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-1']" + "\n")
    file.write("f, e = p.faces, p.edges" + "\n")
    file.write("""t = p.MakeSketchTransform(sketchPlane=f[4], sketchUpEdge=e[3], sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, origin=(0.0, 0.0, 0.0))""" + "\n")
    file.write("""s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=15.75, gridSpacing=0.39, transform=t)""" + "\n")
    file.write("g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints" + "\n")
    file.write("s.setPrimaryObject(option=SUPERIMPOSE)" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-1']" + "\n")
    file.write("p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)" + "\n")
    file.write("s.CircleByCenterPerimeter(center=(0.0, 0.0), point1=(" + str(r_outer1) + ", 0.0))" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-1']" + "\n")
    file.write("f1, e1 = p.faces, p.edges" + "\n")
    file.write("""p.SolidExtrude(sketchPlane=f1[4], sketchUpEdge=e1[3], sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, sketch=s, depth=""" + str(hr) + ", flipExtrudeDirection=OFF)" + "\n")
    file.write("s.unsetPrimaryObject()" + "\n")
    file.write("del mdb.models['Model-1'].sketches['__profile__']" + "\n")


    file.write("session.viewports['Viewport: 1'].setValues(displayedObject=None)" + '\n')
    file.write("""session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues( referenceRepresentation=ON)""" + '\n')
    file.write("s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=10.0)" + '\n')
    file.write('g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints' + '\n')
    file.write('s.setPrimaryObject(option=STANDALONE)' + '\n')
    file.write('s.CircleByCenterPerimeter(center=(0.0, 0.0), point1=(' + str(radius2) + ', 0.0))'+ '\n')
    file.write("p = mdb.models['Model-1'].Part(name='DIN660-" + diameter + '-' + shank + """-2', dimensionality=THREE_D, type=DEFORMABLE_BODY)""" + '\n')
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank +"-2']" + "\n")
    file.write("p.BaseSolidExtrude(sketch=s, depth=" + str(shank_number) + ")" + '\n')
    file.write('s.unsetPrimaryObject()' + '\n')
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-2']" + "\n")
    file.write("session.viewports['Viewport: 1'].setValues(displayedObject=p)" + "\n")
    file.write("del mdb.models['Model-1'].sketches['__profile__']" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-2']" + "\n")
    file.write("f, e = p.faces, p.edges" + "\n")
    file.write("""t = p.MakeSketchTransform(sketchPlane=f[1], sketchUpEdge=e[0], sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, origin=(0.0, 0.0, """ + str(shank_number) + "))" + "\n")
    file.write("""s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=15.75, gridSpacing=0.39, transform=t)""" + "\n")
    file.write("g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints" + "\n")
    file.write("s1.setPrimaryObject(option=SUPERIMPOSE)" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-2']" + "\n")
    file.write("p.projectReferencesOntoSketch(sketch=s1, filter=COPLANAR_EDGES)" + "\n")
    file.write("s1.CircleByCenterPerimeter(center=(0.0, 0.0), point1=(" + str(r_outer2) + ", 0.0))" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-2']" + "\n")
    file.write("f1, e1 = p.faces, p.edges" + "\n")
    file.write("""p.SolidExtrude(sketchPlane=f1[1], sketchUpEdge=e1[0], sketchPlaneSide=SIDE1, sketchOrientation=RIGHT, sketch=s1, depth=""" + str(hr) + ", flipExtrudeDirection=OFF)" + "\n")
    file.write("s1.unsetPrimaryObject()" + "\n")
    file.write("del mdb.models['Model-1'].sketches['__profile__']" + "\n")
    file.write("session.viewports['Viewport: 1'].view.setValues(session.views['Back'])" + "\n")
```

```
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-2']" + "\n")
    file.write("f, e = p.faces, p.edges" + "\n")
    file.write("""t = p.MakeSketchTransform(sketchPlane=f[4], sketchUpEdge=e[3], sketchPlaneSide=SIDE1,
sketchOrientation=RIGHT, origin=(0.0, 0.0, 0.0))""" + "\n")
    file.write("""s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=15.75, gridSpac-
ing=0.39, transform=t)""" + "\n")
    file.write("g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints" + "\n")
    file.write("s.setPrimaryObject(option=SUPERIMPOSE)" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-2']" + "\n")
    file.write("p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)" + "\n")
    file.write("s.CircleByCenterPerimeter(center=(0.0, 0.0), point1=(" + str(r_outer2) + ", 0.0))" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-2']" + "\n")
    file.write("f1, e1 = p.faces, p.edges" + "\n")
    file.write("""p.SolidExtrude(sketchPlane=f1[4], sketchUpEdge=e1[3], sketchPlaneSide=SIDE1, sketchOri-
entation=RIGHT, sketch=s, depth=""" + str(hr) + ", flipExtrudeDirection=OFF)" + "\n")
    file.write("s.unsetPrimaryObject()" + "\n")
    file.write("del mdb.models['Model-1'].sketches['__profile__']" + "\n")


    file.write("session.viewports['Viewport: 1'].setValues(displayedObject=None)" + '\n')
    file.write("""session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues( referenceRepresen-
tation=ON)""" + '\n')
    file.write("s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=10.0)" + '\n')
    file.write('g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints' + '\n')
    file.write('s.setPrimaryObject(option=STANDALONE)' + '\n')
    file.write('s.CircleByCenterPerimeter(center=(0.0, 0.0), point1=(' + str(radius3) + ', 0.0))'+ '\n')
    file.write("p = mdb.models['Model-1'].Part(name='DIN660-" + diameter + '-' + shank + """-3', dimension-
ality=THREE_D, type=DEFORMABLE_BODY)""" + '\n')
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank +"-3']" + "\n")
    file.write("p.BaseSolidExtrude(sketch=s, depth=" + str(shank_number) + ")" + '\n')
    file.write('s.unsetPrimaryObject()' + '\n')
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-3']" + "\n")
    file.write("session.viewports['Viewport: 1'].setValues(displayedObject=p)" + "\n")
    file.write("del mdb.models['Model-1'].sketches['__profile__']" + "\n")


    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-3']" + "\n")
    file.write("f, e = p.faces, p.edges" + "\n")
    file.write("""t = p.MakeSketchTransform(sketchPlane=f[1], sketchUpEdge=e[0], sketchPlaneSide=SIDE1,
sketchOrientation=RIGHT, origin=(0.0, 0.0, """ + str(shank_number) + "))" + "\n")
    file.write("""s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=15.75, gridSpac-
ing=0.39, transform=t)""" + "\n")
    file.write("g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints" + "\n")
    file.write("s1.setPrimaryObject(option=SUPERIMPOSE)" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-3']" + "\n")
    file.write("p.projectReferencesOntoSketch(sketch=s1, filter=COPLANAR_EDGES)" + "\n")
    file.write("s1.CircleByCenterPerimeter(center=(0.0, 0.0), point1=(" + str(r_outer3) + ", 0.0))" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-3']" + "\n")
    file.write("f1, e1 = p.faces, p.edges" + "\n")
    file.write("""p.SolidExtrude(sketchPlane=f1[1], sketchUpEdge=e1[0], sketchPlaneSide=SIDE1, sketchOri-
entation=RIGHT, sketch=s1, depth=""" + str(hr) + ", flipExtrudeDirection=OFF)" + "\n")
    file.write("s1.unsetPrimaryObject()" + "\n")
    file.write("del mdb.models['Model-1'].sketches['__profile__']" + "\n")


    file.write("session.viewports['Viewport: 1'].view.setValues(session.views['Back'])" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-3']" + "\n")
    file.write("f, e = p.faces, p.edges" + "\n")
```

```
    file.write("""t = p.MakeSketchTransform(sketchPlane=f[4], sketchUpEdge=e[3], sketchPlaneSide=SIDE1,
sketchOrientation=RIGHT, origin=(0.0, 0.0, 0.0))""" + "\n")
    file.write("""s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=15.75, gridSpac-
ing=0.39, transform=t)""" + "\n")
    file.write("g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints" + "\n")
    file.write("s.setPrimaryObject(option=SUPERIMPOSE)" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-3']" + "\n")
    file.write("p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)" + "\n")
    file.write("s.CircleByCenterPerimeter(center=(0.0, 0.0), point1=(" + str(r_outer3) + ", 0.0))" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-3']" + "\n")
    file.write("f1, e1 = p.faces, p.edges" + "\n")
    file.write("""p.SolidExtrude(sketchPlane=f1[4], sketchUpEdge=e1[3], sketchPlaneSide=SIDE1, sketchOri-
entation=RIGHT, sketch=s, depth=""" + str(hr) + ", flipExtrudeDirection=OFF)" + "\n")
    file.write("s.unsetPrimaryObject()" + "\n")
    file.write("del mdb.models['Model-1'].sketches['__profile__']" + "\n")


    file.write("""s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=100.0)""" + "\n")
    file.write("g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints" + "\n")
    file.write("s1.setPrimaryObject(option=STANDALONE)" + "\n")
    file.write("s1.Line(point1=(" + str(plate_left_x) + ", " + str(plate_top_y) + "), point2=(" + str(plate_right_x)
+ ", " + str(plate_top_y) + "))" + "\n")
    file.write("s1.HorizontalConstraint(entity=g[2], addUndoState=False)" + "\n")
    file.write("s1.Line(point1=(" + str(plate_right_x) + ", " + str(plate_top_y) + "), point2=(" + str(plate_right_x)
+ ", " + str(plate_bottom_y) + "))" + "\n")
    file.write("s1.VerticalConstraint(entity=g[3], addUndoState=False)" + "\n")
    file.write("s1.PerpendicularConstraint(entity1=g[2], entity2=g[3], addUndoState=False)" + "\n")
    file.write("s1.Line(point1=(" + str(plate_right_x) + ", " + str(plate_bottom_y) + "), point2=(" + str(plate_left_x)
+ ", " + str(plate_bottom_y) + "))" + "\n")
    file.write("s1.HorizontalConstraint(entity=g[4], addUndoState=False)" + "\n")
    file.write("s1.PerpendicularConstraint(entity1=g[3], entity2=g[4], addUndoState=False)" + "\n")
    file.write("s1.Line(point1=(" + str(plate_left_x) + ", " + str(plate_bottom_y) + "), point2=(" + str(plate_left_x)
+ ", " + str(plate_top_y) + "))" + "\n")
    file.write("s1.VerticalConstraint(entity=g[5], addUndoState=False)" + "\n")
    file.write("s1.PerpendicularConstraint(entity1=g[4], entity2=g[5], addUndoState=False)" + "\n")
    file.write("s1.CircleByCenterPerimeter(center=(" + str(hole_1_x) + ", 0.0), point1=(" + str(hole_13_d) + ",
0.0))" + "\n")
    file.write("s1.CircleByCenterPerimeter(center=(" + str(hole_2_x) + ", 0.0), point1=(" + str(hole_2_d) + ",
0.0))" + "\n")
    file.write("s1.CircleByCenterPerimeter(center=(" + str(hole_3_x) + ", 0.0), point1=(" + str(hole_31_d) + ",
0.0))" + "\n")
    file.write("""p = mdb.models['Model-1'].Part(name='Plate', dimensionality=THREE_D, type=DEFORMABLE_BODY)"""
+ "\n")
    file.write("p = mdb.models['Model-1'].parts['Plate']" + "\n")
    file.write("p.BaseSolidExtrude(sketch=s1, depth=" + str(t_1) + ")" + "\n")
    file.write("s1.unsetPrimaryObject()" + "\n") file.write("p = mdb.models['Model-1'].parts['Plate']" + "\n")
    file.write("session.viewports['Viewport: 1'].setValues(displayedObject=p)" + "\n")
    file.write("del mdb.models['Model-1'].sketches['__profile__']" + "\n")


    file.write("""s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=100.0)""" + "\n")
    file.write("g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints" + "\n")
    file.write("s1.setPrimaryObject(option=STANDALONE)" + "\n")
    file.write("s1.Line(point1=(" + str(plate_left_x) + ", " + str(plate_top_y) + "), point2=(" + str(plate_right_x)
+ ", " + str(plate_top_y) + "))" + "\n")
    file.write("s1.HorizontalConstraint(entity=g[2], addUndoState=False)" + "\n")
    file.write("s1.Line(point1=(" + str(plate_right_x) + ", " + str(plate_top_y) + "), point2=(" + str(plate_right_x)
+ ", " + str(plate_bottom_y) + "))" + "\n")
```

```
    file.write("s1.VerticalConstraint(entity=g[3], addUndoState=False)" + "\n")
    file.write("s1.PerpendicularConstraint(entity1=g[2], entity2=g[3], addUndoState=False)" + "\n")
    file.write("s1.Line(point1=(" + str(plate_right_x) + ", " + str(plate_bottom_y) + "), point2=(" + str(plate_left_x)
+ ", " + str(plate_bottom_y) + "))" + "\n")
    file.write("s1.HorizontalConstraint(entity=g[4], addUndoState=False)" + "\n")
    file.write("s1.PerpendicularConstraint(entity1=g[3], entity2=g[4], addUndoState=False)" + "\n")
    file.write("s1.Line(point1=(" + str(plate_left_x) + ", " + str(plate_bottom_y) + "), point2=(" + str(plate_left_x)
+ ", " + str(plate_top_y) + "))" + "\n")
    file.write("s1.VerticalConstraint(entity=g[5], addUndoState=False)" + "\n")
    file.write("s1.PerpendicularConstraint(entity1=g[4], entity2=g[5], addUndoState=False)" + "\n")
    file.write("s1.CircleByCenterPerimeter(center=(" + str(hole_1_x) + ", 0.0), point1=(" + str(hole_1_d) + ",
0.0))" + "\n")
    file.write("s1.CircleByCenterPerimeter(center=(" + str(hole_2_x) + ", 0.0), point1=(" + str(hole_2_d) + ",
0.0))" + "\n")
    file.write("s1.CircleByCenterPerimeter(center=(" + str(hole_3_x) + ", 0.0), point1=(" + str(hole_3_d) + ",
0.0))" + "\n")
    file.write("""p = mdb.models['Model-1'].Part(name='Plate_2', dimensionality=THREE_D, type=DEFORMABLE_BODY)"""
+ "\n")
    file.write("p = mdb.models['Model-1'].parts['Plate_2']" + "\n")
    file.write("p.BaseSolidExtrude(sketch=s1, depth=" + str(t_2) + ")" + "\n")
    file.write("s1.unsetPrimaryObject()" + "\n") file.write("p = mdb.models['Model-1'].parts['Plate_2']" + "\n")
    file.write("session.viewports['Viewport: 1'].setValues(displayedObject=p)" + "\n") file.write("del mdb.models['Model-
1'].sketches['__profile__']" + "\n")


    file.write("""session.viewports['Viewport: 1'].partDisplay.setValues(sectionAssignments=ON, engineeringFea-
tures=ON)""" + "\n")
    file.write("""session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues( referenceRepresen-
tation=OFF)""" + "\n")
    file.write("from material import createMaterialFromDataString" + "\n")
    file.write("'createMaterialFromDataString('Model-1', 'Mild-Steel-no-plastic', '6-13', """'name': 'Mild-Steel-
no-plastic', 'materialIdentifier': '', 'description': '', 'elastic': 'temperatureDependency': OFF, 'moduli': LONG_TERM,
'noCompression': OFF, 'noTension': OFF, 'dependencies': 0, 'table': ((198500.0, 0.29),), 'type': ISOTROPIC,
'density': 'temperatureDependency': OFF, 'table': ((8.0e-09,),), 'dependencies': 0, 'fieldName': '', 'distribution-
Type': UNIFORM""")"' + "\n")
    #: Material 'Mild-Steel-no-plastic' has been copied to the current model.


    file.write("""session.viewports['Viewport: 1'].partDisplay.setValues(sectionAssignments=ON, engineeringFea-
tures=ON)""" + "\n")
    file.write("""session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues( referenceRepresen-
tation=OFF)""" + "\n")
    file.write("from material import createMaterialFromDataString" + "\n")
    file.write("'createMaterialFromDataString('Model-1', 'Aluminum', '6-13', """'name': 'Aluminum', 'materi-
alIdentifier': '', 'description': '', 'elastic': 'temperatureDependency': OFF, 'moduli': LONG_TERM, 'noCom-
pression': OFF, 'noTension': OFF, 'dependencies': 0, 'table': ((73100.0, 0.33),), 'type': ISOTROPIC, 'density':
'temperatureDependency': OFF, 'table': ((2.78e-09,),), 'dependencies': 0, 'fieldName': '', 'distributionType':
UNIFORM""")"' + "\n")
    #: Material 'Mild-Steel-no-plastic' has been copied to the current model.


    file.write("""mdb.models['Model-1'].HomogeneousSolidSection(name='Section-1', material='Mild-Steel-
no-plastic', thickness=None)""" + "\n")
    #mdb.save()
    file.write("""mdb.models['Model-1'].HomogeneousSolidSection(name='Section-2', material='Aluminum',
thickness=None)""" + "\n")
    file.write("p = mdb.models['Model-1'].parts['Plate']" + "\n")
    file.write("c = p.cells" + "\n")
```

```
    file.write("cells = c.getSequenceFromMask(mask=('[#1 ]', ), )" + "\n")
    file.write("region = p.Set(cells=cells, name='Set-1')" + "\n")
    file.write("p = mdb.models['Model-1'].parts['Plate']" + "\n")
    file.write("""p.SectionAssignment(region=region, sectionName='Section-2', offset=0.0, offsetType=MIDDLE_SURFACE,
offsetField='', thicknessAssignment=FROM_SECTION)""" + "\n")
    file.write("p = mdb.models['Model-1'].parts['Plate_2']" + "\n")
    file.write("c = p.cells" + "\n")
    file.write("cells = c.getSequenceFromMask(mask=('[#1 ]', ), )" + "\n")
    file.write("region = p.Set(cells=cells, name='Set-1')" + "\n")
    file.write("p = mdb.models['Model-1'].parts['Plate_2']" + "\n")
    file.write("""p.SectionAssignment(region=region, sectionName='Section-2', offset=0.0, offsetType=MIDDLE_SURFACE,
offsetField='', thicknessAssignment=FROM_SECTION)""" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-3']" + "\n")
    file.write("session.viewports['Viewport: 1'].setValues(displayedObject=p)" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-3']" + "\n")
    file.write("c = p.cells" + "\n")
    file.write("cells = c.getSequenceFromMask(mask=('[#1 ]', ), )" + "\n")
    file.write("region = p.Set(cells=cells, name='Set-1')" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-3']" + "\n")
    file.write("""p.SectionAssignment(region=region, sectionName='Section-1', offset=0.0, offsetType=MIDDLE_SURFACE,
offsetField='', thicknessAssignment=FROM_SECTION)""" + "\n")
    file.write("session.viewports['Viewport: 1'].view.setValues(session.views['Iso'])" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-1']" + "\n")
    file.write("session.viewports['Viewport: 1'].setValues(displayedObject=p)" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-1']" + "\n")
    file.write("c = p.cells" + "\n")
    file.write("cells = c.getSequenceFromMask(mask=('[#1 ]', ), )" + "\n")
    file.write("region = p.Set(cells=cells, name='Set-1')" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-1']" + "\n")
    file.write("""p.SectionAssignment(region=region, sectionName='Section-1', offset=0.0, offsetType=MIDDLE_SURFACE,
offsetField='', thicknessAssignment=FROM_SECTION)""" + "\n")
    file.write("session.viewports['Viewport: 1'].view.setValues(session.views['Iso'])" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-2']" + "\n")
    file.write("session.viewports['Viewport: 1'].setValues(displayedObject=p)" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-2']" + "\n")
    file.write("c = p.cells" + "\n")
    file.write("cells = c.getSequenceFromMask(mask=('[#1 ]', ), )" + "\n")
    file.write("region = p.Set(cells=cells, name='Set-1')" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-2']" + "\n")
    file.write("""p.SectionAssignment(region=region, sectionName='Section-1', offset=0.0, offsetType=MIDDLE_SURFACE,
offsetField='', thicknessAssignment=FROM_SECTION)""" + "\n")
    file.write("session.viewports['Viewport: 1'].view.setValues(session.views['Iso'])" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("session.viewports['Viewport: 1'].setValues(displayedObject=a)" + "\n")
    file.write("""session.viewports['Viewport: 1'].assemblyDisplay.setValues( optimizationTasks=OFF, geomet-
ricRestrictions=OFF, stopConditions=OFF)""" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("a.DatumCsysByDefault(CARTESIAN)" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-1']" + "\n")
    file.write("a.Instance(name='DIN660-" + diameter + "-" + shank + "-1', part=p, dependent=OFF)" + "\n")
    file.write("session.viewports['Viewport: 1'].view.fitView()" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-2']" + "\n")
    file.write("a.Instance(name='DIN660-" + diameter + "-" + shank + "-2', part=p, dependent=OFF)" + "\n")
    file.write("p1 = a.instances['DIN660-" + diameter + "-" + shank + "-2']" + "\n")
    file.write("p1.translate(vector=(" + str(d2) + ", 0.0, 0.0))" + "\n")
```

```
    file.write("session.viewports['Viewport: 1'].view.fitView()" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("p = mdb.models['Model-1'].parts['DIN660-" + diameter + "-" + shank + "-3']" + "\n")
    file.write("a.Instance(name='DIN660-" + diameter + "-" + shank + "-3', part=p, dependent=OFF)" + "\n")
    file.write("p1 = a.instances['DIN660-" + diameter + "-" + shank + "-3']" + "\n")
    file.write("p1.translate(vector=(" + str(d3) + ", 0.0, 0.0))" + "\n")
    file.write("session.viewports['Viewport: 1'].view.fitView()" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("p = mdb.models['Model-1'].parts['Plate']" + "\n")
    file.write("a.Instance(name='Plate-1', part=p, dependent=OFF)" + "\n")
    file.write("p1 = a.instances['Plate-1']" + "\n")
    file.write("p1.translate(vector=(" + str(d4) + ", 0.0, 0.0))" + "\n")
    file.write("session.viewports['Viewport: 1'].view.fitView()" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("p = mdb.models['Model-1'].parts['Plate_2']" + "\n")
    file.write("a.Instance(name='Plate-2', part=p, dependent=OFF)" + "\n")
    file.write("p1 = a.instances['Plate-2']" + "\n")
    file.write("p1.translate(vector=(" + str(d5) + ", 0.0, 0.0))" + "\n")
    file.write("session.viewports['Viewport: 1'].view.fitView()" + "\n")

    file.write("""session.viewports['Viewport: 1'].view.setValues(nearPlane=326.24, farPlane=465.167, width=58.1236,
height=28.1657, viewOffsetX=-30.3011, viewOffsetY=20.5228)""" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("a.translate(instanceList=('DIN660-" + diameter + "-" + shank + "-2', ), vector=(" + str(d4_d2) +
", 0.0, -" + str(t_2) + "))" + "\n")
    #: The instance DIN660-4,8-4-2 was translated by 49.76, 0., -2. with respect to the assembly coordinate
system
    file.write("""session.viewports['Viewport: 1'].view.setValues(nearPlane=318.667, farPlane=473.895, width=119.294,
height=57.8076, viewOffsetX=-8.81276, viewOffsetY=13.9707)""" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("a.translate(instanceList=('DIN660-" + diameter + "-" + shank + "-3', ), vector=(" + str(d4_d3plushole3x)
+ ", 0.0, -" + str(t_2) + "))" + "\n")
    #: The instance DIN660-4,8-4-3 was translated by 59.28, 0., -2. with respect to the assembly coordinate
system

    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("a.translate(instanceList=('DIN660-" + diameter + "-" + shank + "-1', ), vector=(" + str(d4_hole3x)
+ ", 0.0, -" + str(t_2) + "))" + "\n")
    #: The instance DIN660-4,8-4-1 was translated by 40.24, 0., -2. with respect to the assembly coordinate
system

    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("a.translate(instanceList=('Plate-2', ), vector=(-" + str(d5_d4) + ", 0.0, -" + str(t_2) + "))" + "\n")
    #: The instance Plate-2 was translated by -75.68, 0., -2. with respect to the assembly coordinate system
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("a.rotate(instanceList=('Plate-1', ), axisPoint=(" + str(d4) + """, 0.0, 5.0), axisDirection=(0.0, 0.0,
-10.0), angle=180.0)""" + "\n")
    #: The instance Plate-1 was rotated by 180. degrees about the axis defined by the point 59.44, 0., 5. and
the vector 0., 0., -10.
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("c1 = a.instances['DIN660-" + diameter + "-" + shank + "-1'].cells" + "\n")
    file.write("cells1 = c1.getSequenceFromMask(mask=('[#1 ]', ), )" + "\n")
    file.write("c2 = a.instances['DIN660-" + diameter + "-" + shank + "-2'].cells" + "\n")
    file.write("cells2 = c2.getSequenceFromMask(mask=('[#1 ]', ), )" + "\n")
    file.write("c3 = a.instances['DIN660-" + diameter + "-" + shank + "-3'].cells" + "\n")
```

```
    file.write("cells3 = c3.getSequenceFromMask(mask=('[#1 ]', ), )" + "\n")
    file.write("pickedCells = cells1+cells2+cells3" + "\n")
    file.write("f1 = a.instances['Plate-1'].faces" + "\n")
    file.write("a.PartitionCellByExtendFace(extendFace=f1[7], cells=pickedCells)" + "\n")

    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("c1 = a.instances['DIN660-" + diameter + "-" + shank + "-1'].cells" + "\n")
    file.write("cells1 = c1.getSequenceFromMask(mask=('[#2 ]', ), )" + "\n")
    file.write("c2 = a.instances['DIN660-" + diameter + "-" + shank + "-2'].cells" + "\n")
    file.write("cells2 = c2.getSequenceFromMask(mask=('[#2 ]', ), )" + "\n")
    file.write("c3 = a.instances['DIN660-" + diameter + "-" + shank + "-3'].cells" + "\n")
    file.write("cells3 = c3.getSequenceFromMask(mask=('[#2 ]', ), )" + "\n")
    file.write("pickedCells = cells1+cells2+cells3" + "\n")
    file.write("f21 = a.instances['Plate-2'].faces" + "\n")
    file.write("a.PartitionCellByExtendFace(extendFace=f21[8], cells=pickedCells)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-2']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("c1 = a.instances['DIN660-" + diameter + "-" + shank + "-1'].cells" + "\n")
    file.write("cells1 = c1.getSequenceFromMask(mask=('[#1 ]', ), )" + "\n")
    file.write("c2 = a.instances['DIN660-" + diameter + "-" + shank + "-2'].cells" + "\n")
    file.write("cells2 = c2.getSequenceFromMask(mask=('[#1 ]', ), )" + "\n")
    file.write("c3 = a.instances['DIN660-" + diameter + "-" + shank + "-3'].cells" + "\n")
    file.write("cells3 = c3.getSequenceFromMask(mask=('[#1 ]', ), )" + "\n")
    file.write("pickedCells = cells1+cells2+cells3" + "\n")
    file.write("f1 = a.instances['Plate-1'].faces" + "\n")
    file.write("a.PartitionCellByExtendFace(extendFace=f1[8], cells=pickedCells)" + "\n")

    file.write("leaf = dgm.Leaf(leafType=DEFAULT_MODEL)" + "\n")
    file.write("""session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.replace( leaf=leaf)""" + "\n")

    file.write("a1 = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("a1.translate(instanceList=('DIN660-" + diameter + "-" + shank + "-1', 'DIN660-" + diameter +
"-" + shank + """-2', 'DIN660-""" + diameter + "-" + shank + "-3', 'Plate-1', 'Plate-2'), vector=(-" + str(d4) + ",
0.0, 0.0))" + "\n")
    #: The instances were translated by allowable translation with respect to the assembly coordinate system

    file.write("""session.viewports['Viewport: 1'].assemblyDisplay.setValues( adaptiveMeshConstraints=ON)"""
+ "\n")
    file.write("""mdb.models['Model-1'].StaticStep(name='Pretension', previous='Initial', maxNumInc=100000,
initialInc=0.01, maxInc=0.1, nlgeom=ON)""" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.setValues(step='Pretension')" + "\n")
    file.write("""mdb.models['Model-1'].StaticStep(name='Traction', previous='Pretension', maxNumInc=100000,
initialInc=0.01, maxInc=0.1)""" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.setValues(step='Traction')" + "\n")

    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("a.DatumPointByCoordinate(coords=(" + str(plate_right_x) +", 0.0, -" + str(z_point_force) + "))"
+ "\n")

    file.write("""session.viewports['Viewport: 1'].assemblyDisplay.setValues(interactions=ON, constraints=ON,
connectors=ON, engineeringFeatures=ON, adaptiveMeshConstraints=OFF)""" + "\n")
    file.write("mdb.models['Model-1'].ContactProperty('Contact')" + "\n")
```

```
    file.write("""mdb.models['Model-1'].interactionProperties['Contact'].TangentialBehavior( formulation=PENALTY,
directionality=ISOTROPIC, slipRateDependency=OFF, pressureDependency=OFF, temperatureDependency=OFF,
dependencies=0, table=(( 0.3, ), ), shearStressLimit=None, maximumElasticSlip=FRACTION, fraction=0.005,
elasticSlipStiffness=None)""" + "\n")
    file.write("""mdb.models['Model-1'].interactionProperties['Contact'].NormalBehavior( pressureOverclo-
sure=HARD, allowSeparation=ON, constraintEnforcementMethod=DEFAULT)""" + "\n")
    #: The interaction property "Contact" has been created.
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-2']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")

    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-1']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-1']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.setValues(step='Initial')" + "\n")

    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-1']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-1']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['DIN660-" + diameter + "-" + shank + "-1'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#80 ]', ), )" + "\n")
    file.write("region1=a.Surface(side1Faces=side1Faces1, name='m_Surf-1')" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['Plate-1'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#80 ]', ), )" + "\n")
    file.write("region2=a.Surface(side1Faces=side1Faces1, name='s_Surf-1')" + "\n")
    file.write("""mdb.models['Model-1'].SurfaceToSurfaceContactStd(name='DinInnerHead-Top', createStep-
Name='Initial', master=region1, slave=region2, sliding=SMALL, enforcement=NODE_TO_SURFACE, thick-
ness=ON, interactionProperty='Contact', surfaceSmoothing=NONE, adjustMethod=TOLERANCE, smooth=0.2,
initialClearance=OMIT, datumAxis=None, clearanceRegion=None, tied=OFF, adjustTolerance=0.0)""" + "\n")
    #: The interaction "DinInnerHead-Top" has been created.

    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-1']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['DIN660-" + diameter + "-" + shank +
"-1']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-1']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
    file.write("""session.viewports['Viewport: 1'].view.setValues(nearPlane=151.038, farPlane=211.346, width=54.068,
height=26.2004, viewOffsetX=8.30588, viewOffsetY=-0.301202)""" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['DIN660-" + diameter + "-" + shank + "-1'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#2 ]', ), )" + "\n")
    file.write("region1=a.Surface(side1Faces=side1Faces1, name='m_Surf-3')" + "\n")
```

```
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['Plate-1'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#40 ]', ), )" + "\n")
    file.write("region2=a.Surface(side1Faces=side1Faces1, name='s_Surf-3')" + "\n")
    file.write("""mdb.models['Model-1'].SurfaceToSurfaceContactStd(name='DinInnerCollar-Top', createStepName='Initial', master=region1, slave=region2, sliding=SMALL, enforcement=NODE_TO_SURFACE, thickness=ON, interactionProperty='Contact', surfaceSmoothing=NONE, adjustMethod=TOLERANCE, smooth=0.2, initialClearance=OMIT, datumAxis=None, clearanceRegion=None, tied=OFF, adjustTolerance=0.0)""" + "\n")
    #: The interaction "DinInnerCollar-Top" has been created.

    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-1']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-1']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['DIN660-" + diameter + "-" + shank + "-2'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#80 ]', ), )" + "\n")
    file.write("region1=a.Surface(side1Faces=side1Faces1, name='m_Surf-5')" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['Plate-1'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#80 ]', ), )" + "\n")
    file.write("region2=a.Surface(side1Faces=side1Faces1, name='s_Surf-5')" + "\n")
    file.write("""mdb.models['Model-1'].SurfaceToSurfaceContactStd(name='DinMiddleHead-Top', createStepName='Initial', master=region1, slave=region2, sliding=SMALL, enforcement=NODE_TO_SURFACE, thickness=ON, interactionProperty='Contact', surfaceSmoothing=NONE, adjustMethod=TOLERANCE, smooth=0.2, initialClearance=OMIT, datumAxis=None, clearanceRegion=None, tied=OFF, adjustTolerance=0.0)""" + "\n")
    #: The interaction "DinMiddleHead-Top" has been created.

    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-1']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['DIN660-" + diameter + "-" + shank + "-2']" + "\n") file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-1']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['DIN660-" + diameter + "-" + shank + "-2'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#2 ]', ), )" + "\n")
    file.write("region1=a.Surface(side1Faces=side1Faces1, name='m_Surf-7')" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['Plate-1'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#10 ]', ), )" + "\n")
    file.write("region2=a.Surface(side1Faces=side1Faces1, name='s_Surf-7')" + "\n")
    file.write("""mdb.models['Model-1'].SurfaceToSurfaceContactStd(name='DinMiddleCollar-Top', createStepName='Initial', master=region1, slave=region2, sliding=SMALL, enforcement=NODE_TO_SURFACE, thickness=ON, interactionProperty='Contact', surfaceSmoothing=NONE, adjustMethod=TOLERANCE, smooth=0.2, initialClearance=OMIT, datumAxis=None, clearanceRegion=None, tied=OFF, adjustTolerance=0.0)""" + "\n")
    #: The interaction "DinMiddleCollar-Top" has been created.
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-1']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
```

file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-1']" + "\n")
file.write("leaf = dgm.LeafFromInstance(instances=(i1,))" + "\n")
file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
file.write("s1 = a.instances['DIN660-" + diameter + "-" + shank + "-3'].faces" + "\n")
file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#80]',),)" + "\n")
file.write("region1=a.Surface(side1Faces=side1Faces1, name='m_Surf-9')" + "\n")
file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
file.write("s1 = a.instances['Plate-1'].faces" + "\n")
file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#80]',),)" + "\n")
file.write("region2=a.Surface(side1Faces=side1Faces1, name='s_Surf-9')" + "\n")
file.write("""mdb.models['Model-1'].SurfaceToSurfaceContactStd(name='DinOutHead-Top', createStepName='Initial', master=region1, slave=region2, sliding=SMALL, enforcement=NODE_TO_SURFACE, thickness=ON, interactionProperty='Contact', surfaceSmoothing=NONE, adjustMethod=TOLERANCE, smooth=0.2, initialClearance=OMIT, datumAxis=None, clearanceRegion=None, tied=OFF, adjustTolerance=0.0)""" + "\n")
#: The interaction "DinOutHead-Top" has been created.

file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-1']" + "\n")
file.write("leaf = dgm.LeafFromInstance(instances=(i1,))" + "\n")
file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['DIN660-" + diameter + "-" + shank + "-3']" + "\n") file.write("leaf = dgm.LeafFromInstance(instances=(i1,))" + "\n")
file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-1']" + "\n")
file.write("leaf = dgm.LeafFromInstance(instances=(i1,))" + "\n")
file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
file.write("s1 = a.instances['DIN660-" + diameter + "-" + shank + "-3'].faces" + "\n")
file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#2]',),)" + "\n")
file.write("region1=a.Surface(side1Faces=side1Faces1, name='m_Surf-11')" + "\n")
file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
file.write("s1 = a.instances['Plate-1'].faces" + "\n")
file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#20]',),)" + "\n")
file.write("region2=a.Surface(side1Faces=side1Faces1, name='s_Surf-11')" + "\n")
file.write("""mdb.models['Model-1'].SurfaceToSurfaceContactStd(name='DinOutCollar-Top', createStepName='Initial', master=region1, slave=region2, sliding=SMALL, enforcement=NODE_TO_SURFACE, thickness=ON, interactionProperty='Contact', surfaceSmoothing=NONE, adjustMethod=TOLERANCE, smooth=0.2, initialClearance=OMIT, datumAxis=None, clearanceRegion=None, tied=OFF, adjustTolerance=0.0)""" + "\n")
#: The interaction "DinOutCollar-Top" has been created.

file.write("leaf = dgm.Leaf(leafType=DEFAULT_MODEL)" + "\n")
file.write("""session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.replace(leaf=leaf)""" + "\n")

file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-1']" + "\n")
file.write("leaf = dgm.LeafFromInstance(instances=(i1,))" + "\n")
file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-2']" + "\n")
file.write("leaf = dgm.LeafFromInstance(instances=(i1,))" + "\n")
file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-2']" + "\n")
file.write("leaf = dgm.LeafFromInstance(instances=(i1,))" + "\n")
file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-2']" + "\n")
file.write("leaf = dgm.LeafFromInstance(instances=(i1,))" + "\n")
file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")

```
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['DIN660-" + diameter + "-" + shank +
"-3']" + "\n") file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['DIN660-" + diameter + "-" + shank +
"-3']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['DIN660-" + diameter + "-" + shank +
"-1']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-2']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)""" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['DIN660-" + diameter + "-" + shank +
"-1']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['DIN660-" + diameter + "-" + shank +
"-1']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['Plate-2'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#100 ]', ), )" + "\n")
    file.write("region1=a.Surface(side1Faces=side1Faces1, name='m_Surf-13')" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['DIN660-" + diameter + "-" + shank + "-1'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#10 ]', ), )" + "\n")
    file.write("region2=a.Surface(side1Faces=side1Faces1, name='s_Surf-13')" + "\n")
    file.write("""mdb.models['Model-1'].SurfaceToSurfaceContactStd(name='DinInnerHead-Bottom', createStep-
Name='Initial', master=region1, slave=region2, sliding=SMALL, enforcement=NODE_TO_SURFACE, thick-
ness=ON, interactionProperty='Contact', surfaceSmoothing=NONE, adjustMethod=TOLERANCE, smooth=0.2,
initialClearance=OMIT, datumAxis=None, clearanceRegion=None, tied=OFF, adjustTolerance=0.0)""" + "\n")
    #: The interaction "DinInnerHead-Bottom" has been created. file.write("i1 = mdb.models['Model-1'].rootAssembly.allIns
2']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['DIN660-" + diameter + "-" + shank +
"-1']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")


    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-2']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['DIN660-" + diameter + "-" + shank +
"-1']" + "\n") file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['Plate-2'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#20 ]', ), )" + "\n")
    file.write("region1=a.Surface(side1Faces=side1Faces1, name='m_Surf-15')" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['DIN660-" + diameter + "-" + shank + "-1'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#400 ]', ), )" + "\n")
```

```
    file.write("region2=a.Surface(side1Faces=side1Faces1, name='s_Surf-15')" + "\n")
    file.write("""mdb.models['Model-1'].SurfaceToSurfaceContactStd(name='DinInnerCollar-Bottom', creat-
eStepName='Initial', master=region1, slave=region2, sliding=SMALL, enforcement=NODE_TO_SURFACE, thick-
ness=ON, interactionProperty='Contact', surfaceSmoothing=NONE, adjustMethod=TOLERANCE, smooth=0.2,
initialClearance=OMIT, datumAxis=None, clearanceRegion=None, tied=OFF, adjustTolerance=0.0)""" + "\n")
    #: The interaction "DinInnerCollar-Bottom" has been created.

    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-2']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-2']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['Plate-2'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#100 ]', ), )" + "\n")
    file.write("region1=a.Surface(side1Faces=side1Faces1, name='m_Surf-17')" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['DIN660-" + diameter + "-" + shank + "-2'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#10 ]', ), )" + "\n")
    file.write("region2=a.Surface(side1Faces=side1Faces1, name='s_Surf-17')" + "\n")
    file.write("""mdb.models['Model-1'].SurfaceToSurfaceContactStd(name='DinMiddleHead-Bottom', creat-
eStepName='Initial', master=region1, slave=region2, sliding=SMALL, enforcement=NODE_TO_SURFACE, thick-
ness=ON, interactionProperty='Contact', surfaceSmoothing=NONE, adjustMethod=TOLERANCE, smooth=0.2,
initialClearance=OMIT, datumAxis=None, clearanceRegion=None, tied=OFF, adjustTolerance=0.0)""" + "\n")
    #: The interaction "DinMiddleHead-Bottom" has been created.
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['DIN660-" + diameter + "-" + shank +
"-2']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-2']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-2']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-2']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['DIN660-" + diameter + "-" + shank +
"-2']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['Plate-2'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#10 ]', ), )" + "\n")
    file.write("region1=a.Surface(side1Faces=side1Faces1, name='m_Surf-19')" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['DIN660-" + diameter + "-" + shank + "-2'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#400 ]', ), )" + "\n")
    file.write("region2=a.Surface(side1Faces=side1Faces1, name='s_Surf-19')" + "\n")
    file.write("""mdb.models['Model-1'].SurfaceToSurfaceContactStd(name='DinMiddleCollar-Bottom', cre-
ateStepName='Initial', master=region1, slave=region2, sliding=SMALL, enforcement=NODE_TO_SURFACE,
thickness=ON, interactionProperty='Contact', surfaceSmoothing=NONE, adjustMethod=TOLERANCE, smooth=0.2,
initialClearance=OMIT, datumAxis=None, clearanceRegion=None, tied=OFF, adjustTolerance=0.0)""" + "\n")
    #: The interaction "DinMiddleCollar-Bottom" has been created.
```

```
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-2']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['DIN660-" + diameter + "-" + shank +
"-3']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['DIN660-" + diameter + "-" + shank +
"-3']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-2']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['Plate-2'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#100 ]', ), )" + "\n")
    file.write("region1=a.Surface(side1Faces=side1Faces1, name='m_Surf-21')" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['DIN660-" + diameter + "-" + shank + "-3'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#10 ]', ), )" + "\n")
    file.write("region2=a.Surface(side1Faces=side1Faces1, name='s_Surf-21')" + "\n")
    file.write("""mdb.models['Model-1'].SurfaceToSurfaceContactStd(name='DinOutHead-Bottom', createStep-
Name='Initial', master=region1, slave=region2, sliding=SMALL, enforcement=NODE_TO_SURFACE, thick-
ness=ON, interactionProperty='Contact', surfaceSmoothing=NONE, adjustMethod=TOLERANCE, smooth=0.2,
initialClearance=OMIT, datumAxis=None, clearanceRegion=None, tied=OFF, adjustTolerance=0.0)""" + "\n")
    #: The interaction "DinOutHead-Bottom" has been created.

    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-2']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['DIN660-" + diameter + "-" + shank +
"-3']" + "\n") file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-2']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['DIN660-" + diameter + "-" + shank +
"-3']" + "\n") file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['Plate-2'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#40 ]', ), )" + "\n")
    file.write("region1=a.Surface(side1Faces=side1Faces1, name='m_Surf-23')" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['DIN660-" + diameter + "-" + shank + "-3'].faces" + "\n")
    file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#400 ]', ), )" + "\n")
    file.write("region2=a.Surface(side1Faces=side1Faces1, name='s_Surf-23')" + "\n")
    file.write("""mdb.models['Model-1'].SurfaceToSurfaceContactStd(name='DinOutCollar-Bottom', createStep-
Name='Initial', master=region1, slave=region2, sliding=SMALL, enforcement=NODE_TO_SURFACE, thick-
ness=ON, interactionProperty='Contact', surfaceSmoothing=NONE, adjustMethod=TOLERANCE, smooth=0.2,
initialClearance=OMIT, datumAxis=None, clearanceRegion=None, tied=OFF, adjustTolerance=0.0)""" + "\n")
    #: The interaction "DinOutCollar-Bottom" has been created.

    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-2']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
```

```
file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-1']" + "\n")
file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.add(leaf=leaf)" + "\n")
file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
file.write("s1 = a.instances['Plate-2'].faces" + "\n")
file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#80 ]', ), )" + "\n")
file.write("region1=a.Surface(side1Faces=side1Faces1, name='m_Surf-25')" + "\n")
file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
file.write("s1 = a.instances['Plate-1'].faces" + "\n")
file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#100 ]', ), )" + "\n")
file.write("region2=a.Surface(side1Faces=side1Faces1, name='s_Surf-25')" + "\n")
file.write("""mdb.models['Model-1'].SurfaceToSurfaceContactStd(name='Plate1-Plate2', createStepName='Initial',
master=region1, slave=region2, sliding=SMALL, enforcement=NODE_TO_SURFACE, thickness=ON, interac-
tionProperty='Contact', surfaceSmoothing=NONE, adjustMethod=TOLERANCE, smooth=0.2, initialClearance=OMIT,
datumAxis=None, clearanceRegion=None, tied=OFF, adjustTolerance=0.0)""" + "\n")
#: The interaction "Plate1-Plate2" has been created.

file.write("session.viewports['Viewport: 1'].view.setValues(session.views['Iso'])" + "\n")
file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
file.write("d11 = a.datums" + "\n") file.write("a.ReferencePoint(point=d11[15])" + "\n")

file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
file.write("r1 = a.referencePoints" + "\n")
file.write("refPoints1=(r1[42], )" + "\n")
file.write("region1=a.Set(referencePoints=refPoints1, name='m_Set-1')" + "\n")
file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
file.write("s1 = a.instances['Plate-2'].faces" + "\n")
file.write("side1Faces1 = s1.getSequenceFromMask(mask=('[#2 ]', ), )" + "\n")
file.write("region2=a.Surface(side1Faces=side1Faces1, name='s_Surf-27')" + "\n")
file.write("""mdb.models['Model-1'].Coupling(name='Kinematic coupling', controlPoint=region1, surface=region2,
influenceRadius=WHOLE_SURFACE, couplingType=KINEMATIC, localCsys=None, u1=ON, u2=ON, u3=ON,
ur1=ON, ur2=ON, ur3=ON)""" + "\n")

file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
file.write("e1 = a.instances['DIN660-" + diameter + "-" + shank + "-3'].edges" + "\n")
file.write("a.DatumAxisByTwoPoint(point1=a.instances['DIN660-" + diameter + "-" + shank + """-3'].InterestingPoint(
edge=e1[6], rule=CENTER), point2=a.instances['DIN660-""" + diameter + "-" + shank + """-3'].InterestingPoint(edge=e1[4],
rule=CENTER))""" + "\n")
file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
file.write("e21 = a.instances['DIN660-" + diameter + "-" + shank + "-2'].edges" + "\n")
file.write("a.DatumAxisByTwoPoint(point1=a.instances['DIN660-" + diameter + "-" + shank + """-2'].InterestingPoint(
edge=e21[4], rule=CENTER), point2=a.instances['DIN660-""" + diameter + "-" + shank + """-2'].InterestingPoint(edge=e21[6],
rule=CENTER))""" + "\n")
file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
file.write("e1 = a.instances['DIN660-" + diameter + "-" + shank + "-1'].edges" + "\n")
file.write("a.DatumAxisByTwoPoint(point1=a.instances['DIN660-" + diameter + "-" + shank + """-1'].InterestingPoint(
edge=e1[4], rule=CENTER), point2=a.instances['DIN660-""" + diameter + "-" + shank + """-1'].InterestingPoint(edge=e1[6],
rule=CENTER))""" + "\n")

file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
file.write("f1 = a.instances['Plate-1'].faces" + "\n")
file.write("faces1 = f1.getSequenceFromMask(mask=('[#2 ]', ), )" + "\n")
file.write("region = a.Set(faces=faces1, name='Set-2')" + "\n")
```

```
    file.write("""mdb.models['Model-1'].EncastreBC(name='Clamped', createStepName='Initial', region=region,
localCsys=None)""" + "\n")

    file.write("session.viewports['Viewport: 1'].assemblyDisplay.setValues(step='Pretension')" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-1']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("i1 = mdb.models['Model-1'].rootAssembly.allInstances['Plate-2']" + "\n")
    file.write("leaf = dgm.LeafFromInstance(instances=(i1, ))" + "\n")
    file.write("session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.remove(leaf=leaf)" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['DIN660-" + diameter + "-" + shank + "-1'].faces" + "\n")
    file.write("side2Faces1 = s1.getSequenceFromMask(mask=('[#1 ]', ), )" + "\n")
    file.write("region = a.Surface(side2Faces=side2Faces1, name='Surf-28')" + "\n")
    file.write("datumAxis = mdb.models['Model-1'].rootAssembly.datums[47]" + "\n")
    file.write("""mdb.models['Model-1'].BoltLoad(name='Inner riveting', createStepName='Pretension', region=region,
magnitude=""" + str(f_in) + """, boltMethod=APPLY_FORCE, datumAxis=datumAxis)""" + "\n")

    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['DIN660-" + diameter + "-" + shank + "-2'].faces" + "\n")
    file.write("side2Faces1 = s1.getSequenceFromMask(mask=('[#1 ]', ), )" + "\n")
    file.write("region = a.Surface(side2Faces=side2Faces1, name='Surf-29')" + "\n")
    file.write("datumAxis = mdb.models['Model-1'].rootAssembly.datums[46]" + "\n")
    file.write("""mdb.models['Model-1'].BoltLoad(name='Middle riveting', createStepName='Pretension', re-
gion=region, magnitude=""" + str(f_cent) + """, boltMethod=APPLY_FORCE, datumAxis=datumAxis)""" + "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("s1 = a.instances['DIN660-" + diameter + "-" + shank + "-3'].faces" + "\n")
    file.write("side2Faces1 = s1.getSequenceFromMask(mask=('[#1 ]', ), )" + "\n")
    file.write("region = a.Surface(side2Faces=side2Faces1, name='Surf-30')" + "\n")
    file.write("datumAxis = mdb.models['Model-1'].rootAssembly.datums[45]" + "\n")
    file.write("""mdb.models['Model-1'].BoltLoad(name='Outer riveting', createStepName='Pretension', region=region,
magnitude=""" + str(f_out) + """, boltMethod=APPLY_FORCE, datumAxis=datumAxis)""" + "\n")

    file.write("leaf = dgm.Leaf(leafType=DEFAULT_MODEL)" + "\n")
    file.write("""session.viewports['Viewport: 1'].assemblyDisplay.displayGroup.replace( leaf=leaf)""" + "\n")
    file.write("""session.viewports['Viewport: 1'].view.setValues(nearPlane=135.326, farPlane=242.54, width=84.5445,
height=40.9688, cameraPosition=(164.577, 84.9723, 37.2871), cameraUpVector=(-0.207934, -0.652731, 0.728496),
cameraTarget=(6.29769, 2.49684, 1.5022))""" + "\n")

    file.write("session.viewports['Viewport: 1'].assemblyDisplay.setValues(step='Traction')" + "\n")

    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("r1 = a.referencePoints" + "\n")
    file.write("refPoints1=(r1[42], )" + "\n")
    file.write("region = a.Set(referencePoints=refPoints1, name='Set-3')""" + "\n")
    file.write("""mdb.models['Model-1'].ConcentratedForce(name='Traction', createStepName='Traction', re-
gion=region, cf1=""" + str(f_trac) + """, distributionType=UNIFORM, field='', localCsys=None, follower=ON)"""
+ "\n")

    file.write("""session.viewports['Viewport: 1'].assemblyDisplay.setValues(mesh=ON, loads=OFF, bcs=OFF,
predefinedFields=OFF, connectors=OFF)""" + "\n")
    file.write("""session.viewports['Viewport: 1'].assemblyDisplay.meshOptions.setValues( meshTechnique=ON)"""
+ "\n")
    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("c1 = a.instances['DIN660-" + diameter + "-" + shank + "-1'].cells" + "\n")
```

```
    file.write("cells1 = c1.getSequenceFromMask(mask=('[#f ]', ), )" + "\n")
    file.write("c2 = a.instances['DIN660-" + diameter + "-" + shank + "-2'].cells" + "\n")
    file.write("cells2 = c2.getSequenceFromMask(mask=('[#f ]', ), )" + "\n")
    file.write("c3 = a.instances['DIN660-" + diameter + "-" + shank + "-3'].cells" + "\n")
    file.write("cells3 = c3.getSequenceFromMask(mask=('[#f ]', ), )" + "\n")
    file.write("c4 = a.instances['Plate-1'].cells" + "\n")
    file.write("cells4 = c4.getSequenceFromMask(mask=('[#1 ]', ), )" + "\n")
    file.write("c5 = a.instances['Plate-2'].cells" + "\n")
    file.write("cells5 = c5.getSequenceFromMask(mask=('[#1 ]', ), )" + "\n")
    file.write("pickedRegions = cells1+cells2+cells3+cells4+cells5" + "\n") file.write("a.setMeshControls(regions=pickedRegions,
algorithm=MEDIAL_AXIS)" + "\n")

    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("partInstances =(a.instances['DIN660-" + diameter + "-" + shank + "-1'], a.instances['DIN660-" +
diameter + "-" + shank + """-2'], a.instances['DIN660-""" + diameter + "-" + shank + """-3'], a.instances['Plate-
1'], a.instances['Plate-2'], )""" + "\n")
    file.write("a.seedPartInstance(regions=partInstances, size=" + str(m_size) + """, deviationFactor=0.1, min-
SizeFactor=0.1)""" + "\n")

    file.write("a = mdb.models['Model-1'].rootAssembly" + "\n")
    file.write("partInstances =(a.instances['DIN660-" + diameter + "-" + shank + "-1'], a.instances['DIN660-" +
diameter + "-" + shank + """-2'], a.instances['DIN660-""" + diameter + "-" + shank + """-3'], a.instances['Plate-
1'], a.instances['Plate-2'], )""" + "\n")
    file.write("a.generateMesh(regions=partInstances)" + "\n")

    file.write("session.viewports['Viewport: 1'].assemblyDisplay.setValues(mesh=OFF)" + "\n")
    file.write("""session.viewports['Viewport: 1'].assemblyDisplay.meshOptions.setValues( meshTechnique=OFF)"""
+ "\n")

    file.write("""session.viewports['Viewport: 1'].assemblyDisplay.setValues(interactions=OFF, constraints=OFF,
engineeringFeatures=OFF)""" + "\n")
    file.write("""mdb.models['Model-1'].fieldOutputRequests['F-Output-1'].setValues(variables=( 'S', 'PE', 'PEEQ',
'PEMAG', 'LE', 'U', 'RF', 'CF', 'NFORC', 'CSTRESS', 'CDISP', 'CFORCE'))""" + "\n")
    file.write("mdb.Job(name='Job-" + valor_fil + """', model='Model-1', description='', type=ANALYSIS, at-
Time=None, waitMinutes=0, waitHours=0, queue=None, memory=90, memoryUnits=PERCENTAGE, getMem-
oryFromAnalysis=True, explicitPrecision=SINGLE, nodalOutputPrecision=SINGLE, echoPrint=OFF, model-
Print=OFF, contactPrint=OFF, historyPrint=OFF, userSubroutine='', scratch='', multiprocessingMode=DEFAULT,
numCpus=1, numGPUs=0)""" + "\n")
    file.write("mdb.jobs['Job-" + valor_fil + "'].submit(consistencyChecking=OFF)" + "\n")
    file.close()

    # With this line, the function ends. Thus, when copying the rest of the code you need to tabulate back to
the left.


    import pandas as pd
    cases= pd.read_excel('Cases.xlsx')
    models= cases.iloc[:,0]
    for iter in models:
    # With this line, the loop begins. Thus, when copying the rest of the code you need to tabulate them to the
right.
    #inner rivet shank diamter
    dr_s_1= cases.iloc[iter-1,1]
    #middle rivet shank diameter
    dr_s_2= cases.iloc[iter-1,2]
```

```
    #outer rivet shank diameter
    dr_s_3= cases.iloc[iter-1,3]
    #inner rivet head diameter
    dr_h_1= cases.iloc[iter-1,4]
    #middle rivet head diameter
    dr_h_2= cases.iloc[iter-1,5]
    #outer rivet head diameter
    dr_h_3= cases.iloc[iter-1,6]
    #rivet head height
    hr= cases.iloc[iter-1,7]
    #inner hole diameter
    d_h_1= cases.iloc[iter-1,8]
    #middle hole diameter
    d_h_2= cases.iloc[iter-1,9]
    #outer hole diameter
    d_h_3= cases.iloc[iter-1,10]
    #upper plate thickness
    t_1= cases.iloc[iter-1,11]
    #lower plate thickness
    t_2= cases.iloc[iter-1,12]
    #offset between plates in x-direction
    o_x= cases.iloc[iter-1,13]
    #edge distance
    e= cases.iloc[iter-1,14]
    #rivets pitch
    p= cases.iloc[iter-1,15]
    #Pretension force for inner rivet
    f_in= cases.iloc[iter-1,16]
    #Pretension force for middle rivet
    f_cent= cases.iloc[iter-1,17]
    #Pretension force for outer rivet
    f_out= cases.iloc[iter-1,18]
    #Traction force
    f_trac= cases.iloc[iter-1,19]
    #Mesh size
    m_size= cases.iloc[iter-1,20]
    RpyFileCreator(dr_s_1, dr_s_2, dr_s_3, dr_h_1, dr_h_2, dr_h_3, hr, d_h_1, d_h_2, d_h_3, t_1, t_2, o_x, e, p,
f_in, f_cent, f_out, f_trac, m_size, iter)
```

Bibliography

[1] R. Hojjati-Talmei, M.A. Wahab, T. Yue and L. D'Alvise, "*On fretting fatigue behaviour of single bolted lap joint*", International Journal of Fracture Fatigue and Wear, Volume 2, 2014.

[2] Why Are Airplanes Manufactured With Riveted Joints Instead of Welded? (2020). [Online] Available in: https://lc.cx/SHyP_q [Watched: October 9th, 2024].

[3] Types of Rivets: A Guide to Process, Uses, And Materials (2024). [Online] Available in: https://lc.cx/CZNACB [Watched: October 10th, 2024].

[4] The Basics of Floating Anchor Nuts and How They Work (2022). [Online] Available in: https://monroeaerospace.com/blog/the-basics-of-floating-anchor-nuts-and-how-they-work/ [Watched: December 20th, 2024].

[5] G. Saputro, M. Akhlis Rizza, U. S. Amrullah, and H. Wicaksono, "*The effect of rivet gun operating pressure and hole clearance on rivet shear strength in sheet metal riveting process*" Jun. 2024, DOI: https://doi.org/10.11113/jurnalteknologi.v86.20999

[6] J. Běhal and R. Růžek, "*Effect of the rivet-hole tolerance on the stress-severity factor*" vol. 55, no. 2, p. 237, Apr. 2021, DOI: https://doi.org/10.17222/mit.2020.146

[7] J. Mucha and W. Witkowski, "*The Structure of the Strength of Riveted Joints Determined in the Lap Joint Tensile Shear Test*" vol. 9, no. 1, p. 44, Mar. 2015, DOI:https://doi.org/10.1515/ama-2015-0009

[8] A. Lipski, "*The Influence of The Degree of the Rivet Hole Sizing on the Fatigue Life*" vol. 2012, no. 4, Jan. 2012, DOI:https://doi.org/10.2478/v10164-012-0057-2

[9] J.O. Dow, "Introduction to Problem Definition and Development" in *A Unified Approach to the Finite Element Method and Error Analysis Procedures*, 1999. DOI: https://doi.org/10.1016/B978-0-12-221440-0.X5027-4.

[10] Vigé, D. (2010). 13 - Vehicle interior noise refinement – cabin sound package design and development. In X. Wang (Ed.), *Vehicle Noise and Vibration Refinement* (pp. 286–317). Woodhead Publishing. DOI:https://doi.org/10.1533/9781845698041.3.286

[11] "Advanced aircraft deisgn" class notes, Escuela de Arquitectura, Ingeniería, Ciencia y Computación, Universidad Europea de Madrid, winter quarter, 2024.

[12] C. M. Wai, A. Rivai, and O. Bapokutty, "*Modelling optimization involving different types of elements in finite element analysis*" vol. 50, Dec. 2013, DOI: https://doi.org/10.1088/1757-899x/50/1/012036.

[13] P. Bolcos. Hyperelasticity and Viscoelasticity in Abaqus (2021). [Online] Available in: https://lc.cx/PVfXSF [Watched: October 13th, 2024].

[14] Y.Çapar. Material Hardening Laws (2021). [Online] Available in: https://lc.cx/f2HKSh [Watched: October 15th, 2024].

[15] R. Kuliiev, S. Keller, and N. Kashaev, "*Identification of Johnson-Cook material model parameters for laser shock peening process simulation for AA2024, Ti–6Al–4V and Inconel 718*" vol. 28, p. 1975, Dec. 2023, DOI:https://doi.org/10.1016/j.jmrt.2023.11.168

[16] V. Kumar Reddy Sirigiri, V. Yadav Gudiga, U. Shankar Gattu, G. Suneesh, and K. Mohan Buddaraju, "*A review on Johnson Cook material model*" vol. 62, pp. (3450–3456), 2022, DOI: https://doi.org/10.1016/j.matpr.2022.04.279.

[17] Contact In FEA – What Is It And How Should It Be Implemented? [Online] Available in: https://lc.cx/GTg_e4 [Watched: October 17th, 2024].

[18] C. Obbink-Huizer. Implicit Vs Explicit Finite Element Analysis: When to Use Which? (2021). [Online] Available in: https://lc.cx/w0ZuT7 [Watched: October 15th, 2024].

[19] A. S. Dabit, A. E. Lianto, S. A. Branta, F. B. Laksono, A. R. Prabowo, and N. Muhayat, "*Finite Element Analysis (FEA) on Autonomous Unmanned Surface Vehicle Feeder Boat subjected to Static Loads*" vol. 27, p. 163, 2020, DOI:https://doi.org/10.1016/j.prostr.2020.07.022

[20] S. M. M. Tahir and N. M. Jali, "*An impact loading simulation of Rubber-Toughened Poly Methyl Methacrylate (RT-PMMA) using Abaqus Explicit*" vol. 15, no. 2, Aug. 2023, DOI:https://doi.org/10.1016/j.asej.2023.102392

[21] M. C. Y. Niu, "*Airframe stress analysis and sizing*". 2nd ed: Adaso, 2001.

[22] DIN660, "*Metric DIN 660 Round Head Solid Rivets*".

[23] "*Online Materials Information Resource - MatWeb - AISI Type 304 Stainless Steel*". Available in https://asm.matweb.com/search/SpecificMaterial.asp?bassnum=mq304a. [Watched: October 15th, 2024]

[24] "*Online Materials Information Resource - MatWeb - Aluminum 2024-T3*". Available in https://www.matweb.com/search/datasheet.aspx?MatGUID=57483b4d782940faaf12964a1821fb61&ckck=1. [Watched: October 15th, 2024]