Numerical and Experimental Characterization of Stress Concentration Factors on PDMS

ELIMAR WÜRMLI JÚNIOR

Final Report of Dissertation presented to

Escola Superior de Tecnologia e de Gestão

Instituto Politécnico de Bragança

For Obtaining a Master's Degree in

INDUSTRIAL ENGINEERING

November 2020

Numerical and Experimental Characterization of Stress Concentration Factors on PDMS

ELIMAR WÜRMLI JÚNIOR

Final Report of Dissertation presented to

Escola Superior de Tecnologia e de Gestão

Instituto Politécnico de Bragança

For Obtaining a Master's Degree in

INDUSTRIAL ENGINEERING

Supervisor from Instituto Politécnico de Bragança

Prof. Dr. João Eduardo Pinto Castro Ribeiro

Supervisor from Universidade Tecnológica Federal do Paraná

Prof. Dr. Marcelo Vasconcelos Carvalho

November 2020

Escola Superior de Tecnologia e de Gestão is not responsible for the opinions expressed in this document.

I dedicate this work to my wife, who helped me all the time during this journey, supporting me in everything.

Acknowledgements

First of all, I would like to thank to God, who allowed me to get where I am. For of him, and through him, and to him, are all things, to whom be glory for ever.

I would like to thank to my wife, who accepted to do this journey with me, and always were able to help me in everything. I love you. Also I would like to thank also to my parents and sisters, who gave me the support to go on and always taught me the road I should walk.

I also would like to thank my supervisors, Prof. Dr. João Eduardo Pinto Castro Ribeiro, by the support, the lectures, the helps and the patience with me during this work, and Prof. Dr. Marcelo Vasconcelos Carvalho, who helped me to get where I am today and encouraged me to do this great challenge in my life.

A great acknowledgements to the Federal University of Technology, Paraná, on Ponta Grossa, for gave me the opportunity to conclude my studies abroad, also to the Polythecnical Institute of Bragança, for receiving me so well and have opened its doors to me.

Resumo

O Polidimetilsiloxano (Polydimethylsiloxane (PDMS)) é um polímero com cadeias baseadas em silício que apresenta excelentes qualidades para ser usado na indústria biomédica (inerte, transparente, inodoro) além de uma elevada elasticidade que permite que este seja modelado como um material hiperelástico. Existem inúmeros modelos constitutivos que descrevem o comportamento de tal material, mas ainda não existem muitos estudos que levam em consideração concentradores de tensões nesses materiais. Sendo assim, este trabalho tem como objetivo avaliar a influência de três diferentes concentradores de tensão em materiais hiperelásticos através de simulações numéricas com três diferentes modelos constitutivos, e também comparar estes modelos constitutivos com os valores reais de deformação desses materiais, através de testes de Correlação Digital de Imagem (CDI), de modo a observar qual dos modelos estudados é mais adequado para concentradores de tensão. Os três espécimes foram projetados de modo a cada espécime ter um tipo diferente de concentrador de tensão. Foram fabricados os modelos em PDMS, realizados os ensaios uniaxiais de tração e então foram avaliadas as deformações através das imagens captadas por meio da técnica de CDI. Em conjunto, os projetos foram inseridos em ANSYS para as simulações para cada modelo constitutivo e então foi feita a comparação entre as simulações por meio de ferramentas estatísticas (Análise de Variância e teste de Tukey). Pôde-se observar que não existiu diferença significativa entre os modelos e que os concentradores de tensão diferem muito do que ja conhecemos em outros materiais.

Palavras-chave: PDMS, modelos constitutivos, concentradores de tensão, correlação digital de imagem.

Abstract

Polydimethylsiloxane (PDMS) is a polymer with silicon-based chains that has excellent qualities to be used in the biomedical industry (inert, transparent, odorless) in addition to a high elasticity that allows it to be modeled as a hyperelastic material. There are numerous constitutive models that describe the behavior of such material, but there are not many studies that consider stress concentrators in these materials. Therefore, this work aims to evaluate the influence of three different stress concentrators on hyperelastic materials through numerical simulations with three different constitutive models, and also to compare these constitutive models with the real deformation values of these materials, through Digital Image Correlation (DIC) tests, in order to observe which of the studied models is most suitable for voltage concentrators. The three specimens were designed so that each specimen has a different type of stress concentrator. PDMS models were -manu-factured, uniaxial tensile tests were performed and then deformations were evaluated through images captured using the DIC technique. At the same time, the projects were also made in ANSYS for the simulations for each constitutive model and then the comparisons between the simulations were made using statistical tools (Analysis of Variance and Tukey test). It was observed that there was no significant difference between the models and that the stress concentrators differ a lot from what we already know in other materials.

Keywords: PDMS, constitutive models, stress concentrators, digital image correlation.

Contents

List of Tables

List of Figures

Signs And Acronyms List

K Stress Concentration Factor. 8, 9, 52, 54

$\boldsymbol{C}$ Right Cauchy-Green Tensor. 5

$\boldsymbol{C}$ Left Cauchy-Green Tensor. 5

$\boldsymbol{F}$ Deformation Gradient. 3

$\boldsymbol{R}$ Rotation Tensor. 5

$\boldsymbol{U}$ Stretch Tensor. 5

$\boldsymbol{\Psi}$ Helmholtz Free-Energy Function. 6

$\boldsymbol{\sigma}$ Stress Tensor. 5

$\vec{\boldsymbol{X}}$ reference body. 4

$\vec{\boldsymbol{x}}$ deformed body. 4

CNC Computer Numerical Control. 17

DIC Digital Image Correlation. 2, 23, 27, 29, 39, 53

IPB Instituto Politécnico de Bragança. 17, 20

PDMS Polydimethylsiloxane. vii, 1, 2, 17, 18

Chapter 1

Introduction

1.1 Contextualization of the Work

It is a known fact that computers are changing our life, improving and automatizating process which previously were manual. The mechanical properties measurement could not be out of this change. With acquisition image systems and computers, it is possible to see almost exactly how each part of a body behaves under an external load. This can be very used for material that do not have its behavior well known, like general polymers, and more specific, rubber-like materials.

The application of high-performance polymers in the world is well known, helping the development of micro and nanotechnologies, due its rapid prototyping and high-fidelity system construction. Between these polymers, there is the poly(dimethyl)siloxane (PDMS), that is also very used in biomedical applications [1].

In order to understand the behaviour of these high-performance polymers, many constitutive models were developed to predict the behavior of these material under external forces. Such models are based on a strain energy density function, that can give the stress-strain behavior of each model [2], [3].

1.2 Motivations and Objectives

With such models available on the bibliography, one may wonder which model is the best for a specific application that requires a stress concentrator in the material. Therefore, the main goal of this work is to verify the constitutive models behavior in certain circumstances by comparing the displacement and strain fields on PDMS under uniaxial tensile test through Digital Image Correlation (DIC) experimental model with numerical simulations using different constitutive models in the same conditions in order to find which model is suitable to characterize stress concentrators (once there is no calculations or studies for stress concentrators in hyperelastic materials), and verify the behavior of the stress concentrators under these circumstances , by comparing the simulations with the stress concentrators with a simulation without the stress concentrator.

For such main goal, some specific goals were needed, such as build the mold of the desired profiles, the specimens manufacturing, the image captures during the uniaxial test, the numerical simulation of the experimental test, and finally the code development that compares the simulations with the DIC test.

1.3 Structure of the Work

This work presents 6 (six) main chapters, being the first one the introduction of the work and the main aspects of this one, the second a theoretical review of the principal needed topics to the understanding of this research, the third approaches the methodology used in the work to obtain the results, the fourth is a presentation of the results, the fifth is a discussion of the results obtained, involving some statistic methods, and the last one (sixth) is the conclusion of the work, with some proposal for future work.

Chapter 2

Theoretical Review

2.1 Hiperelasticity

Hyperelasticity is a generalization of the linear elasticity that allows the analysis of large non-linear deformations [4].

For the very beginning of this subject, it's necessary to understand on what is based this theory. The **Continuum Theory** is well known in the engineering, described by inumerous authors, like Holtzapfel [2], Malvern [3], Bergstrom [4], and its subject is well developed in the engineering. However, some aspects of its theory will be reviewed along this subject.

2.1.1 Stress-Strain Relations

The **Continuum Theory** uses the concepts of **Tensorial Algebra** for its own development, as well as the principal equations on the engineering (Mass Balance, Momentum Balance,...). In this way, a series of mathematical elements, relations and concepts are developed, which allows us to understand the behaviour of hyperelastic materials. In the following, will be presented some of these concepts, that will be useful in some future approachs.

The Deformation Gradient ($\boldsymbol{F}$) is a Tensor, that relates the variation of the vector of

the Deformed body ($\vec{\boldsymbol{x}}$), represented in 2.1 with respect to the vector of the Reference body ($\vec{\boldsymbol{X}}$), represented in 2.2 [4], [5], defined as follows:

$$\vec{\boldsymbol{x}} = \{x_1, x_2, x_3\} \tag{2.1}$$

$$\vec{\boldsymbol{X}} = \{X_1, X_2, X_3\} \tag{2.2}$$

$$\boldsymbol{F} = \frac{\partial \vec{\boldsymbol{x}}(\vec{\boldsymbol{X}}, t)}{\partial \vec{\boldsymbol{X}}} \tag{2.3}$$

As an example, one can imagine an uniaxial loading shown in Figure 2.1, with initial length $\boldsymbol{L_0}$ and final length $\boldsymbol{L}$.

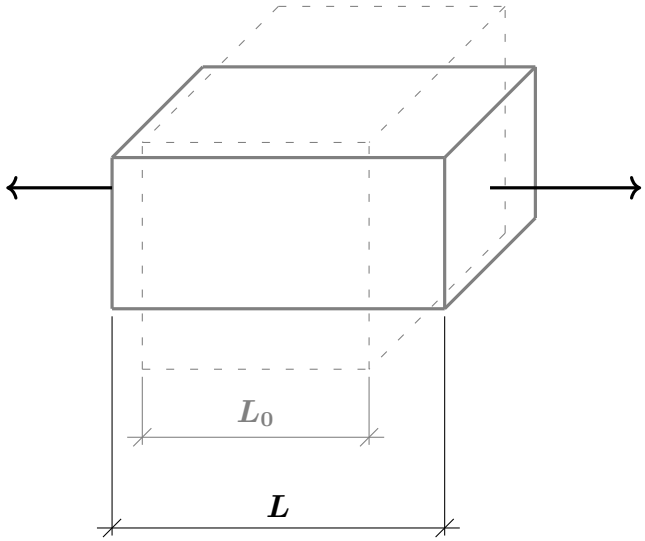


Figure 2.1: An uniaxial loading

For this case, the stretch ratio between the bodies can be described as $\mu = L/L_0$, and, as the volume remains constant during the process, it can be shown that:

$$x_1 = \mu X_1 \qquad x_2 = \frac{1}{\sqrt{\mu}} X_2 \qquad x_3 = \frac{1}{\sqrt{\mu}} X_3$$

Which implies:

$$\boldsymbol{F} = \begin{bmatrix} \mu & 0 & 0 \\ 0 & 1/\sqrt{\mu} & 0 \\ 0 & 0 & 1/\sqrt{\mu} \end{bmatrix}$$

The tensor $\boldsymbol{F}$ can be decomposed in various other tensors, as follows [2], [5]:

- The Stretch Tensor ($\boldsymbol{U}$) an the Rotation Tensor ($\boldsymbol{R}$) - Convert the Deformation Gradient in a pure stretch and a pure rotation, through the equation $\boldsymbol{F} = \boldsymbol{RU}$

- The Right Cauchy-Green Tensor ($\boldsymbol{C}$), which is defined as $\boldsymbol{C} = \boldsymbol{F^T F}$, and has as invariants (I_1, I_2 and I_3) as follows:

$$I_1 = \lambda_1^2 + \lambda_2^2 + \lambda_3^2$$

$$I_2 = \lambda_1^2\lambda_2^2 + \lambda_1^2\lambda_3^2 + \lambda_2^2\lambda_3^2$$

$$I_3 = \lambda_1^2\lambda_2^2\lambda_3^2$$

  where:

  - λ_1, λ_2 and λ_3 are the eigenvectors of $\boldsymbol{F}$, and also the principal stretches along the directions 1, 2 and 3;

- The Left Cauchy-Green Tensor ($\boldsymbol{C}$), which is defined as $\boldsymbol{b} = \boldsymbol{F F^T}$ and has the same variants of the Right Cauchy-Green Tensor ($\boldsymbol{C}$).

The Stress Tensor ($\boldsymbol{\sigma}$), also called Cauchy Stress Tensor is defined analyzing how much a material will deform when subject to external loads [2], [3], [5]. In this way, the stress tensor can be shown as a tensor that transform a vector that is normal to an area of application $\vec{\boldsymbol{n}}$ in a vector that is the real vector of traction acting on the same area $\vec{\boldsymbol{t}}$ (See Figure 2.2), as shown below:

$$\vec{\boldsymbol{t}} = \boldsymbol{\sigma}\vec{\boldsymbol{n}}$$

This tensor is a symmetric tensor [2], having six components that describes the nature of the loading.

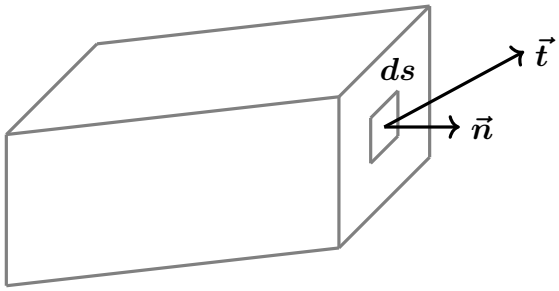


Figure 2.2: Vector $\vec{t}$ and $\vec{n}$ applied at a deformed body

2.1.2 Hyperelastic Materials

Also called as Green elastic material, this one still is **elastic**, that means it returns to it's original shape once the forces have been removed. It's a material that the stress-strain relation derives from a Strain Energy Density function, also called Helmholtz Free-Energy Function ($\boldsymbol{\Psi}$) [2], [3], [5].

This function can be related with the Cauchy Stress Tensor as follows [5]:

$$\boldsymbol{\sigma} = \frac{2}{J}\left[\frac{\partial \boldsymbol{\Psi}}{\partial I_1^*} + \frac{\partial \boldsymbol{\Psi}}{\partial I_2^*} I_1^*\right]\boldsymbol{b}^* - \frac{2}{J}\frac{\partial \boldsymbol{\Psi}}{\partial I_2^*}(\boldsymbol{b}^*)^2 + \left[\frac{\partial \boldsymbol{\Psi}}{\partial J} - \frac{2I_1^*}{3J}\frac{\partial \boldsymbol{\Psi}}{\partial I_1^*} - \frac{4I_2^*}{3J}\frac{\partial \boldsymbol{\Psi}}{\partial I_2^*}\right]\boldsymbol{I} \tag{2.4}$$

Where:

- $\boldsymbol{\Psi}$ is the **Helmholtz Free-Energy Function**;
- J is the determinant of $\boldsymbol{F}$;
- $\boldsymbol{b}^*$ is the **Isochoric left Cauchy-green Tensor** ($\boldsymbol{b}^* = J^{-2/3}\boldsymbol{b}$);
- $I_1^* = J^{-2/3} I_1$ and $I_2^* = J^{-4/3} I_2$;
- $\boldsymbol{I}$ is the Isotropic Tensor (an unitary matrix).

The equation 2.4 depends ultimately on the principal stretches λ_1, λ_2 and λ_3 and of $\boldsymbol{\Psi}$. If the principal stretches where known, also the Helmholtz Free-Energy Function, it's possible to determine the stress-strain behavior of material. In Hyperelasticity, the function $\boldsymbol{\Psi}$ ultimately also depends on λ_1, λ_2 and λ_3 [6].

The constitutive models for hyperelastic material are models that define the strain-energy density function ($\mathbf{\Psi}$) and can be classified according their types of formulations (The approach that was used to get in that strain-energy function) [7]:

- The phenomenological models - Such models came from mathematical developments of the strain-energy density function ($\mathbf{\Psi}$) and are usually hard to deretmine their constants.

- The experimental models - Developed by directly observations of experimental behavior of materials

- The physically-based models - These models are developed by observations of the behavior of polymeric chains network and by statistical approaches.

Mooney-Rivlin Model

This model can be derived from a hyperelastic model made of a polynomial series of $(I_1 - 3)$ and $(I_2 - 3)$ [7]–[9], as shown in Equation 2.5:

$$\mathbf{\Psi} = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} C_{ij} (I_1 - 3)^i (I_2 - 3)^j \tag{2.5}$$

where:

$$C_{00} = 0 \qquad C_{10} + C_{01} = \nu \qquad \sum_{i+j=n}^{\infty} C_{ij} = 0 \quad \forall n \in N - \{1\} \tag{2.6}$$

where ν is the infinitesimal shear modulus. Usually, this equation is used for i and j minor or equal to 3 [7]. The constants are defined by the stress-strain curve of the material. The condition on 2.6_1 imposes the null energy in the reference position, while the condition on 2.6_2 guarantees a relationship with the linear elasticity theory, and 2.6_3 allows linear relationships between torque, shear stress and shear strain, and in the amount of twist in simple shear and torsion, respectively [9].

Ogden Model

Ogden [10] proposed an alternative form of the strain-energy function. According to him, the strain energy can be written by the following:

$$\boldsymbol{\Psi} = \sum_{i=1}^{\infty} \mu_i \phi(\alpha_i) \tag{2.7}$$

Where:

$$\boldsymbol{\phi}(\alpha_i) = (\lambda_1^{\alpha_i} + \lambda_2^{\alpha_i} + \lambda_3^{\alpha_i} - 3)/\alpha_i \tag{2.8}$$

In both Equations (2.7 and 2.8) the μ_i and α_i are determined by the stress-strain curve of the material, just like the Mooney-Rivlin model. The lambdas (λs) are the principal stretches on the three directions (x, y, and z). Ogden [10] shows that for uniaxial tensions, a two-term strain-energy function produces very satisfactory results when it is compared with experimental data.

Neo Hookean Model

This model consider the variation of $\boldsymbol{\Psi}$ only in function of the I_1 invariant [11], as shown in Equation 2.9.

$$\boldsymbol{\Psi} = C_1(I_1 - 3) \tag{2.9}$$

2.2 Stress Concentration Factor

The Stress Concentration Factor (K) is a ratio, between the highest value of the stress at a discontinuity and the nominal stress that could be applied if there where no discontinuity [12], [13], as shown in Equation 2.10. There are a lot of possible discontinuities, each one with its current K. For metallic materials, there is known relationships between the K and the geometric parameters of the material. As an example, one has a plate with a center hole, as shown in figure 2.3.

$$K = \frac{Highest\ possible\ value\ of\ tension\ supported\ by\ the\ element}{Nominal\ stress\ at\ minimum\ cross-section} \tag{2.10}$$

This K is a theoretical factor, derived, or from the theory of elasticity, or from laboratory stress analysis experiments [12], [13].

For a plate with a hole, as shown in figure 2.3, for $d/H < 0.3$ the factor K is given [13] as:

$$K = 2 + \left(1 - \frac{d}{H}\right)^3 \tag{2.11}$$

Figure 2.3: A plate with a center hole.

For a plate with symmetric circular shoulder fillets, as shown in figure 2.4, K can be obtained by the figure 2.5 [13]:

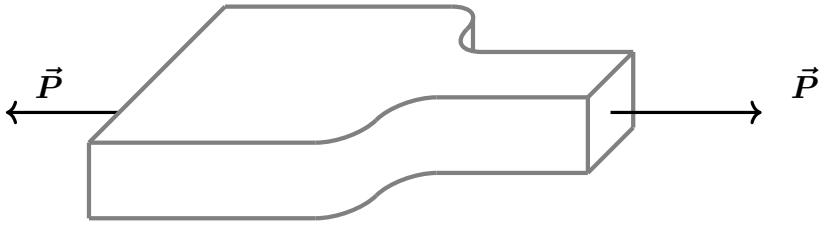


Figure 2.4: A plate with symmetric circular shoulder fillets.

For a plate with symmetric U-Shaped edge notches, as shown in figure 2.6, K is presented in figure 2.7 [13].

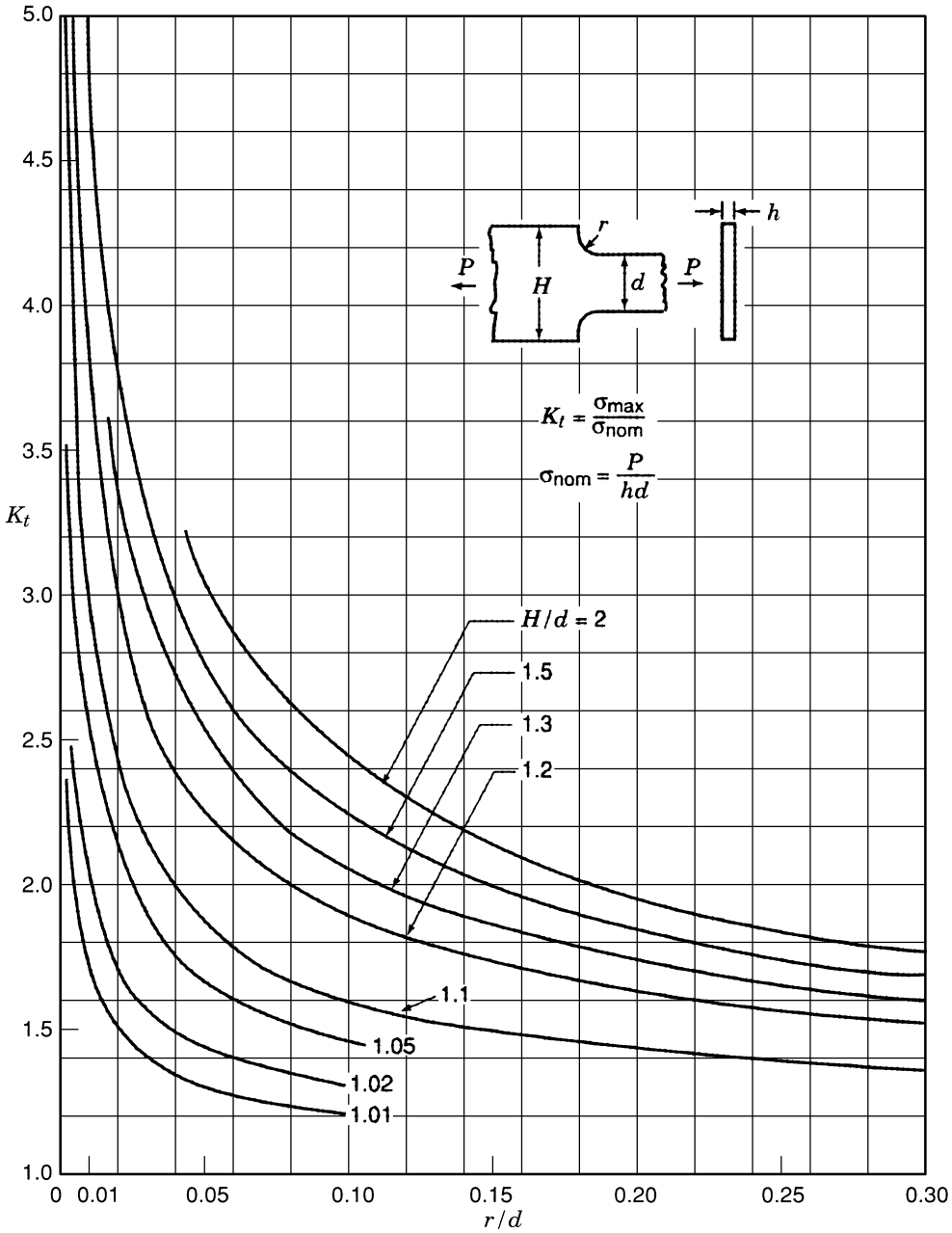


Figure 2.5: Stress Concentration Factor k for a plate with shoulder fillets.

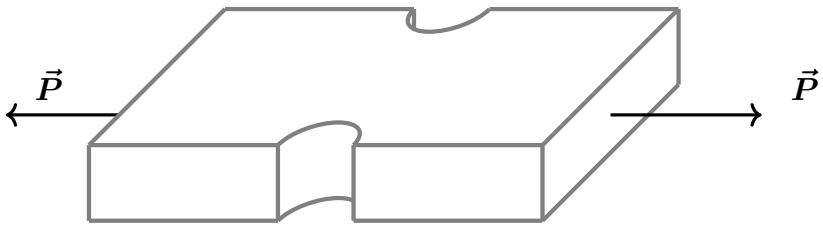


Figure 2.6: A plate with symmetric U-Shaped edge notches.

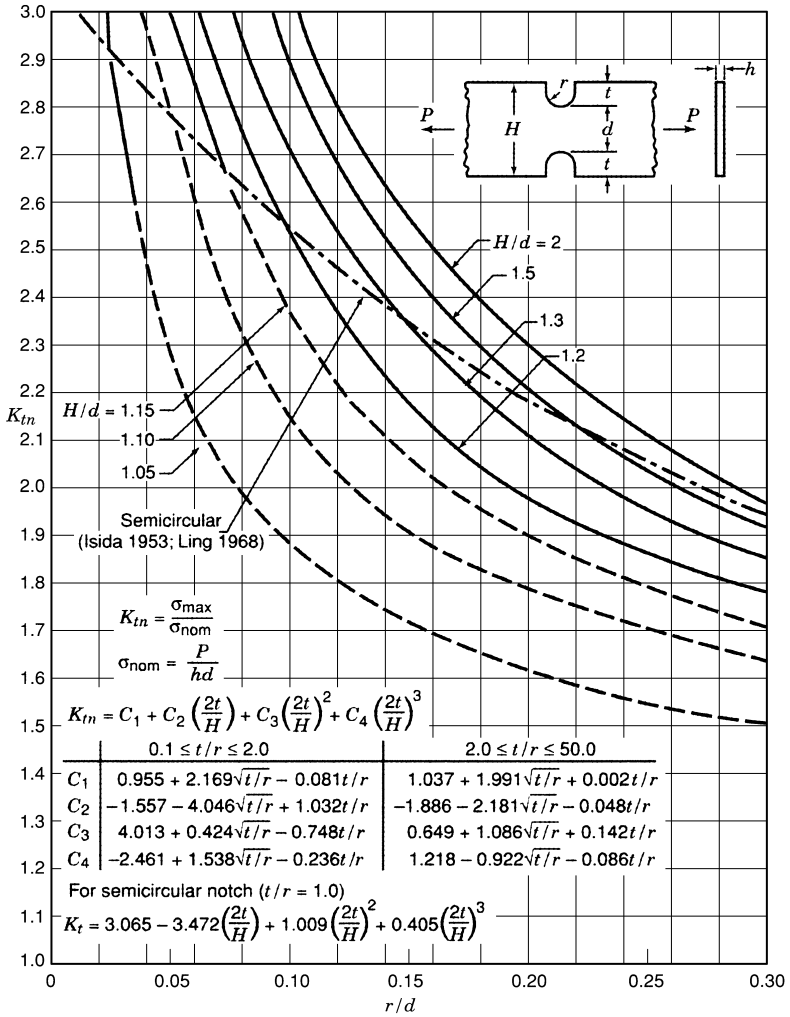


Figure 2.7: Stress Concentration Factor k for a plate with symmetric U-Shaped edge notches.

2.3 PDMS

The **polydimethylsiloxane** (**PDMS**) is a silicone elastomer, composed by dimetylsiloxy units and trimethylsilyl end groups, as shown in figure 2.8.

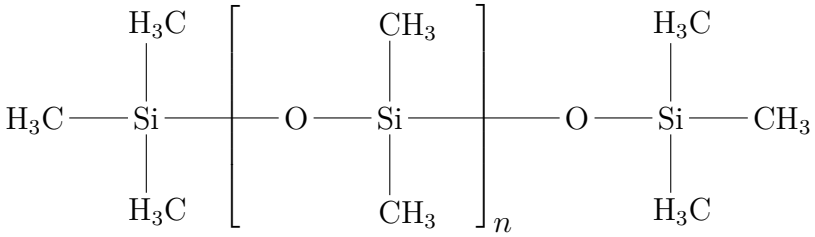
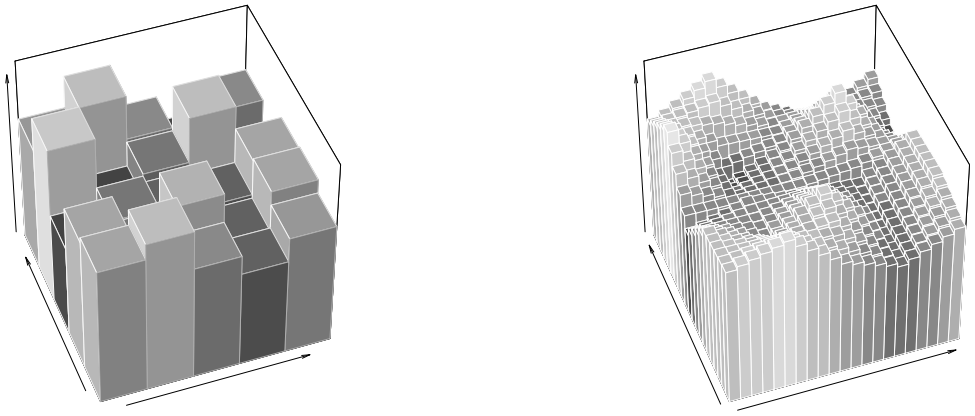


Figure 2.8: Polydimethylsiloxane

This polymer is clear, odourless and colourless at normal temperature and pressure, exhibiting a huge stability to chemical and oxidative degradation, it is permeable to gases, it is easy to handle and manipulate and doesn't require protective measures for its manipulation. [14]–[16]. These characteristics allows the usage of this polymer in biomedical microeletromechanical systems through Soft Lithography in the last years, and also in catheters, membrane oxygenators, drainage tubing, and many others [14]. The fabrication of PDMS is simple and can be made by mixing the curing agent and the prepolymer, cast this PDMS on a mold, cure, and then remove the cured PDMS from the mold [14], [17].

2.4 Digital Image Correlation

Digital Image Correlation (DIC) is a noncontacting optical full-field deformation measurement approach that allows to measure the strain and the deformation before, during and after the application of a load. A camera is used to take frames of the observed object and, with a software, this frame collection is converted in the strain field of the material [18]–[20]. A set of cameras can be used together to obtain a tridimensional analysis of material, or a single camera, to obtain a bidimensional analysis. Also, there is the need of a computer and lens that can produce the ideal image of the material [18], [21]. In this



(a) Pixels values of grey-scale distribution before Bi-linear Interpolation.

(b) Values of grey-scale distribution after Bi-linear Interpolation.

Figure 2.9: Bilinear Interpolation For a Subset of Points.

work, a bidimensional analysis was done, so this analysis is the one to be explored in this section. This technique is based in compare the image where the body is deformed with the image where the body is undeformed. First of all, the image is converted in grayscale matrix, where the value of each element is a number that represents the grey level of that pixel, and its position (i and j) in the matrix is the position of that pixel in the image. Then, to increase the continuity of the image, a bilinear interpolation is used, as shown in figure 2.9 and 2.10. So, the image is divided in a subset of pixels by blocks, that can be mapped in a reference position and then be found in any deformed image. for this reason, each block needs to be unique and with a high contrast and definition (for that, a lightning system is required) [21]. It is assumed that the deformation of the subsets are homogeneous, so the deformed position of a subset centroid can be expressed as a Taylor series, in function of the current position and the gradient of the deformation. In this way, it is possible to calculate the displacement field by finding the correct position of the subsets in the undeformed image. This is made calculating a correlation coefficient for all the subsets that are around the region that the subset initially was. The subset that

obtains the lower value of the coefficient is consider as the correct subset, and then the taylor series for this coefficient is calculated [20].

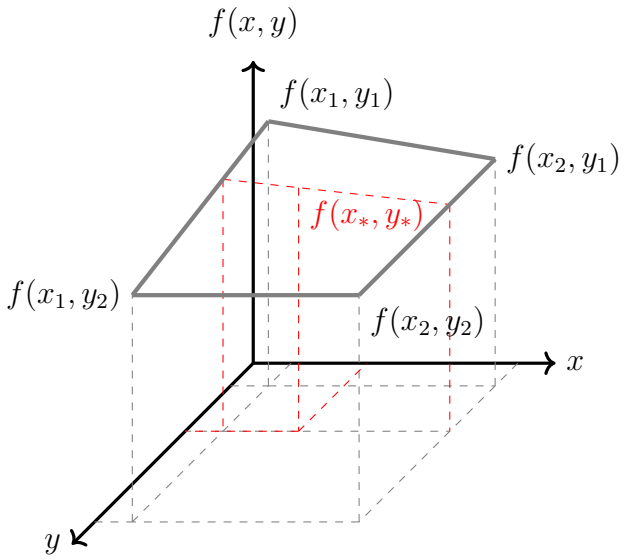


Figure 2.10: Bi-linear Interpolation Schema Based on Four Points.

There are some problems on doing this comparison because, usually, the material surface itself does not allows to know the correspondent pixels in the both images (one pixel of the undeformed body can be actually two or more pixels in the deformed body). This problem is called the Correspondence Problem [21]. So, to avoid this problem, the creation of an specific pattern in the body surface is useful. This pattern should be random in surface, without a preferred orientation, in order to avoid the Correspondence Problem by using repeated textures. This pattern also should be isotropic (its marks should have approx. the same size). Such pattern is called **Speckle Pattern**, and is shown in Figure 2.11.

This pattern can be either micro-scale (micrometers to nanometers) or macro-scale (meters to millimeters). In this work, macro-scale pattern was used. There are a lot of method used to apply a good speckle pattern in the specimen (by spray bottle, by airbrush, by spin coating, by stamp) [22] but one of the commonly used techniques is the spray (which was used in this work). This technique consists in paint the surface of one color and then spray another color over the previous paint (usually is black over white)

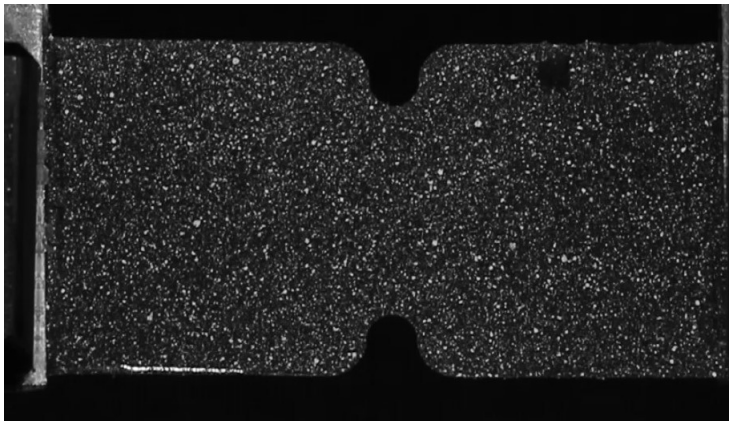


Figure 2.11: Speckle Pattern applied by spray

[18], as shown in Figure 2.11.

Chapter 3

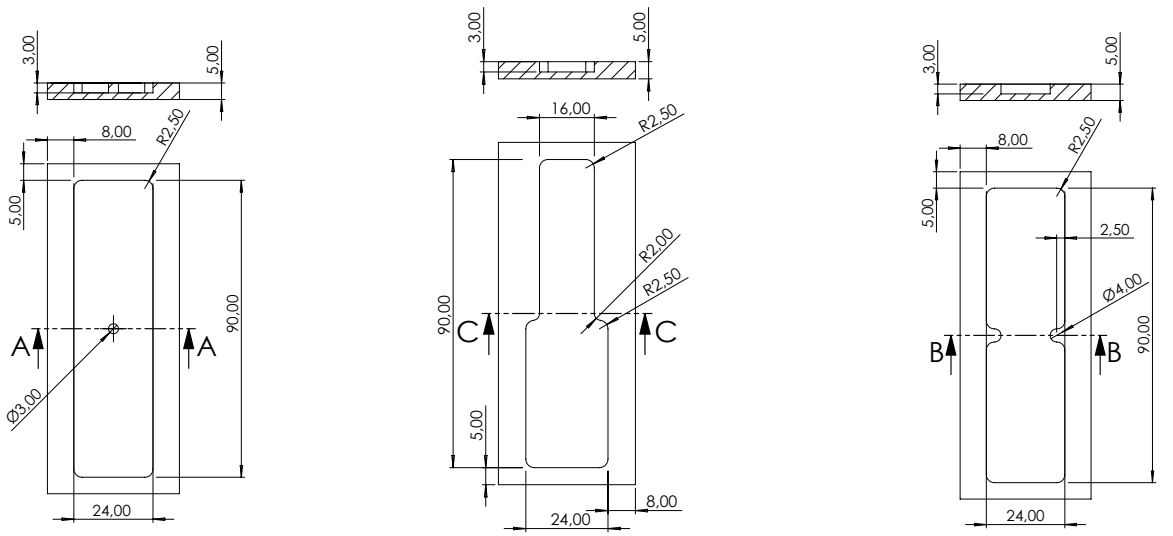
Metodology

3.1 Specimen Preparation

Four molds were needed to produce four different types of specimen: The first was the specimen used in the standart tensile, shown in the next section. The other three were projected by the author to have stress concentrators, shown in figure 3.1. The molds to obtain the PDMS specimen were made in Mechanical Technology Laboratory on Instituto Politécnico de Bragança (IPB), using a Computer Numerical Control (CNC) machine Deckel Maho DMC 63V, as shown in Figure 3.2.

The chosen hyperelastic material is the PDMS Sylgard 184 mixture of 10:1 preploymer and curing agent, in the Fluid Mechanics and Hydraulic Laboratory, also on IPB, using a becker, spatulas, a precision scale, the molds made previously, a vacuum pump and a vacuum chamber, shown in Figure 3.3. The procedure of the specimen confection is shown below:

1. Preparation of the mold (shown previously);
2. Preparation of the PDMS - The prepolymer and the curing agent were stirred in a becker with a spatula, and then placed into the vacuum chamber to apply the vacuum to remove the bubbles of PDMS;
3. Casting The PDMS into the molds - The PDMS was casted into the molds to made



(a) Type 1 - Center hole.

(b) Type 2 - Symmetric circ. shoulder fillets.

(c) Type 3 - Symmetric semicircular edge nodes.

Figure 3.1: Projected molds made for the specimen production.

the specimen, then the molds with PDMS were placed in the vacuum pump again to remove the remaining bubbles;

4. PDMS Curing - Now the PDMS needed some time for curing, so, one entire week were used for this process, at room temperature.

After these four steps, the specimen were ready for the tests.

To perform the Digital Image Correlation, it is necessary a speckle pattern preparation on the surfaces of the material, in order to allow the camera (and consequently the Digital Image Correlation software) to identify the surface and evaluate the deformations in it. Such patterns were produced using black and white opaque spray paints, pulverized over the surface of the specimens in order to produce a speckle pattern on them. In the first specimen, the white paint was used as background while the black paint was used to generate the pattern. In the second specimen, as the specimen is transparent, only the black paint was used. In the third specimen, the black paint was used as background while the white paint was used to produce the patterns. All the patterns in the parts are presented in Figure 3.4.

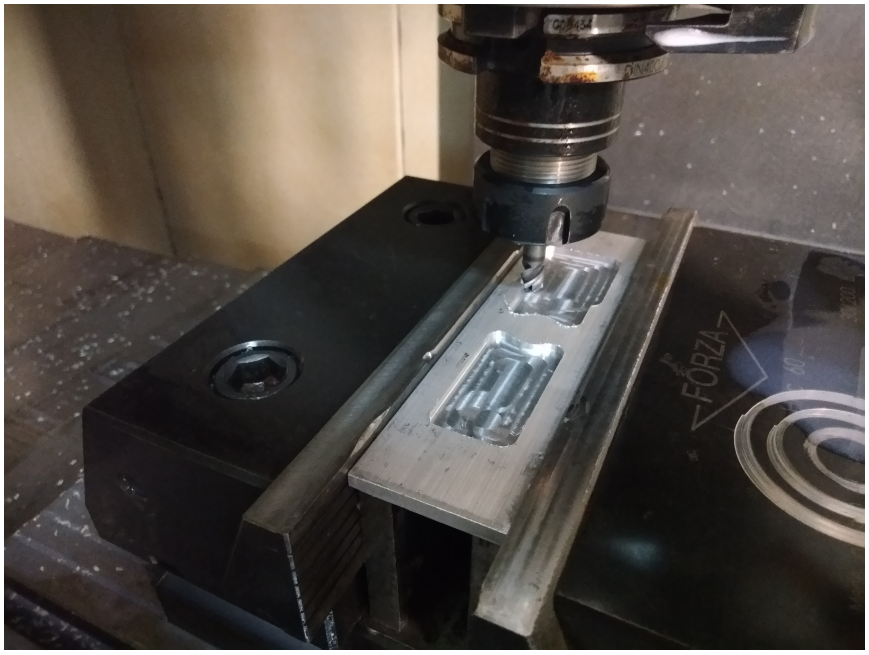
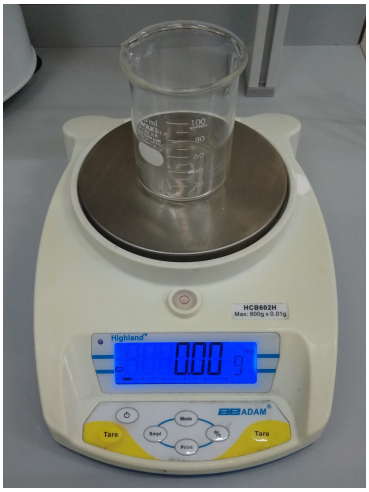
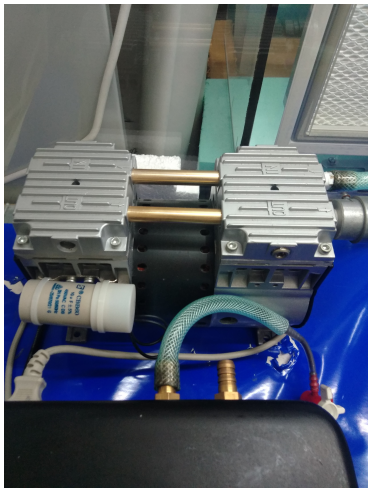


Figure 3.2: Mold being made in a CNC machine.



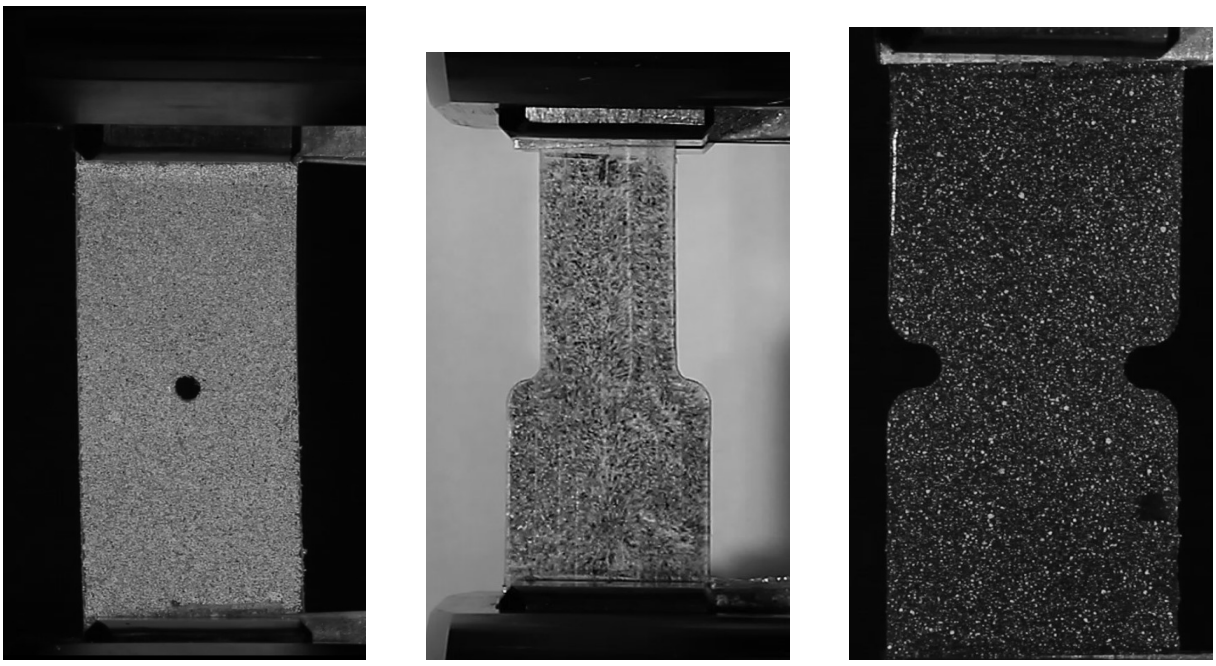
(a) Precision Scale and Becker

(b) Vacuum Chamber



(c) Vacuum Pump

Figure 3.3: Used materials in specimen preparation.



(a) Specimen 1 (b) Specimen 2 (c) Specimen 3

Figure 3.4: Speckle Pattern Applied in the specimen.

3.2 Standard Tensile Tests

The Hiperelastic Models shown in Section 2 need specific constants for the correct definition of the curve. These constants are obtained from the material stress-strain curve, so a standard test is necessary. The material was tested in Strenght of Materials and Structures Laboratory on IPB, on a Shimadzu Autograph AGS-X 10KN universal test machine. The used standart is BS 2782 that defines the ideal dimensions and parameters of the specimen, shown in Figure 3.5. The stress-strain data is shown in Figure 3.6. In this figure there is a zoomed region that is possible to see a slightly variation of the data that can disturb the behavior of the simulation on the numerical analysis. So, in order to achieve a better behavior of the simulation, it is necessary to do a curve adjustment. The chosen method was the **loess** (local weighted regression), an old and well known method developed by Cleveland and Devlin [23] that fits a curve based on a parametric function for estimate the points in a defined neighborhood, through a smoothness parameter (span). The higher the span, the smoother the fitting. For this test, three different spans

were chosen for trying and it can be seen also in Figure 3.6, as being red (10% span), blue (25% span) and green (50% span). All of them seems to present a good approximation for the test points (the lines are overlapped), so it was chosen the 50% (fifty percent) loess model. The plots and the calculations of the points were made in R software. The adapted points and the code are in the Appendix A.

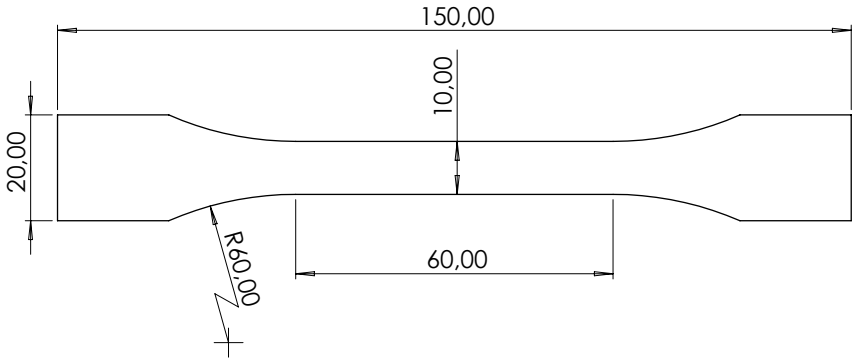


Figure 3.5: Specimen for standard tensile test.

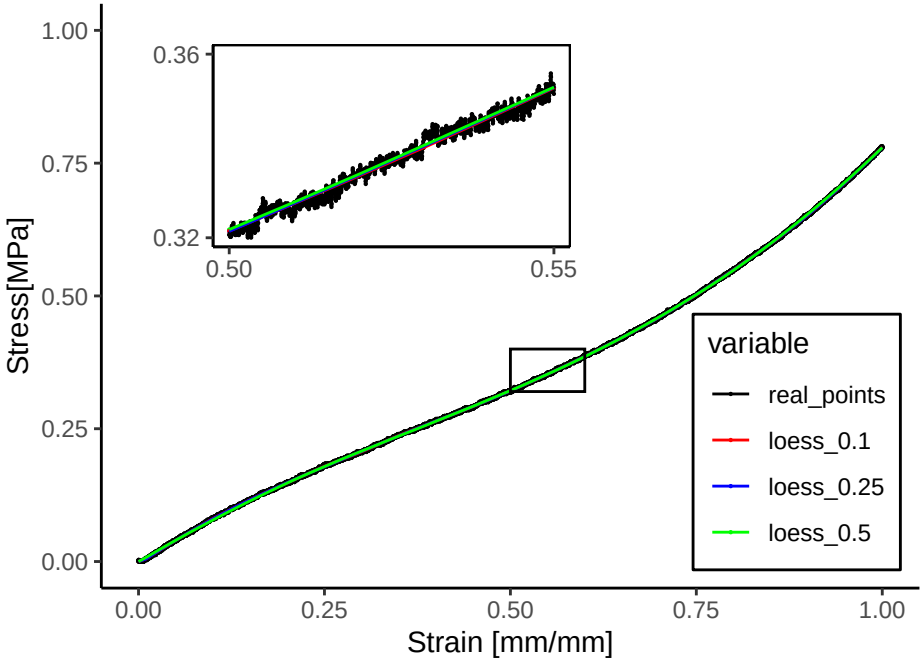


Figure 3.6: Stress-Strain curve data and fitting

3.3 Digital Image Correlation Tests

The Digital Image Correlation Tests (DIC tests) consist in two stages: Tensile test with image capture and Image treatment in a software.

The first stage were made also in Strenght of Materials and Structures Laboratory on IPB, on a Shimadzu Autograph AGS-X 10KN universal test machine, like the previous test. This test was set to pull the specimen until 20% of the original length, wait for ten (10) seconds and then return back to the original position, wait another ten (10) seconds an then initiate the process again. The parameters of the test such as cited before were defined by the author, shown in the Table 3.1. This first stage is shown in Figure 3.7.

Parameters	**Specimen 1 - Values**	**Specimen 2 - Values**	**Specimen 3 - Values**
Test velocity	5 mm/min	5 mm/min	5 mm/min
Max. strain	20%	20%	20%
Repetitions per specimen	2	3	3

Table 3.1: DIC tests parameters.

This stage of the test requires an acquisition image system, which consisted in a camera (a Canon EOS 7D, filming in 1080p HD, $1920 * 1088$ pixels) with proper supports to set the best position and a lighting system for a better caption. This lighting system, in a lack of a proper system, was made of MDF, projected by the author and produced in a laser cutting machine by the Fabrication Laboratory also in IPB. The support is shown in the Figure 3.8, and the project is in Appendix B. This support doesn't appear in the figure 3.7, because the support is behind the camera, at around 1 (one) meter of the specimen, but it is there to ensure the lightining to the photos, along the support shown in figure 3.7, all of them using LED lamps of 6 (six) Watts (3 lamps on total).

The camera filmed all the test and all the videos were used in the second stage.

The second stage were made in the GOM Correlate software, using the frames of the video previously captured. Each test had approximately 3000 (three thousand) frames of useful part, and the software can't evaluate such a massive frames. So, a routine in C++

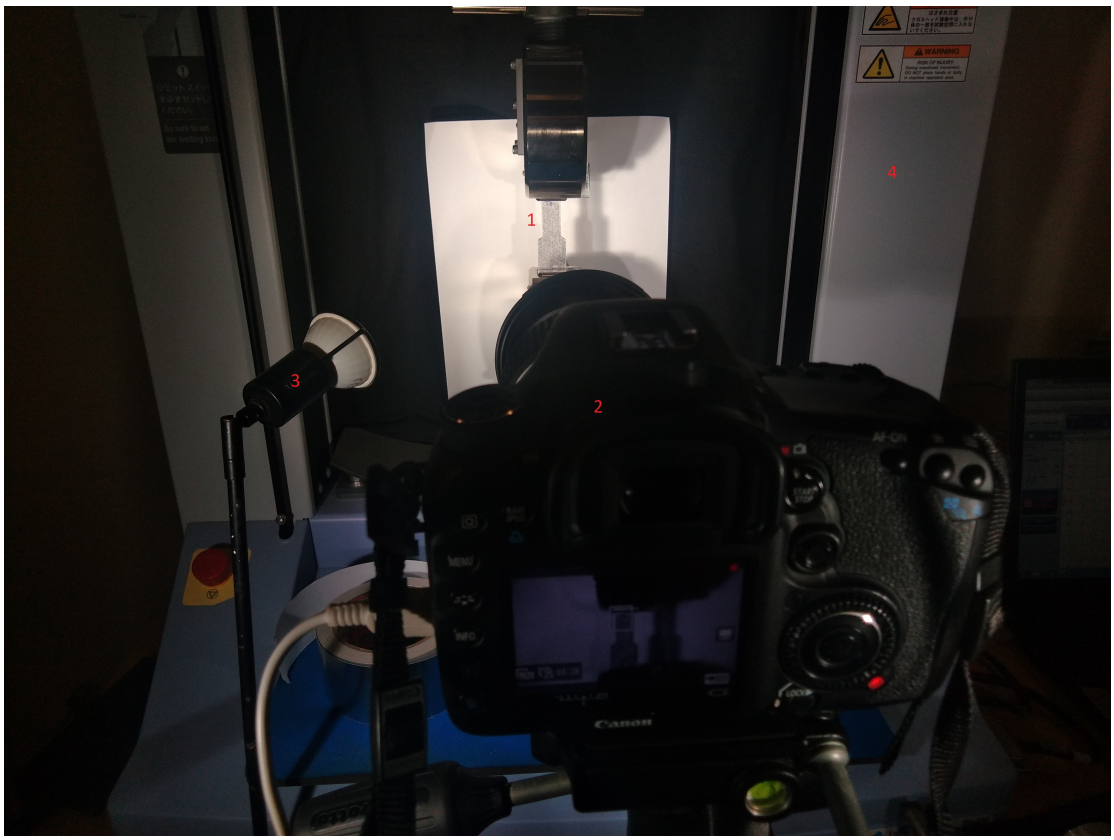


Figure 3.7: Specimen in the Tensile Test for image acquisition indicating: 1 - Specimen; 2 - Camera; 3 - Lightning System; 4 - Universal Test Machine.

was developed in order to delete 90% (ninety percent) of the frames. The logic is: Starting from the first image, at each 9 (nine) images, the 10^{th} (tenth) would be maintained and the previous nine would be erased. This procedure would be repeated until the end of the sample. So, after this routine, each test had approximately 300 (three hundred) frames now for each test, and the software now was able to perform the test. The software only read the images if the frames have names that follows a numeric order, according the real order of images (for example: "frame_1.jpg", "frame_2.jpg", "frame_3.jpg",...) so the routine in C++ also included a way to rename the images to have the proper names. The C++ code is in the Appendix C.

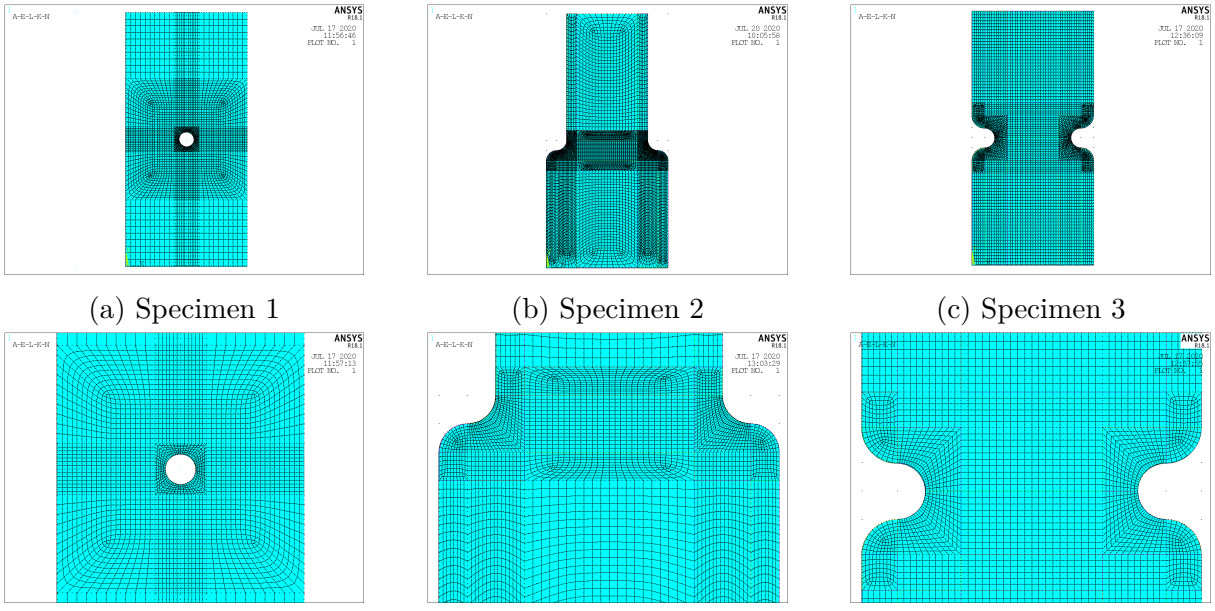
After the frame operation above, it was possible to perform the analysis in the GOM Correlate. Each repetition of all the tests were analized on GOM Correlate, which gives an amount of 8 (eight) DIC analysis. For all these this analysis, the definition of some

Figure 3.8: Light Support

parameters were necessary. First of all, the calibration of the images, where a region with a known spatial measure were defined (for example, the width of the specimens, with $24mm$), then the region of the test (DIC 1 sample 1 - facet size 19 pixels, point distance 16 pixels, DIC 1 sample 2 - facet size 19 pixels, point distance 16 pixels; DIC 2 sample 1 - facet size 19 pixels, point distance 16 pixels, DIC 2 sample 2 - facet size 23 pixels, point distance 19 pixels, DIC 2 sample 3 - facet size 23 pixels, point distance 19 pixels; DIC 3 sample 1 - facet size 23 pixels, point distance 19 pixels, DIC 3 sample 2 - facet size 23 pixels, point distance 19 pixels, DIC 3 sample 3 - facet size 23 pixels, point distance 19 pixels), then the axle alignment (only necessary if the camera would not be vertically aligned), then the inspection points for comparison (with the position defined by the author) and then the total strain in y direction of these points. After this analysis been made for each test, it is possible to export the deformation of the selected points for a comparison analysis.

3.4 Numerical Simulation

The three different specimen were simulated numerically in order to compare the numeric results with the DIC results of strain. All the three specimen were made in ANSYS Mechanical APDL using three different constitutive models for each specimen and the obtained curve in the standard tensile test. The three used models were the 1-parameter Ogden model, the Neo-Hookean model, and the Mooney-Rivlyn 3-parameters model. For all the tests, only the useful region of material were projected in ANSYS (The region off the claw) and the PLANE183 element was chosen, all with 8 nodes. For this element, once the constants of the models were calculated, there is no need to specify any other material property in the model, because all the information is inside the constitutive model.



(a) Specimen 1 (b) Specimen 2 (c) Specimen 3

(d) Specimen 1 zoomed (e) Specimen 2 zoomed (f) Specimen 3 zoomed

Figure 3.9: Mesh of the specimen simulations.

The mesh were developed and are shown in Figure 3.9, in a way the precision in specific zones were higher and the change between the mesh regions were smooth. The final amount of nodes and elements for each specimen is shown in table 3.2. The boundary conditions were applied considering the nodes on bottom did not have any movement on y, the odes on bottom had a displacement of $10mm$ on $y+$ (20% of the total height) and

both nodes did not have any deformation on x. All the simulation was calculated with 5000 substeps, considering as maximum 10000 substeps and minimum as 1000 substeps. After the solution is done, the total strain in y direction of the nodes and elements of the specimen can be exported. For the comparison, it's necessary to have the coordinates of the nodes, the nodes of the elements, and also the strain of the nodes and elements, so, all these information were exported to a ".xls" or ".csv" file, with an Excel help, and then, the comparison between the simulation and the can be made externally.

Mesh Properties	**Specimen 1**	**Specimen 2**	**Specimen 3**
Nodes	10240	16577	16533
Elements	3336	5434	5388

Table 3.2: Amount of elements and nodes of the simulations.

3.5 Comparison

The main goal of this stage is to compare the numeric strains with the DIC strains, and see if there is some significant difference between the constitutive models in this analysis, for each specimen. The comparison consisted in calculate the error between the ANSYS simulation strain ϵ_{Sim} with the DIC simulation points strain ϵ_{DIC} as shown in equation 3.1, and evaluate the errors distribution.

$$E = \frac{|\epsilon_{DIC} - \epsilon_{Sim}|}{\epsilon_{DIC}} * 100\% \tag{3.1}$$

Two approach were made: In the first, the comparison were between the point strain and the strain of the centroid of the element where the point is inscribed. In the second, the comparison were between the point strain and the strain of the closest node of the simulation. In general, the second approach presented lower errors and deviations between them.

For this comparison, some things are necessary:

- To know the strain and the coordinates of all nodes;

- To know the strain of all elements;
- To know the which node belongs to which element.

This way is possible to know the position and the strain of the elements and nodes, and then evaluate the nearest nodes of the defined points an the elements that the defined points are inscribed.

All the comparison analysis were made in R programming language, using the RStudio, a programming interface applied on statistics and data analysis. There were made three R codes, for the three different specimen, which are essentially the same, with the only difference that uses different databases. These codes are divided in four parts, as follows:

The first part consisted in operate the database of the nodes coordinates, elements coordinates, elements nodes, nodes strain and elements strain, in order to know the exact initial position and the final strain of the points and elements of the simulations, for each constitutive model.

The second part verifies the circumscribed elements of the defined points, for each constitutive model.

The third part adjusts the DIC data, calculating the mean final strain of the points and the nearest nodes of them, also prepares the data for the next analysis, for each constitutive model, and also.

The fourth and last part plot the results for each model and evaluates statistically the difference between the constitutive models.

All the codes are in the Appendix D.

At the end, an analysis of the stress concentrators with a reference simulation was done, and then the evaluation of the stress concentration factor for this scenario.

Chapter 4

Results

The strain results obtained by DIC and ANSYS analysis for all the tests are shown in figures 4.1, 4.4 and 4.7. For each figure, it is possible to see the stress concentrator effects on the samples, and how they increase the strain around them. Also, the shape of the tension contour on both simulation and DIC test seems similar, which by itself shows some similarity between the test and the simulation. It is possible to see also that there are differences between the simulations of same specimen, due to the fact each simulation has a different constitutive model.

There are also some differences between the DIC analysis, due the "noise" presented by the images. Each point presented respectively in the figures 4.2, 4.5 and 4.8 have its y strain presented respectively in the figures 4.3, 4.6 and 4.9. All this figures present an oscillation of the points even when the specimen was stopped, which means there is a "noise" in the image, some changes in the images that changes the real results of the strain of each point.

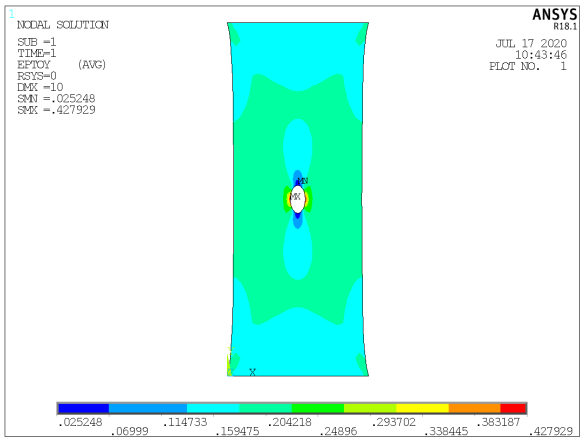
That all been said, it is necessary to see if there is any differences between the DIC tests and the ANSYS simulations. For that, the mean of the last ten evaluations of each point (in red, on figures 4.3, 4.6 and 4.9) was used for the comparison with the ANSYS simulations of the specimens, which is presented in the next chapter.

For the evaluation of the stress concentrators, it is necessary to do a comparison with a simulation of a part without stress concentrators. The simulation of this one is shown

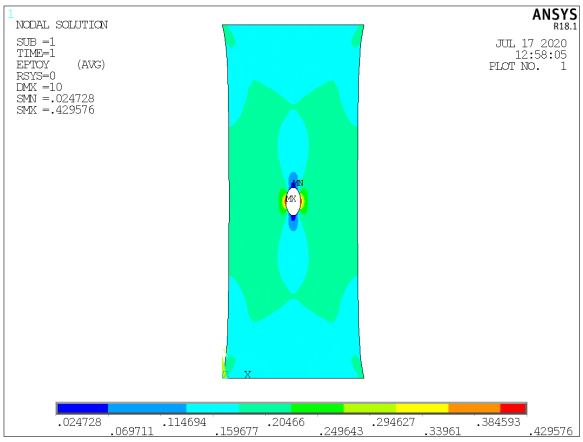
in figure 4.10. In this figure is possible to understand that the stress in the center of the figure is almost constant, and this stress is the one to be consides as reference in the calculation of the stress concentration factor. All the stress distribution in the center of all the samples provided also by ANSYS data using the ogden model, is shown in figure 4.11, and with that, evaluate the stress concentration in this specimen.

(a) First DIC Sample

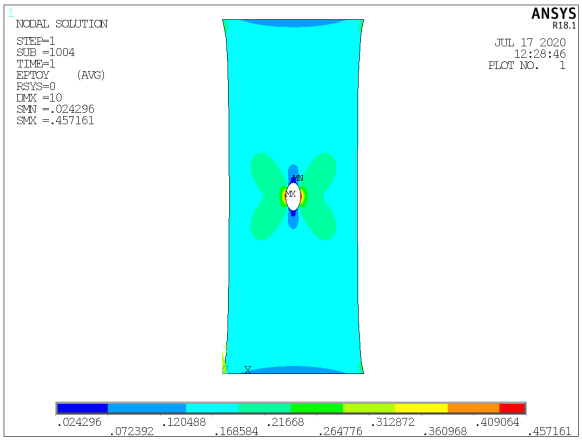
(b) Second DIC Sample



(c) Mooney-Rivlyn Model



(d) Ogden Model



(e) Neo-Hookean Model

Figure 4.1: DIC and Simulation Results for Specimen 1

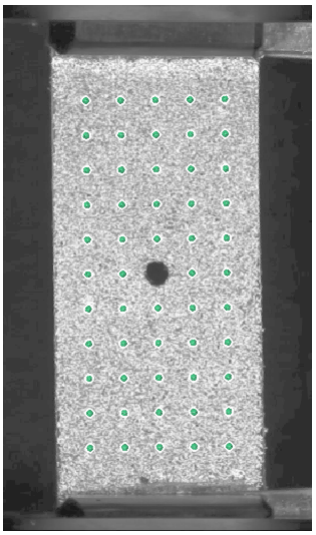
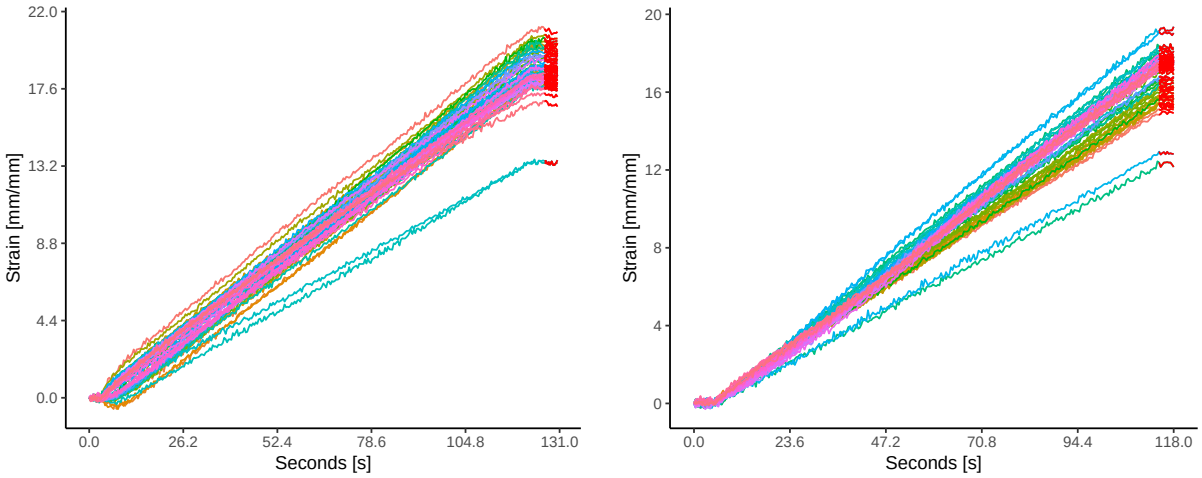


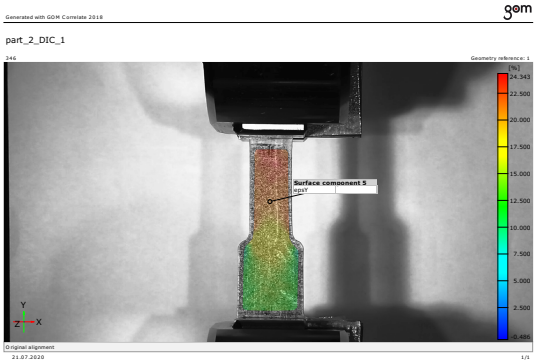
Figure 4.2: DIC Predefined Points of Specimen 1



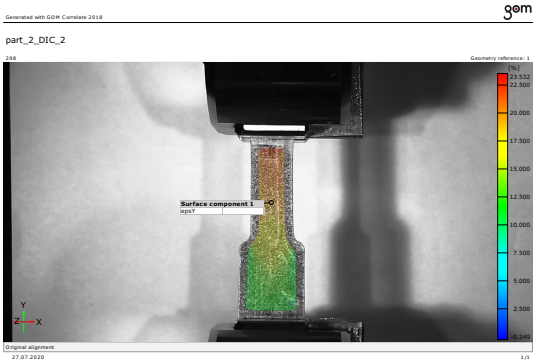
(a) First Sample

(b) Second Sample

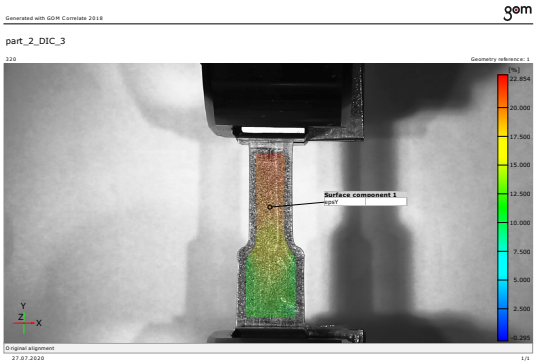
Figure 4.3: DIC Strain of Points for Specimen 1



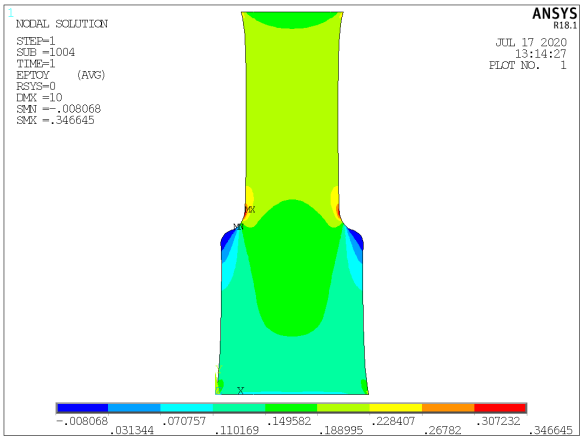
(a) First DIC Sample



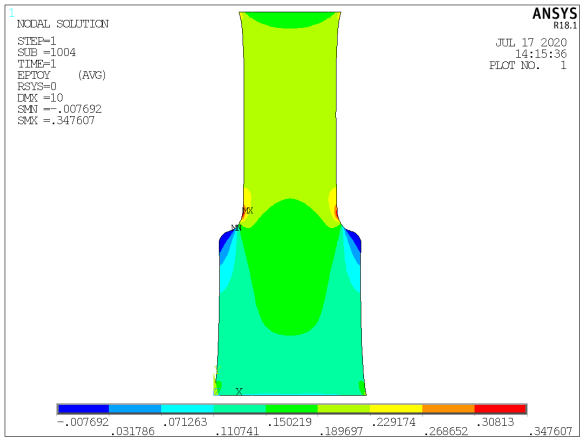
(b) Second DIC Sample



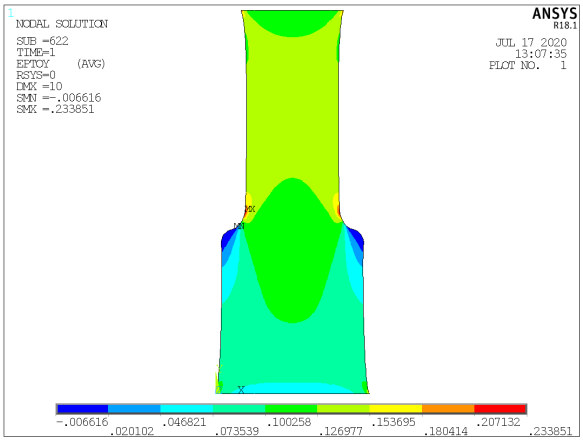
(c) Third DIC Sample



(d) Mooney-Rivlyn Model



(e) Ogden Model



(f) Neo-Hookean Model

Figure 4.4: DIC and Simulation Results for Specimen 2

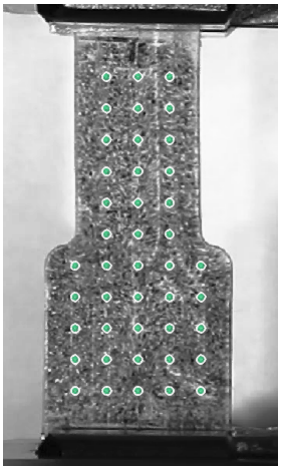
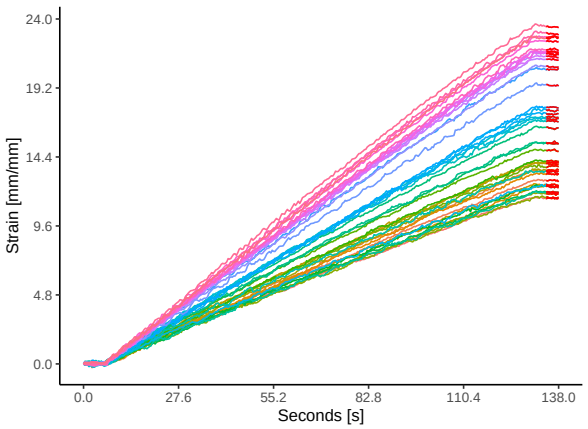
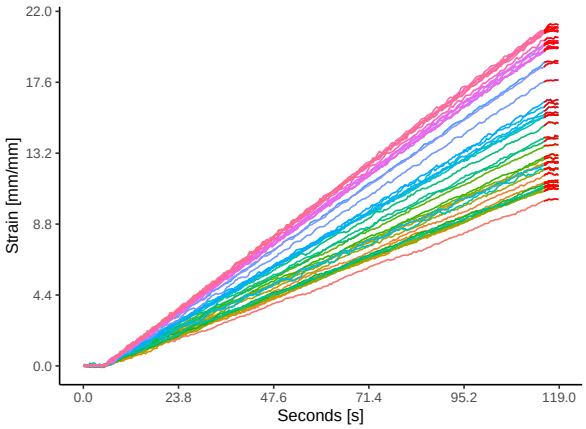


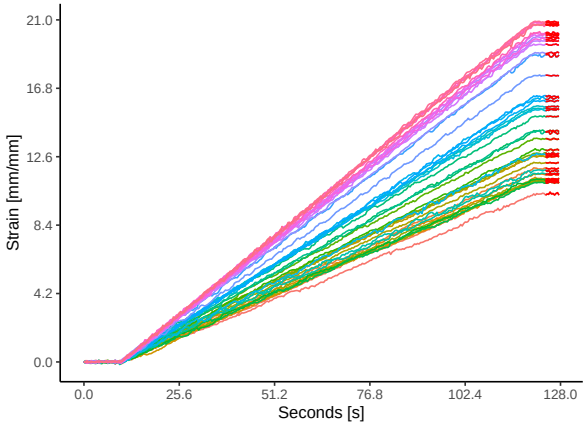
Figure 4.5: DIC Predefined Points of Specimen 2



(a) First Sample



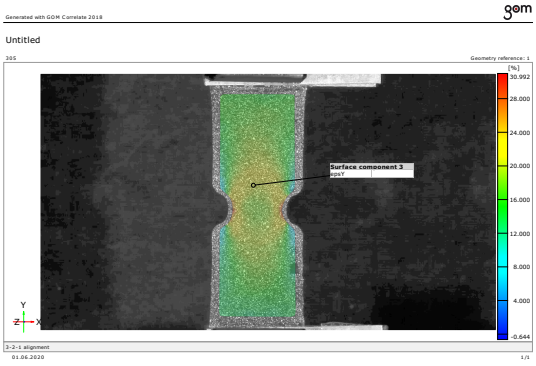
(b) Second Sample



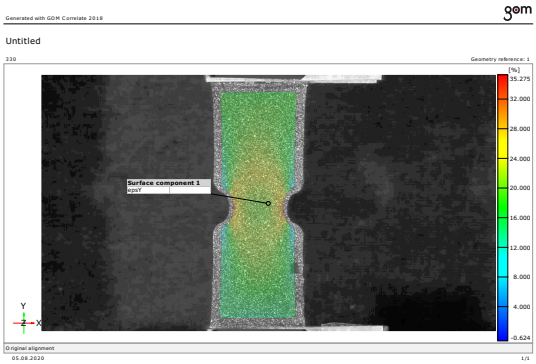
(c) Third Sample

Figure 4.6: DIC Strain of Points for Specimen 2

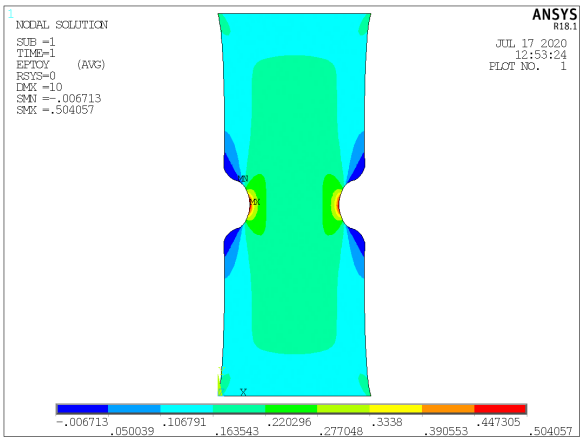
(a) First DIC Sample



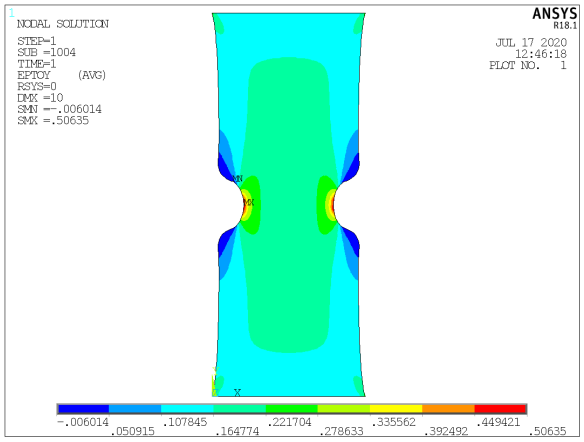
(b) Second DIC Sample



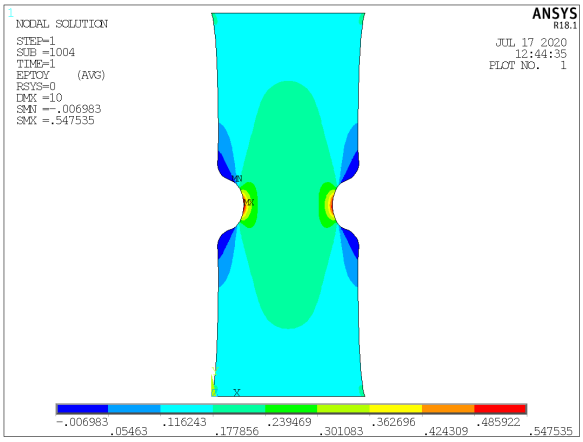
(c) Third DIC Sample



(d) Mooney-Rivlyn Model



(e) Ogden Model



(f) Neo-Hookean Model

Figure 4.7: DIC and Simulation Results for Specimen 3

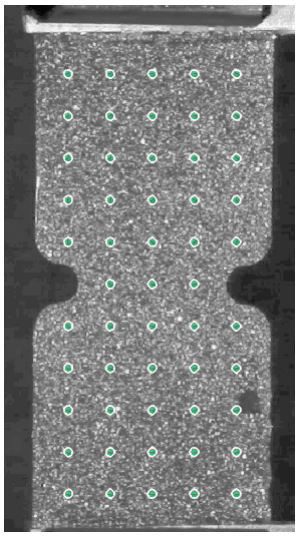
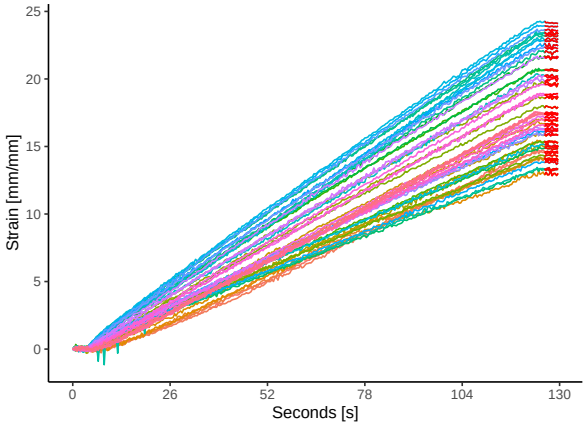
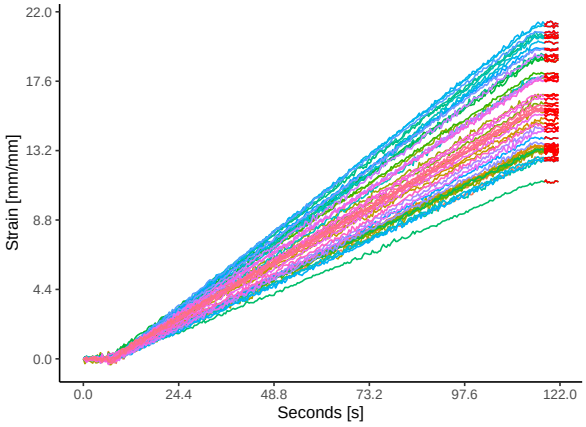


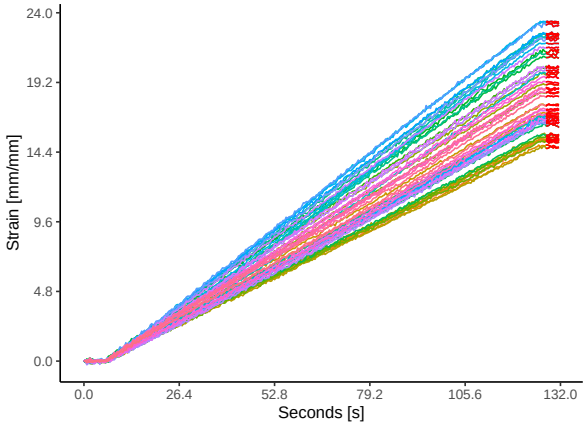
Figure 4.8: DIC Predefined Points of Specimen 3



(a) First Sample



(b) Second Sample



(c) Third Sample

Figure 4.9: DIC Strain of Points for Specimen 3

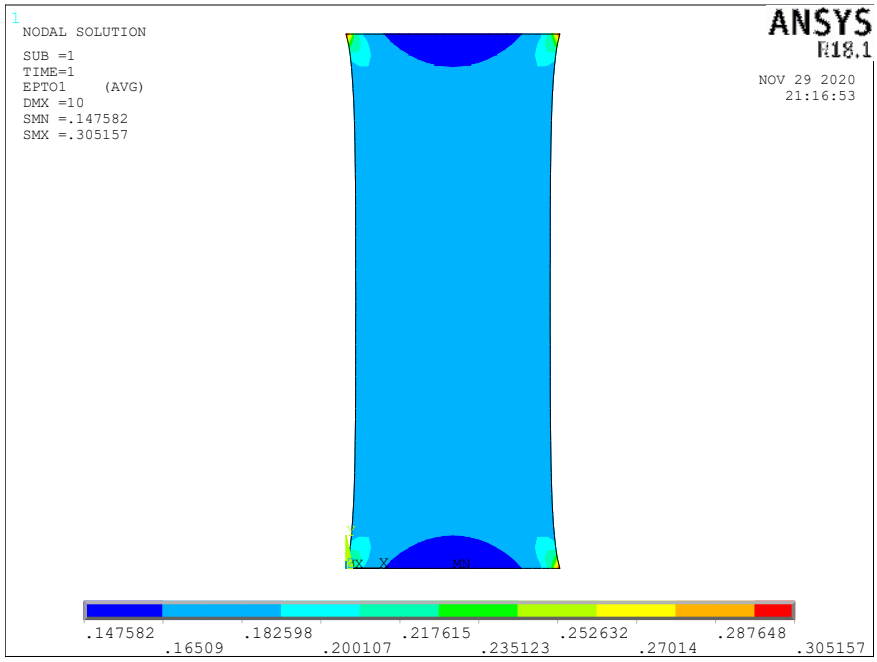


Figure 4.10: ANSYS Stress of the Reference Specimen.

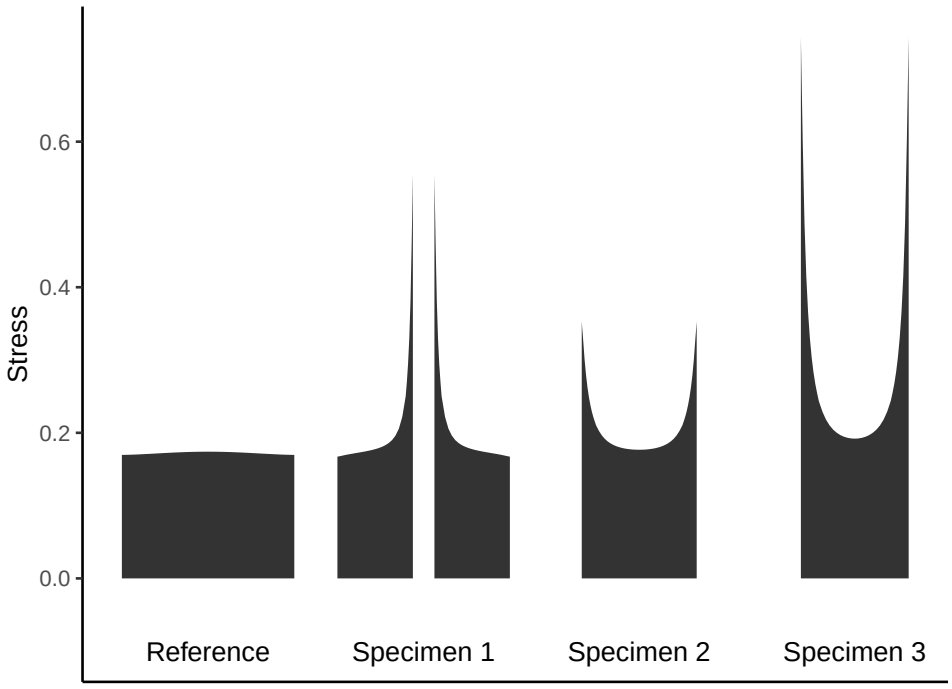


Figure 4.11: ANSYS Stresses of the center of the specimens.

Chapter 5

Discussion

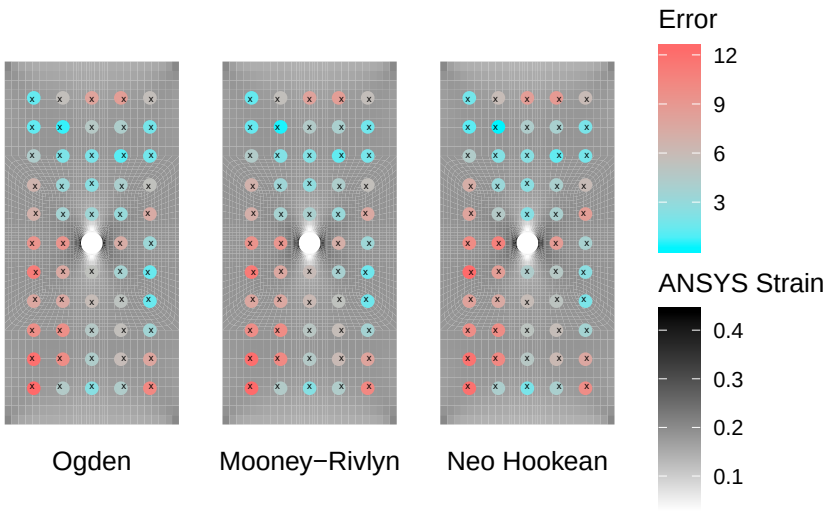
All the comparison were made through the predefined points in the specimens, shown previously in figures 4.2, 4.5 and 4.8. As said previously, the mean of the last ten strain evaluations of this point were compared with the strain evaluation of the nearest nodes of the simulations in ANSYS, and also with the strain evaluation of the circumscribed element of each point.

5.1 Specimen 1 - Center Hole

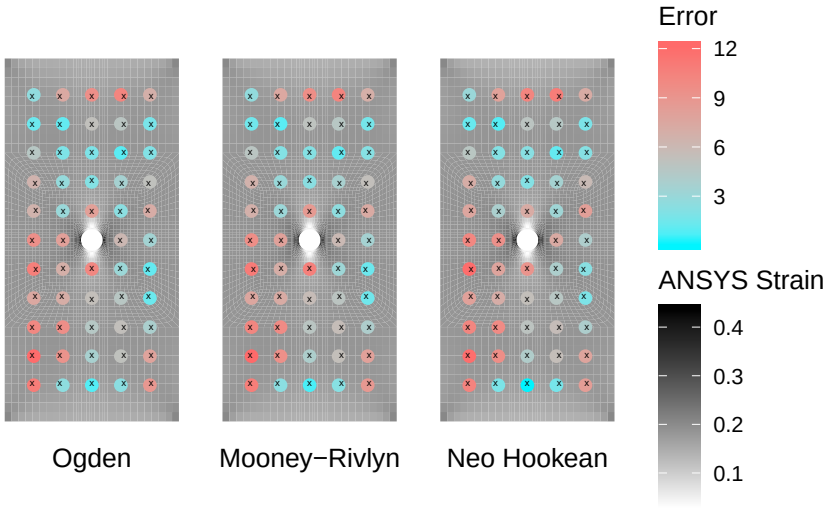
The figure 5.1 shows the strain for each element in ANSYS (in a gray-scale), the error of the selected points (as a blue-red scale) and the position of the nearest point (as an "x") for both comparison methods. The values are in Appendix D. In figure 5.1a the error has the same behavior for all the three constitutive models, similar to the figure 5.1b. The both comparison has the same behavior also between them. Also, the error distribution values seems random, although there are a concentration of the higher values on the bottom-left region of the sample, which can indicate some deviation in the DIC evaluation.

To evaluate the error distribution, a statistical analysis were performed, also in R. The box plots of the errors are shown in figure 5.2, and it can be seen that the errors distribution of all constitutive models are similar for both comparison, and also between the comparison, with an error around 5%. to prove this fact, it is possible to do the

ANOVA test (Analysis of variance test) to see if there is any significant difference. This test is shown in figure 5.3, and also the tukey HSD that allows a visual approach to the comparison. The p-value for both comparison method are above 0.05 (5% confidence interval), which means there are not significant differences between the constitutive models O, M, and N (Ogden, Mooney-Rivlyn and Neo Hookean models, respectively) neither for the nearest nodes approach nor for the circumscribed elements approach. In the Tukey's plot, if the lines of the comparison intervals cross the zero (dashed vertical line), there are no significant difference between these elements, and in the graphs, all the interval lines cross the dashed line.

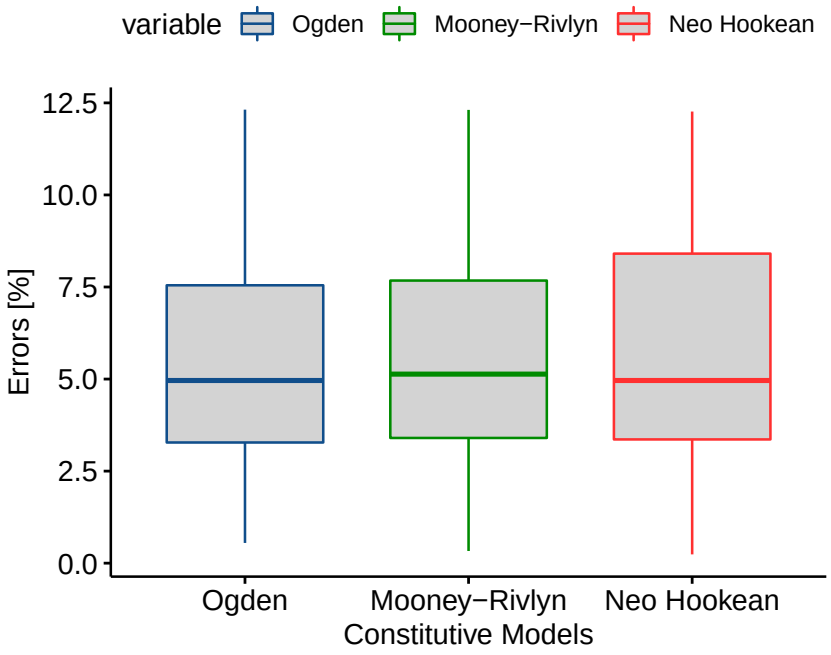


(a) Nearest Nodes Comparison

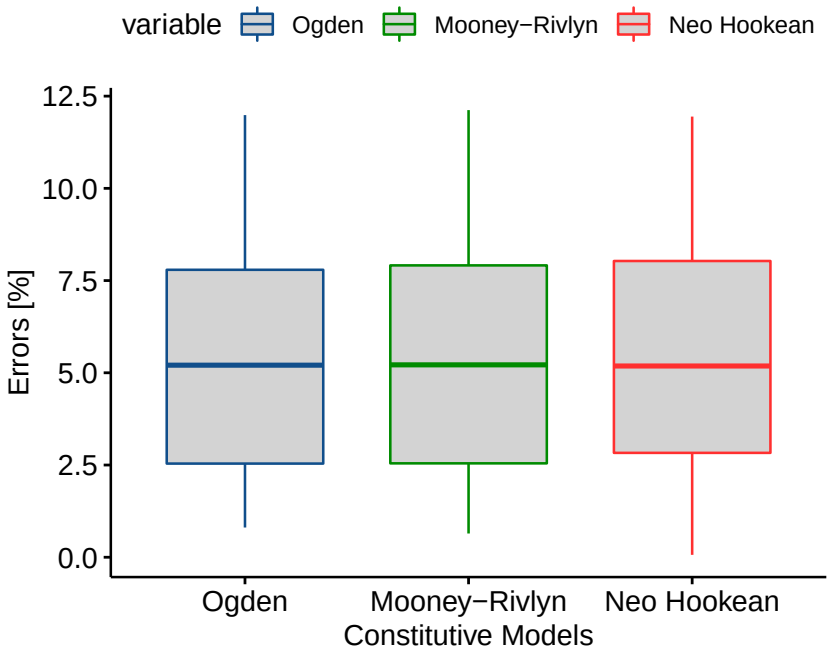


(b) Circunscribed Element Comparison

Figure 5.1: Comparison Between ANSYS simulation and DIC tests for Specimen 1



(a) Nearest Nodes Comparison

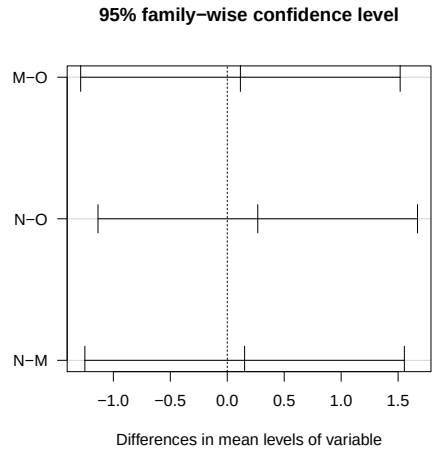


(b) Circumscribe Element Comparison

Figure 5.2: Box Plot of Errors For Each Comparison Method on Specimen 1

```
> summary(aov_node)
             Df    Sum Sq  Mean Sq F value  Pr(>F)
variable      2    1.9477 0.973855 0.10262 0.90253
Residuals   159 1508.8291 9.489491
```

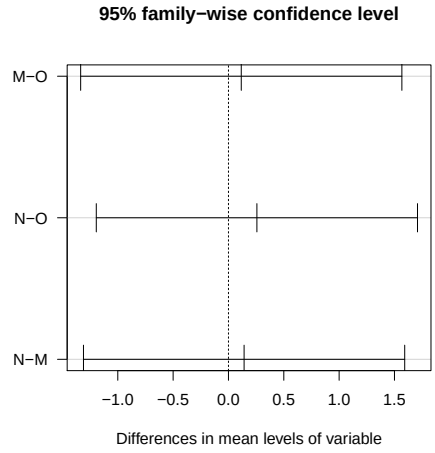
(a) Nearest Nodes ANOVA Test



(b) Nearest Nodes Tukey Plot

```
> summary(aov_elem)
             Df    Sum Sq   Mean Sq F value Pr(>F)
variable      2    1.7881  0.894035 0.08801 0.9158
Residuals   159 1615.2198 10.158615
```

(c) Circumscribe Element ANOVA Test



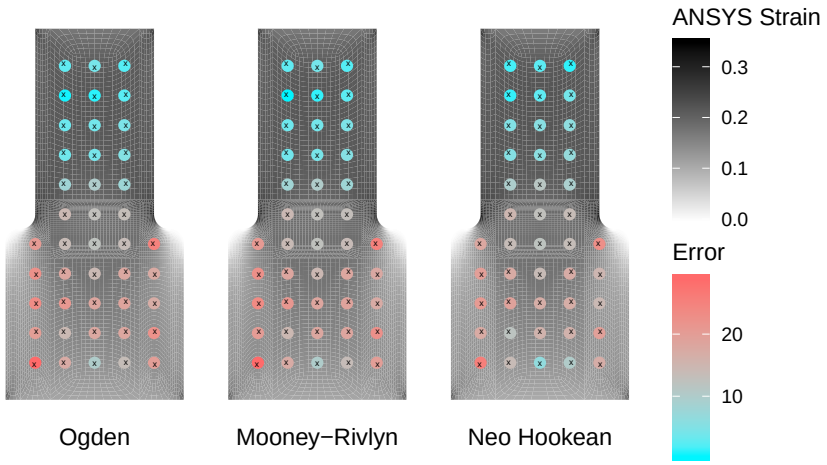
(d) Circumscribe Element Tukey Plot

Figure 5.3: ANOVA Test and TUKEY HSD plot For Specimen 1, For Each Comparison Method

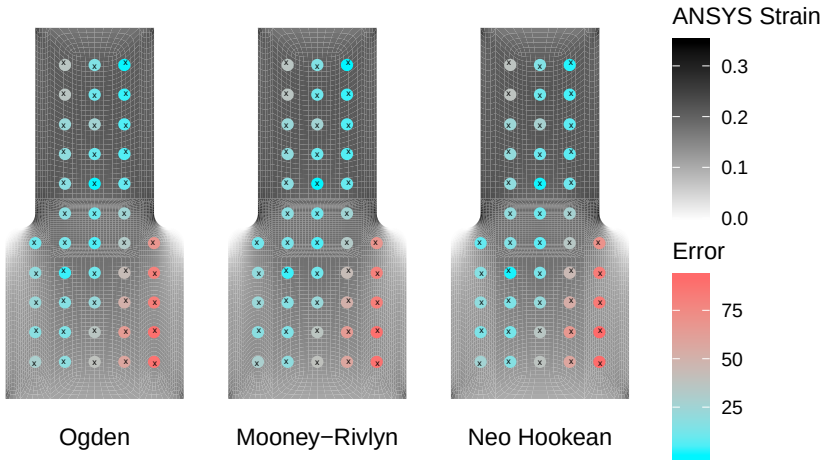
5.2 Specimen 2 - Symmetric Circular Shoulder Fillets

The figure 5.4 shows the strain for each element in ANSYS (in a gray-scale), the error of the selected points (as a blue-red scale) and the position of the nearest point (as an "x") for both comparison methods, just like the previous one. The values are also in Appendix D. In figure 5.4a, the error has the same behavior for all the constitutive models, similar to the figure 5.4b. But here the comparison have a very different behavior between themselves. The error distribution values seems random only for the nearest nodes comparison (although there are a grouping of higher errors in the bottom of the sample), while the circumscribed element comparison presents the most part of the error in blue (around 20% error) and some values in red (near 80%), with the higher errors in the bottom-right region of the sample.

Similar to the previous specimen, a statistical analysis were performed to evaluate the errors distribution. The box plots of the errors are shown in figure 5.5, where it is possible to see that the errors distribution for each comparison method are similar between the constitutive models, but the circumscribed element comparison has higher values than the nearest nodes comparison. An ANOVA test (Analysis of variance test) was made to prove there is any significant difference between the constitutive models. This test is shown in figure 5.6, and also the tukey HSD, that allows a visual approach to the comparison. Like the previous one, The p-value for both comparison method are above 0.05 (5% confidence interval), which means there are not significant differences between the constitutive models O, M, and N (Ogden, Mooney-Rivlyn and Neo Hookean models, respectively) neither for the nearest nodes approach nor for the circumscribed elements approach. The Tukey's plot just confirms that, once the lines of the comparison intervals crosses the zero (dashed vertical line), which means there are no significant difference between these elements.

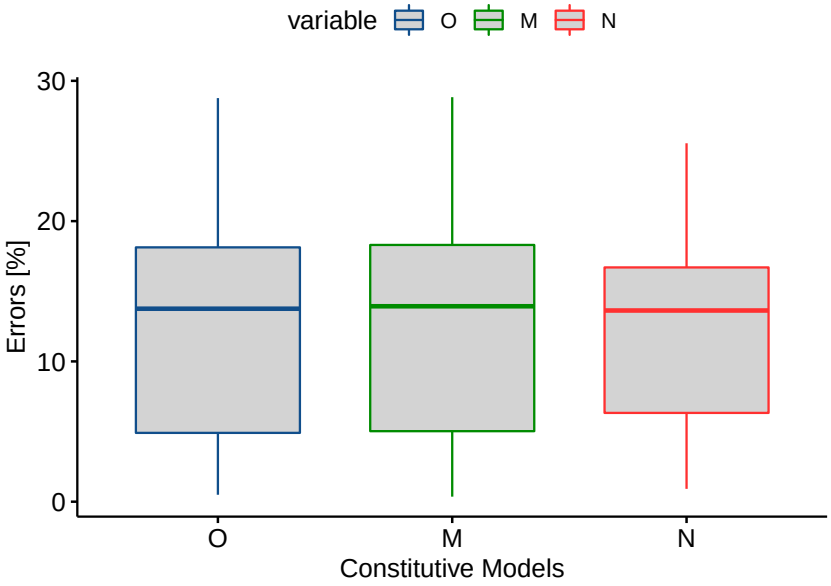


(a) Nearest Nodes Comparison

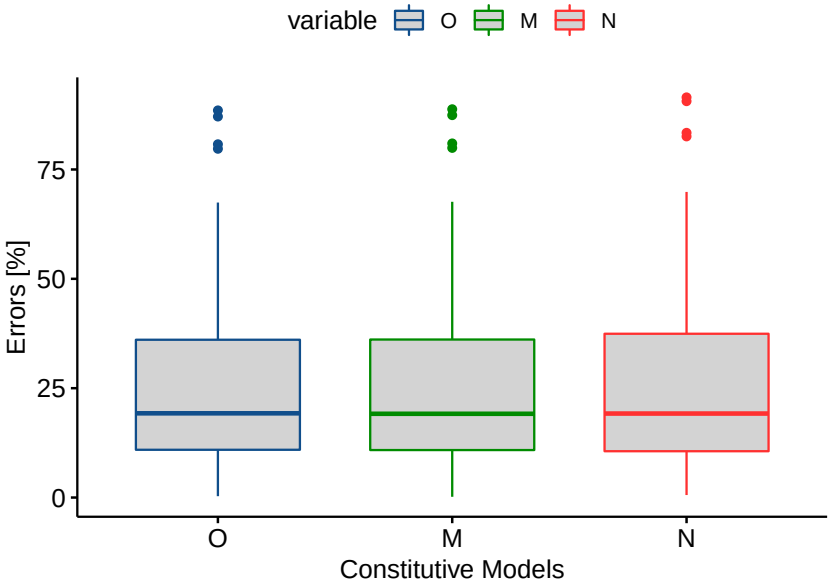


(b) Circunscribed Element Comparison

Figure 5.4: Comparison Between ANSYS simulation and DIC tests for Specimen 2



(a) Nearest Nodes Comparison

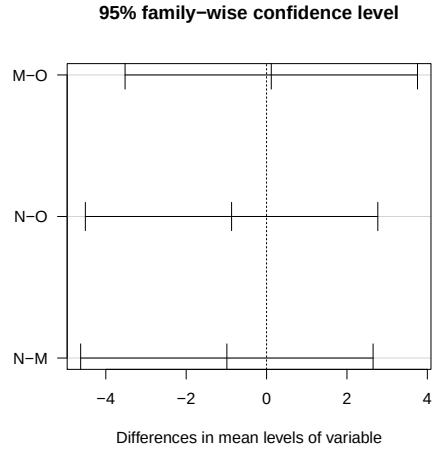


(b) Circumscribe Element Comparison

Figure 5.5: Box Plot of Errors For Each Comparison Method on Specimen 2

```
> summary(aov_node)
             Df   Sum Sq  Mean Sq F value  Pr(>F)
variable      2   25.042 12.52088 0.24755 0.78109
Residuals   126 6373.005 50.57940
```

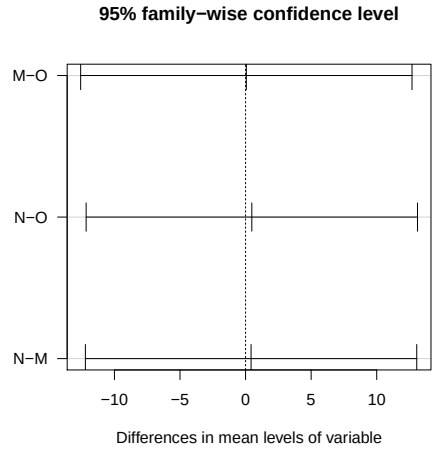
(a) Nearest Nodes ANOVA Test



(b) Nearest Nodes Tukey Plot

```
> summary(aov_elem)
             Df   Sum Sq  Mean Sq F value  Pr(>F)
variable      2     5.82   2.9091 0.00477 0.99524
Residuals   126 76870.82 610.0859
```

(c) Circumscribe Element ANOVA Test



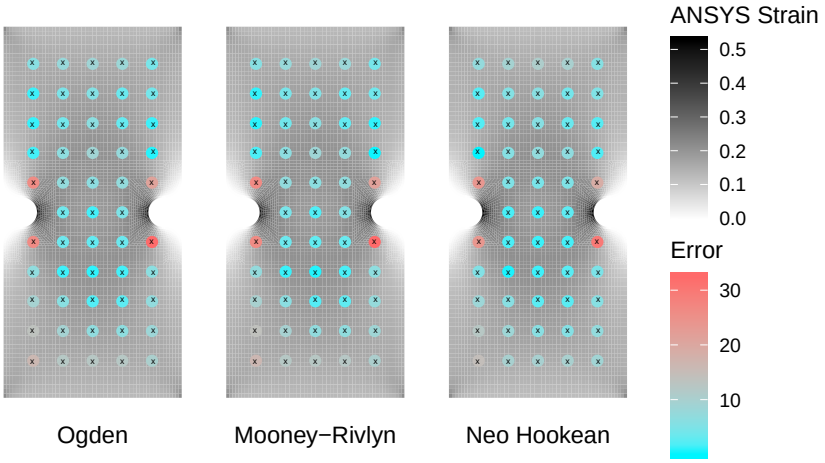
(d) Circumscribe Element Tukey Plot

Figure 5.6: ANOVA Test and TUKEY HSD plot For Specimen 2, For Each Comparison Method

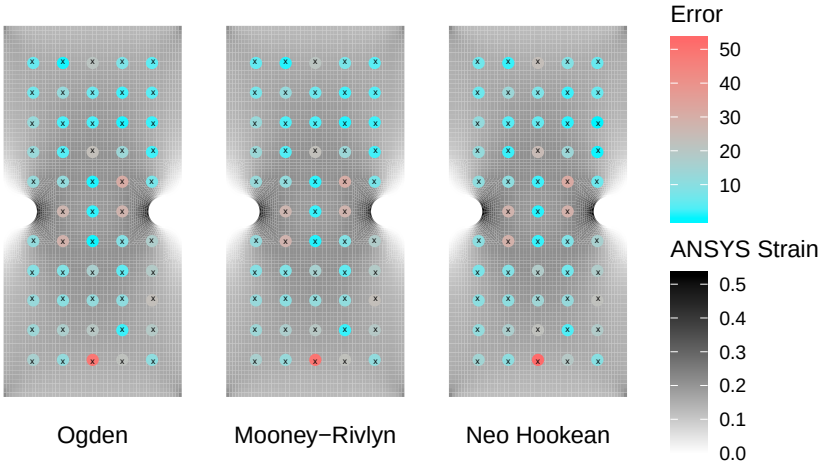
5.3 Specimen 3 - Symmetric Semicircular Edge Nodes

The figure 5.7 shows the strain for each element in ANSYS (in a gray-scale), the error of the selected points (as a blue-red scale) and the position of the nearest point (as an "x") for both comparison methods, just like in the previous specimens. Also, the values are in Appendix D. In figure 5.7a, the error has the same behavior for all the constitutive models, similar to the figure 5.7b. In both comparison the errors looks similar to all points, with some few exceptions. The error distribution on the first comparison presents lower extreme errors than the second, but it is hard to see anything just looking to the values.

Similar to the previous specimens, a statistical analysis were performed to evaluate the errors distribution. The box plots of the errors are shown in figure 5.8, and now it is possible to see the errors of the first comparison are lower than the second, and similar to the previous analysis, the errors distributions of each comparison method looks similar. To prove that, an ANOVA test (Analysis of variance test) was made, along with a tukey HSD test. This test is shown in figure 5.9. Like in the previous specimens, The p-value for both comparison method are above 0.05 (5% confidence interval), showing there are not significant difference between the constitutive models O, M, and N (Ogden, Mooney-Rivlyn and Neo Hookean models, respectively) neither for the nearest nodes approach nor for the circumscribed elements approach. The Tukey's plot just confirms that, once the lines of the comparison intervals crosses the zero (dashed vertical line), showing again there are no significant difference between these elements.

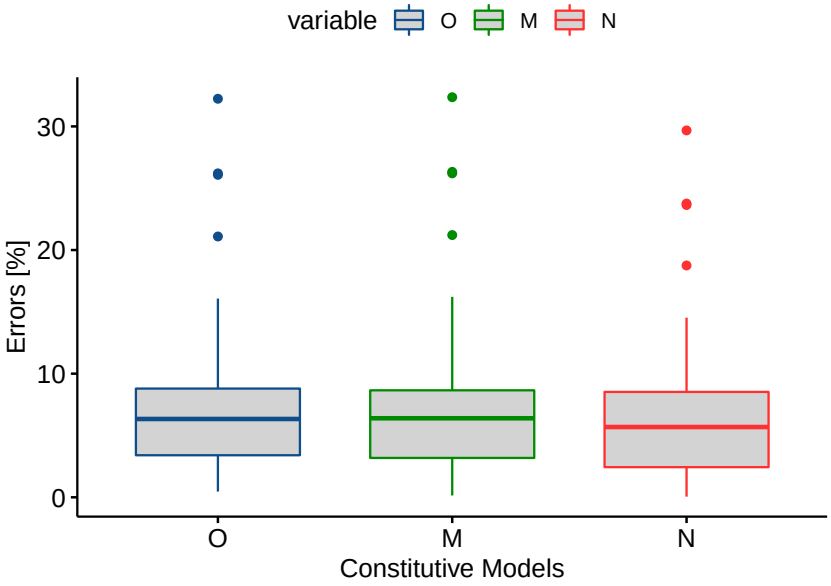


(a) Nearest Nodes Comparison

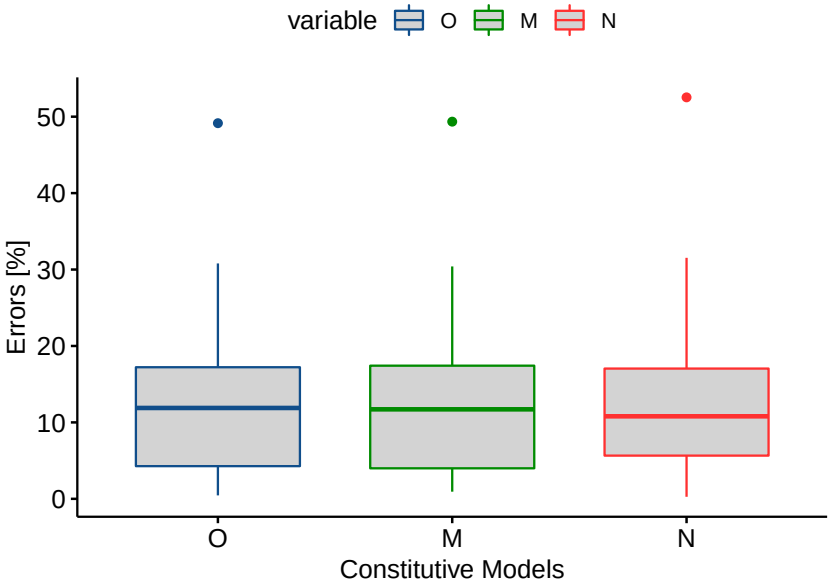


(b) Circumscribed Element Comparison

Figure 5.7: Comparison Between ANSYS simulation and DIC tests for Specimen 3



(a) Nearest Nodes Comparison

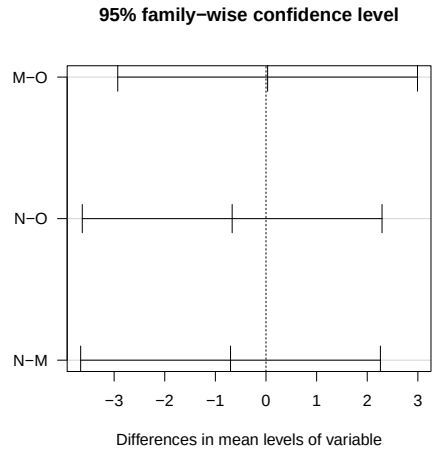


(b) Circumscribe Element Comparison

Figure 5.8: Box Plot of Errors For Each Comparison Method on Specimen 3

```
> summary(aov_node)
             Df   Sum Sq  Mean Sq F value  Pr(>F)
variable      2   16.585  8.29243 0.19989 0.81903
Residuals   156 6471.766 41.48568
```

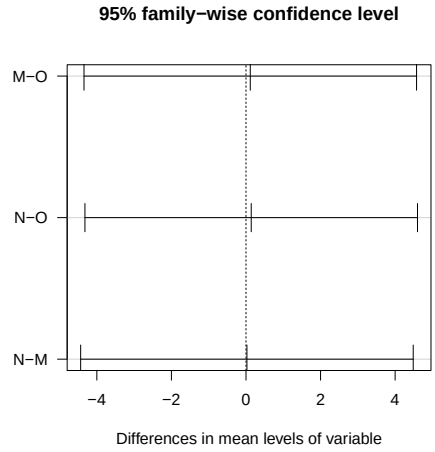
(a) Nearest Nodes ANOVA Test



(b) Nearest Nodes Tukey Plot

```
> summary(aov_elem)
             Df    Sum Sq  Mean Sq F value  Pr(>F)
variable      2     0.595  0.29749 0.00316 0.99684
Residuals   156 14683.288 94.12364
```

(c) Circumscribe Element ANOVA Test



(d) Circumscribe Element Tukey Plot

Figure 5.9: ANOVA Test and TUKEY HSD plot For Specimen 3, For Each Comparison Method

5.4 Stress Concentrator Comparison

With all the results obtained, with also the stress obtained for a specific region of the specimen (the center of all of them), it is possible to evaluate the stress concentration factor (K) for each specimen. This one is obtained by Equation 2.10, using as the numerator the higher stress values of each specimen and the mean of the reference stress in the denominator. This result is shown in Table 5.1.

Specimen 1	**Specimen 2**	**Specimen 3**
3.229	2.051	4.323

Table 5.1: Stress Concentration Factors for each Specimen

The biggest K is of the symmetric semicircular edge nodes specimen, that also have the lowest length of the section, so is understandable. But the lowest K belongs to the symmetric shoulder fillets specimen, that does not have the lowest length of the section. As a comparison, One can see the K for a steel plate, that can be calulated according the equation 2.11 and the figures 2.5 and 2.7, and is presented below.

Specimen 1	**Specimen 2**	**Specimen 3**
2.670	2.1	2.400

Table 5.2: Stress Concentration Factors for a Steel Plate

It is possible to see the results are very different, with exception of the second specimen, that have similar results.

Chapter 6

Conclusion and Proposal for Future Work

The first goal was to evaluate which constitutive model is better to evaluate parts with stress concentrators. This goal was achieved by comparing the values of the ANOVA table. In general, for the conditions presented on this work, there was no significant difference between the constitutive models, which implies that for this conditions, any model can be chosen. This can be reviewed in future work with a better images that allows to have lower errors.

The higher errors were presented in the second sample (symmetric shoulder fillets). This is due to the fact this shape generate some deformations when the specimen were pressed by the claws of the tensile machine.

The fact that there was a similar behavior of the constitutive model for each simulation regarding the position of the errors (for example, in the second sample, where all the constitutive models presented higher errors in the bottom of the figure on the nearest nodes comparison, and in the first example, where all the constitutive models showed higher errors in the left of the sample) indicates that the error can be in the DIC strain evaluations.

It is possible to see the errors don't behave as expecte, with a bilateral simetry for

example, and this is due the fact there are a lot of significative error associated to experimental tests. The material is very soft and doesn't present any resistance to the claws, so it is difficult to ensure the perfect alignment of the specimen on the tensile machine.

The second goal was to evaluate the stress concentrator itself. This was achieved by comparing the simulations with stress concentrators with a specimen without any stress concentration, in order to evaluate the Stress Concentration Factor K. Due to the fact there is no preferable model, the last comparison that involves the stress concentrators specifically uses only the Ogden model for the simulations. The K results for the specimen 1 and 3 differs from the known behavior of the steel, which is expected, once the strain is not linearly related to the stress. That been said, a possible future work can be the evaluation and comparison of these models under biaxial or shear stress, with the variation of some parameters included (more than one total strain, different geometric values for the stress concentrators), the construction of K curves or equations that describes the behavior of these concentrators.

The region of the stress concentrator itself can be more studied, to investigate the region of the stress concentrator and its comparison with the real behavior, as a future work.

Bibliography

[1] M. P. Wolf, G. B. Salieb-Beugelaar, and P. Hunziker, "Pdms with designer functionalities—properties, modifications strategies, and applications", *Progress in Polymer Science*, vol. 83, pp. 97–134, 2018, ISSN: 0079-6700. DOI: https://doi.org/10.1016/j.progpolymsci.2018.06.001.

[2] G. Holzapfel, *Nonlinear solid mechanics: A continuum approach for engineering*. Wiley, 2000, ISBN: 9780471823193.

[3] L. Malvern, *Introduction to the mechanics of a continuous medium*, ser. Prentice-Hall series in engineering of the physical sciences. Prentice-Hall, 1969.

[4] J. Bergstrom, *Mechanics of solid polymers: Theory and computational modeling*, ser. Plastics Design Library. Elsevier Science, 2015, ISBN: 9780323322966.

[5] M. Sasso, G. Palmieri, G. Chiappini, and D. Amodio, "Characterization of hyperelastic rubber-like materials by biaxial and uniaxial stretching tests based on optical methods", *Polymer Testing*, vol. 27, no. 8, pp. 995–1004, 2008, ISSN: 0142-9418. DOI: https://doi.org/10.1016/j.polymertesting.2008.09.001.

[6] R. Ogden, G. Saccomandi, and I. Sgura, "Fitting hyperelastic models to experimental data", *Computational Mechanics*, vol. 34, Nov. 2004. DOI: 10.1007/s00466-004-0593-y.

[7] G. Marckmann and E. Verron, "Comparison of hyperelastic models for rubber-like materials", *Rubber Chemistry and Technology*, vol. 79, pp. 835–858, Nov. 2006. DOI: 10.5254/1.3547969.

[8] R. S. Rivlin and D. W. Saunders, "Large Elastic Deformations of Isotropic Materials. IV. Further Developments of the General Theory", *Philosophical Transactions of the Royal Society of London Series A*, vol. 241, no. 835, pp. 379–397, Oct. 1948. DOI: 10.1098/rsta.1948.0024.

[9] G. Saccomandi and L. Vergori, "Generalised mooney–rivlin models for brain tissue: A theoretical perspective", *International Journal of Non-Linear Mechanics*, vol. 109, pp. 9–14, 2019, ISSN: 0020-7462. DOI: https://doi.org/10.1016/j.ijnonlinmec.2018.09.008.

[10] R. Ogden, "Large deformation isotropic elasticity - on the correlation of theory and experiment for incompressible rubberlike solids", *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 326, no. 1567, pp. 565–584, 1972. DOI: 10.1098/rspa.1972.0026. eprint: http://rspa.royalsocietypublishing.org/content/326/1567/565.full.pdf+html. [Online]. Available: http://rspa.royalsocietypublishing.org/content/326/1567/565.abstract.

[11] B. Kim, S. B. Lee, J. Lee, S. Cho, H. Park, S. Yeom, and S. H. Park, "A comparison among neo-hookean model, mooney-rivlin model, and ogden model for chloroprene rubber", *International Journal of Precision Engineering and Manufacturing*, vol. 13, no. 5, pp. 759–764, May 2012. DOI: 10.1007/s12541-012-0099-y. [Online]. Available: https://doi.org/10.1007%5C%2Fs12541-012-0099-y.

[12] J. Carvill, *Mechanical engineer's data handbook*. Butterworth-Heinemann, 1993, ISBN: 9780750610148.

[13] W. Pilkey and D. Pilkey, *Peterson's stress concentration factors*. Wiley, 2008, ISBN: 9780470048245.

[14] A. Mata, A. J. Fleischman, and S. Roy, "Characterization of polydimethylsiloxane (PDMS) properties for biomedical micro/nanosystems", *Biomedical Microdevices*, vol. 7, no. 4, pp. 281–293, Dec. 2005. DOI: 10.1007/s10544-005-6070-2. [Online]. Available: https://doi.org/10.1007%5C%2Fs10544-005-6070-2.

[15] "Jacc 055 : Linear polydimethylsiloxanes cas no. 63148-62-9", Joint Assessment of Commodity Chemicals, Tech. Rep., Dec. 2011.

[16] N. R. P. Moreira, "Estudo de várias propriedades mecânicas do polidimetilsiloxano (pdms) usado em dispositivos biomédicos", Master's thesis, Escola Superior de Tecnologia e Gestão, Instituto Politécnico de Bragança, The address of the publisher, Oct. 2013.

[17] Z. Isiksacan, M. T. Guler, B. Aydogdu, I. Bilican, and C. Elbuken, "Rapid fabrication of microfluidic PDMS devices from reusable PDMS molds using laser ablation", *Journal of Micromechanics and Microengineering*, vol. 26, no. 3, p. 035 008, Feb. 2016. DOI: 10.1088/0960-1317/26/3/035008. [Online]. Available: https://doi.org/10.1088/0960-1317/26/3/035008.

[18] J. D. L. Rosas, "Implementação de um sistema para medição das deformações no plano pela técnica de correlação digital de imagem", Master's thesis, Escola Superior de Tecnologia e Gestão, Instituto Politécnico de Bragança, http://hdl.handle.net/10400.22/15024, Oct. 2019.

[19] B. Mobasher, "5 - textile fiber composites: Testing and mechanical behavior", in *Textile Fibre Composites in Civil Engineering*, T. Triantafillou, Ed., Woodhead Publishing, 2016, pp. 101–150, ISBN: 978-1-78242-446-8. DOI: https://doi.org/10.1016/B978-1-78242-446-8.00006-9. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9781782424468000069.

[20] M. Sutton, W. Wolters, W. Peters, W. Ranson, and S. McNeill, "Determination of displacements using an improved digital correlation method", *Image and Vision Computing*, vol. 1, no. 3, pp. 133–139, 1983, ISSN: 0262-8856. DOI: https://doi.org/10.1016/0262-8856(83)90064-1. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0262885683900641.

[21] M. Sutton, J.-J. Orteu, and H. Schreier, *Image correlation for shape, motion and deformation measurements. basic concepts, theory and applications*. Jan. 2009, ISBN: 978-0-387-78746-6. DOI: 10.1007/978-0-387-78747-3.

[22] Y. Dong and B. Pan, "A review of speckle pattern fabrication and assessment for digital image correlation", *Experimental Mechanics*, vol. 57, pp. 1–21, May 2017. DOI: 10.1007/s11340-017-0283-1.

[23] W. S. Cleveland and S. J. Devlin, "Locally weighted regression: An approach to regression analysis by local fitting", *Journal of the American Statistical Association*, vol. 83, no. 403, pp. 596–610, 1988. DOI: 10.1080/01621459.1988.10478639. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1988.10478639. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1988.10478639.

Appendix A

Adjusted Stress-Strain Data and R Code

REAL POINTS

strain	loess_0.1	loess_0.25	loess_0.5
0	0	0	0
0,01	0,004972	0,004791	0,007544
0,02	0,01383	0,013833	0,015795
0,03	0,022589	0,022711	0,023955
0,04	0,031263	0,031423	0,032021
0,05	0,039832	0,039969	0,039987
0,06	0,048111	0,04835	0,047849
0,07	0,056353	0,056568	0,055602
0,08	0,064614	0,064619	0,063253
0,09	0,072734	0,072505	0,070807
0,1	0,080678	0,080229	0,078268
0,11	0,088183	0,087784	0,085639
0,12	0,095408	0,095169	0,092924
0,13	0,1023	0,102393	0,100125
0,14	0,109364	0,109408	0,107233
0,15	0,116322	0,116174	0,114227
0,16	0,123077	0,122823	0,121119
0,17	0,129575	0,129448	0,127918
0,18	0,135798	0,135904	0,134638
0,19	0,141929	0,142236	0,141288
0,2	0,148031	0,14853	0,147877
0,21	0,154283	0,154706	0,154308
0,22	0,160603	0,160764	0,160565
0,23	0,166983	0,166784	0,166692
0,24	0,173212	0,172723	0,172734
0,25	0,17923	0,178517	0,178737
0,26	0,184926	0,184302	0,184744
0,27	0,190378	0,190163	0,190751
0,28	0,195581	0,195966	0,196596
0,29	0,20104	0,201743	0,202311
0,3	0,206678	0,207546	0,207951
0,31	0,212555	0,213365	0,213569
0,32	0,218705	0,219171	0,219221
0,33	0,225049	0,224946	0,224961
0,34	0,231305	0,230734	0,230687
0,35	0,237295	0,23655	0,236287
0,36	0,242913	0,242319	0,241815
0,37	0,248359	0,247976	0,247323
0,38	0,253644	0,253541	0,252865
0,39	0,259087	0,259087	0,258494
0,4	0,264564	0,264663	0,264239
0,41	0,270107	0,270106	0,269913
0,42	0,275447	0,27551	0,275512
0,43	0,280767	0,281044	0,281089
0,44	0,286283	0,286633	0,2867
0,45	0,291948	0,292216	0,292398
0,46	0,297779	0,297888	0,298238
0,47	0,303791	0,303673	0,304177
0,48	0,309843	0,309488	0,310051
0,49	0,315582	0,315384	0,315905
0,5	0,321454	0,321401	0,321791
0,51	0,327436	0,327496	0,327759
0,52	0,333478	0,333671	0,333862
0,53	0,339654	0,339939	0,340145
0,54	0,346098	0,346299	0,346475
0,55	0,352617	0,352741	0,352801
0,56	0,359207	0,359263	0,359167
0,57	0,365968	0,365868	0,365616
0,58	0,372788	0,372564	0,372193
0,59	0,379516	0,379342	0,378939
0,6	0,386354	0,386183	0,385858
0,61	0,393283	0,393061	0,392805
0,62	0,400163	0,400027	0,399799
0,63	0,407126	0,407129	0,406877
0,64	0,414172	0,414268	0,414079
0,65	0,421417	0,421482	0,421444
0,66	0,428811	0,428873	0,42901
0,67	0,436324	0,436396	0,436711
0,68	0,443985	0,443993	0,444467
0,69	0,45172	0,451771	0,452312
0,7	0,459752	0,459784	0,460279
0,71	0,467997	0,467943	0,468401
0,72	0,476268	0,476267	0,47671
0,73	0,484674	0,484789	0,485225
0,74	0,493132	0,493543	0,493823
0,75	0,502054	0,502473	0,502501
0,76	0,511268	0,511504	0,511298
0,77	0,520785	0,520681	0,520254
0,78	0,530293	0,530031	0,529409
0,79	0,539752	0,539498	0,538802
0,8	0,549388	0,549032	0,548415
0,81	0,559106	0,558659	0,558151
0,82	0,568936	0,568438	0,568031
0,83	0,578696	0,578393	0,578082
0,84	0,588342	0,588374	0,588328
0,85	0,598253	0,598498	0,598796
0,86	0,608437	0,608924	0,609509
0,87	0,619189	0,619506	0,620423
0,88	0,630127	0,630232	0,631509
0,89	0,641293	0,641256	0,642775
0,9	0,652577	0,652627	0,654226
0,91	0,664106	0,664242	0,665869
0,92	0,675984	0,676097	0,67771
0,93	0,688098	0,688207	0,68975
0,94	0,700471	0,700604	0,701976
0,95	0,71307	0,713228	0,714387
0,96	0,725954	0,726012	0,726986
0,97	0,739227	0,739	0,739774
0,98	0,752741	0,752212	0,752753
0,99	0,766465	0,765632	0,765927
1	0,77991	0,779261	0,779298

```
# ------------ Stress-Strain Curve Fitting ----------

# Needed libraries to run the code
library(dplyr)
library(data.table)
library(reshape2)
library(ggplot2)
library(plotly)
library(forcats)

# Reading Data
setwd("C:/Users/elima/Documents/IPB/Thesis/Frames")
getwd()
rm(list=ls(all=TRUE))
file <- read.csv("standart_test.csv", header = TRUE, sep = ";", dec = ",")
file2 <- filter(file, displacement<=75)

# Converting force-displacement into stress-strain
area <- 10 * 2
gauge_length <- 71
stress <- file2$force/(area)
strain <- file2$displacement/gauge_length
file3 <- as.data.frame(cbind(stress, strain))
#ggplotly(ggplot() + geom_point(data=file3, aes(x=strain, y=stress)) +
theme_classic())

# Creating the fitted curve
loessMod10 <- loess(stress ~ strain, data=file3, span=0.10) # 10% smoothing
span
loessMod25 <- loess(stress ~ strain, data=file3, span=0.25) # 25% smoothing
span
loessMod50 <- loess(stress ~ strain, data=file3, span=0.50) # 50% smoothing
span

smoothed10 <- predict(loessMod10)
smoothed25 <- predict(loessMod25)
smoothed50 <- predict(loessMod50)

#------------------Exporting Points and Plotting-------------

exp_1 <- seq(0,1,length.out = 101)
loess_0.1 <- predict(loessMod10, data.frame(strain = seq(0,1,length.out =
101)))
loess_0.25 <- predict(loessMod25, data.frame(strain = seq(0,1,length.out =
101)))
loess_0.5 <- predict(loessMod50, data.frame(strain = seq(0,1,length.out =
101)))

file6 <- as.data.frame(cbind(exp_1, loess_0.1, loess_0.25, loess_0.5))
file6[1,2:4] <- 0
file_exp <- file6
file6 <- melt(data = file6, id.vars = "exp_1", measure.vars =
c("loess_0.1","loess_0.25","loess_0.5"))
colnames(file6)[1] <- "strain"
file6 <- file6[,order(colnames(file6))]

file_real <- file3
file_real[,"variable"] <- "real_points"
```

```
colnames(file_real)[1] <- "value"
file_real <- file_real[,order(colnames(file_real))]

file_plot <- rbind(file_real, file6)
colnames(file_plot)[2] <- "stress"
file_plot$variable <- as.factor(file_plot$variable)
file_plot$variable <- relevel(file_plot$variable, "real_points")
file_plot <- file_plot %>% arrange(desc(variable))

require(ggplot2)
require(ggsubplot)

# Plotting
p1 <- ggplot(data = file_plot, aes(x=strain, y=stress, group = variable,
color = variable)) +
  geom_point() + geom_line() +  theme_classic() +
  scale_color_manual(values=c("black", "red", "blue", "green")) +
  geom_subplot(data=file_plot, aes(x=5, y=1, group = variable,
                                   subplot = geom_point(aes(x=strain,
y=stress, group = variable, color = variable))),
               width=4, height=4,reference=ref_box(fill = "white", color =
"black"))
p1


require(ggplot2)
main <- ggplot(data = file_plot, aes(x=strain, y=stress, group = variable,
color = variable)) +
  geom_point(size=0.1) + geom_line() +  theme_classic() +
  scale_color_manual(values=c("black", "red", "blue", "green")) +
  geom_rect(data=file_plot[1,],xmin=0.5, ymin=0.32, xmax=0.6, ymax=0.4,
fill="grey", color = "black", alpha=0) +
  scale_x_continuous(limits=c(0, 1)) + scale_y_continuous(limits = c(0,1))
+
  xlab("Strain [mm/mm]") + ylab("Stress[MPa]") +
  theme(legend.position = c(0.85, 0.25),
        legend.background = element_rect(color="black", fill=NA, size=0.5,
linetype="solid"))

sub <- ggplot(data = file_plot, aes(x=strain, y=stress, group = variable,
color = variable)) +
  geom_point(size=0.1) + geom_line() +
  theme_classic() +
  scale_color_manual(values=c("black", "red", "blue", "green")) +
  scale_x_continuous(limits=c(0.5, 0.55), breaks=c(0.5, 0.55)) +
  scale_y_continuous(limits=c(0.32,0.36), breaks=c(0.32, 0.36)) +
  theme(axis.title.x = element_blank(), axis.title.y = element_blank(),
legend.position = "none",
        panel.border = element_rect(colour = "black", fill="NA"),
axis.text = element_text(size = rel(0.8)))

main + annotation_custom(ggplotGrob(sub), xmin=0, xmax=0.6, ymin=0.5,
ymax=1)

setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("stress_strain.pdf", dpi=300)

p2 <- ggplot() +
```

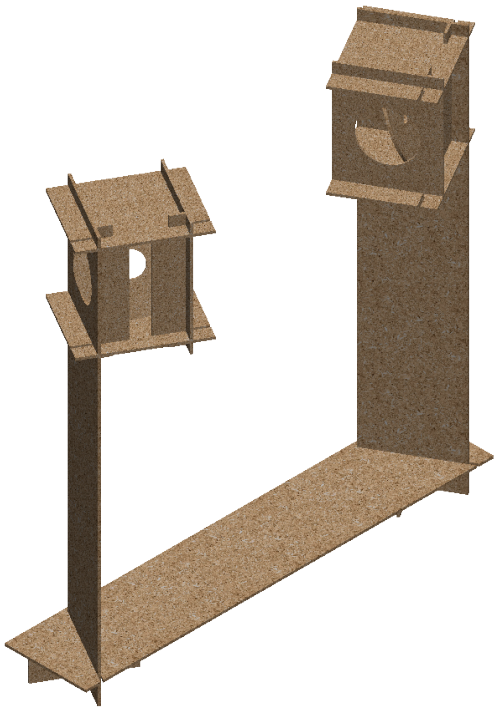
```
  geom_point(data = file3, aes(x = strain, y = stress)) +
  geom_point(data = file6, aes(x = exp_1, y = exp_2), color="blue") +
  geom_line(data = file6, aes(x = exp_1, y = exp_2), color="blue") +
  geom_point(data = file6, aes(x = exp_1, y = exp_3), color="red") +
  geom_line(data = file6, aes(x = exp_1, y = exp_3), color="red") +
  geom_point(data = file6, aes(x = exp_1, y = exp_4), color="green") +
  geom_line(data = file6, aes(x = exp_1, y = exp_4), color="green") +
  ggtitle("Real Points And Fitted Curves") + theme_classic()
p2
ggplotly(p2)

colnames(file_exp)[1] <- "strain"

write.table(file_exp, file = "stress_strain_curve.txt", append = FALSE,
quote = FALSE, sep = "\t", dec = ",",
            row.names = FALSE, col.names = TRUE)
```

Appendix B

Light Support



Nº DO ITEM	Nº DA PEÇA	DESCRIÇÃO	QTD.
1	Base		1
2	Articulador		2
3	Auxilio de posição		4
4	Lateral		2
5	Lateral superior		2
6	Suporte inferior		2
7	Suporte superior		4

SE NÃO ESPECIFICADO:
DIMENSÕES EM MILÍMETROS
ACABAM. SUPERFÍCIE:
TOLERÂNCIAS:
LINEAR:
ANGULAR:

ACABAMENTO:

REBARBAR E QUEBRAR ARESTAS AGUDAS

NÃO MUDAR ESCALA DO DESENHO

REVISÃO

	NOME	ASSINATURA	DATA
DESEN.			
VERIF.			
APROV.			
MANUF.			
QUALID.			

TÍTULO:

Montagem

MATERIAL:

DES. Nº

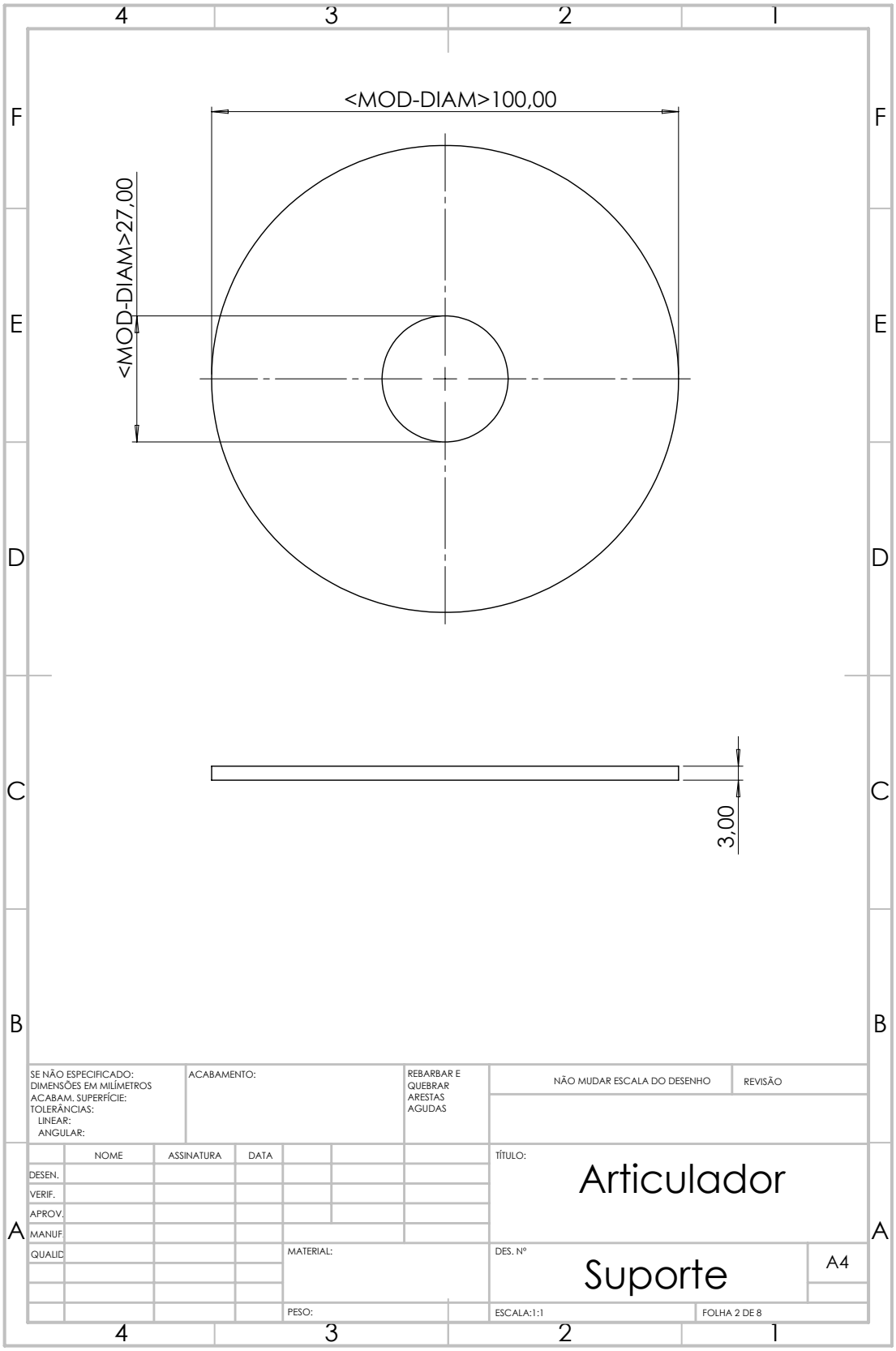
Suporte

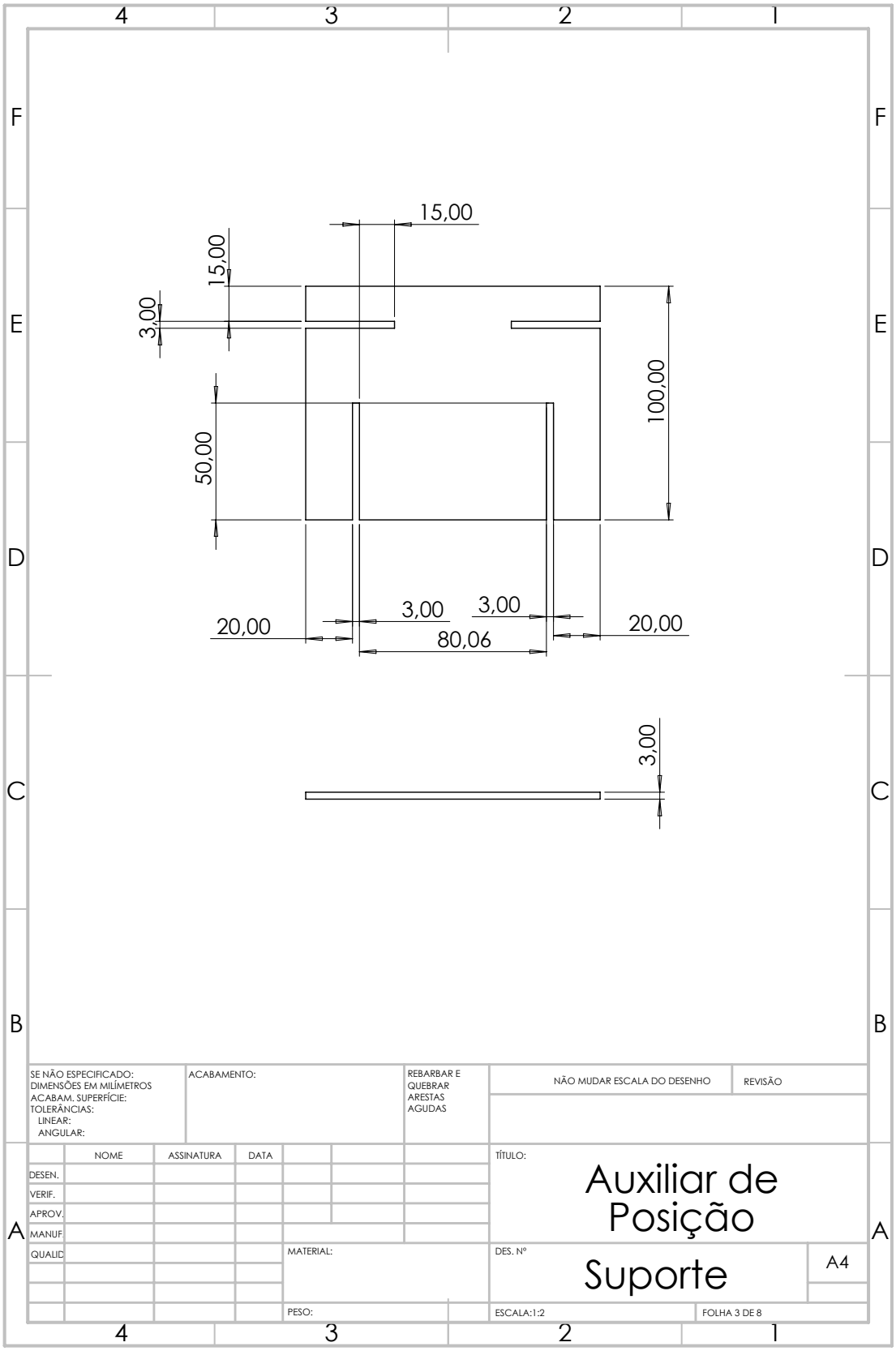
A4

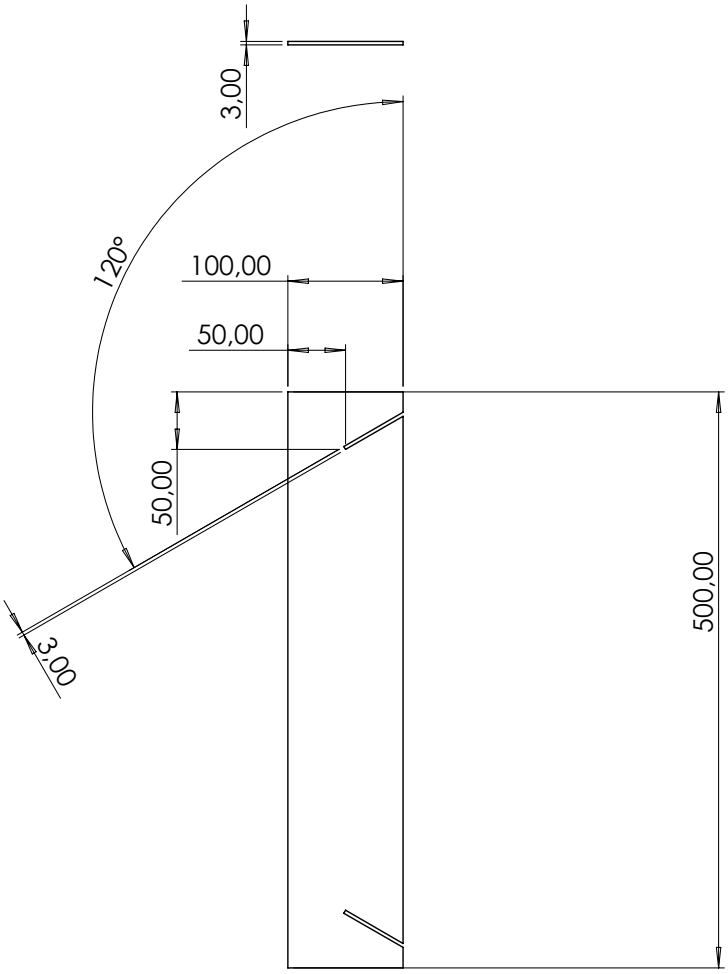
PESO:

ESCALA:1:10

FOLHA 1 DE 8


4
3
2
1
F
E
D
C
B
A
<MOD-DIAM>100,00
<MOD-DIAM>27,00
3,00
SE NÃO ESPECIFICADO:
DIMENSÕES EM MILÍMETROS
ACABAM. SUPERFÍCIE:
TOLERÂNCIAS:
LINEAR:
ANGULAR:
ACABAMENTO:
REBARBAR E QUEBRAR ARESTAS AGUDAS
NÃO MUDAR ESCALA DO DESENHO
REVISÃO
NOME
ASSINATURA
DATA
DESEN.
VERIF.
APROV.
MANUF.
QUALID
TÍTULO:
Articulador
MATERIAL:
DES. Nº
Suporte
A4
PESO:
ESCALA:1:1
FOLHA 2 DE 8


4
3
2
1
F
E
D
C
B
A
15,00
15,00
3,00
100,00
50,00
20,00
3,00
3,00
80,06
20,00
3,00
SE NÃO ESPECIFICADO:
DIMENSÕES EM MILÍMETROS
ACABAM. SUPERFÍCIE:
TOLERÂNCIAS:
LINEAR:
ANGULAR:
ACABAMENTO:
REBARBAR E QUEBRAR ARESTAS AGUDAS
NÃO MUDAR ESCALA DO DESENHO
REVISÃO
NOME
ASSINATURA
DATA
DESEN.
VERIF.
APROV.
MANUF.
QUALID
TÍTULO:
Auxiliar de Posição
MATERIAL:
DES. Nº
Suporte
A4
PESO:
ESCALA:1:2
FOLHA 3 DE 8



SE NÃO ESPECIFICADO: DIMENSÕES EM MILÍMETROS ACABAM. SUPERFÍCIE: TOLERÂNCIAS: LINEAR: ANGULAR:	ACABAMENTO:	REBARBAR E QUEBRAR ARESTAS AGUDAS	NÃO MUDAR ESCALA DO DESENHO	REVISÃO		
	NOME	ASSINATURA	DATA			
DESEN.						
VERIF.						
APROV.						
MANUF.						
QUALID.						
MATERIAL:						
PESO:						

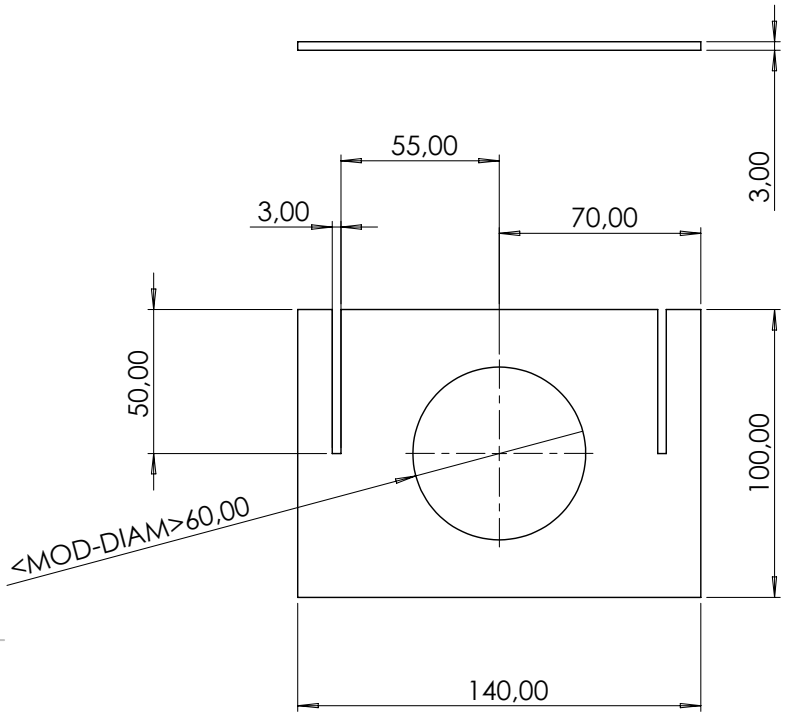
TÍTULO: Base

DES. Nº Suporte

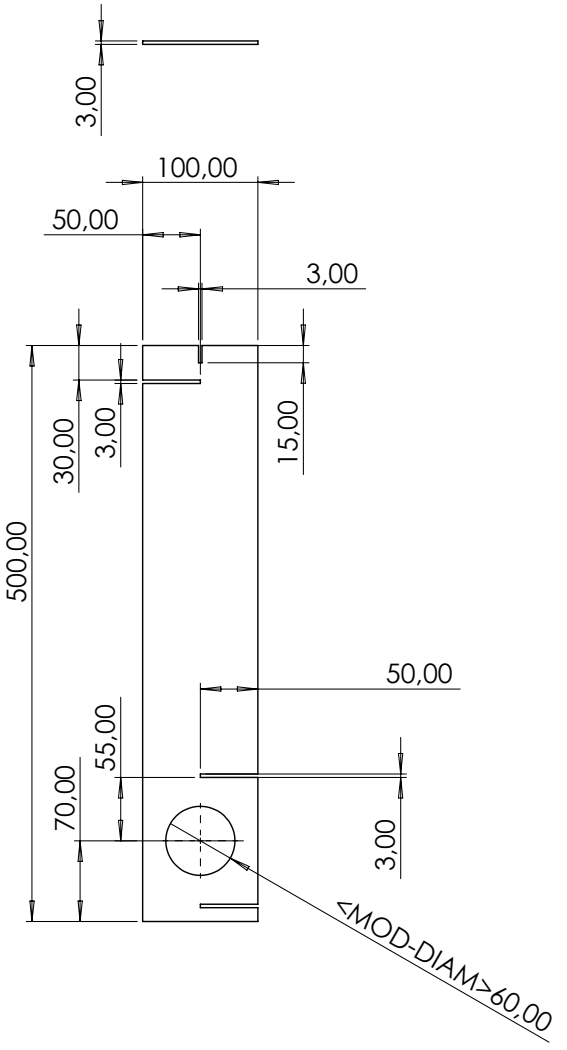
A4

ESCALA:1:5

FOLHA 4 DE 8



SE NÃO ESPECIFICADO: DIMENSÕES EM MILÍMETROS ACABAM. SUPERFÍCIE: TOLERÂNCIAS: LINEAR: ANGULAR:	ACABAMENTO:	REBARBAR E QUEBRAR ARESTAS AGUDAS	NÃO MUDAR ESCALA DO DESENHO	REVISÃO		
	NOME	ASSINATURA	DATA	TÍTULO: Lateral Superior		
DESEN.						
VERIF.						
APROV.						
MANUF.						
QUALID.				MATERIAL:	DES. Nº Suporte	A4
				PESO:	ESCALA:1:2	FOLHA 5 DE 8



SE NÃO ESPECIFICADO: DIMENSÕES EM MILÍMETROS ACABAM. SUPERFÍCIE: TOLERÂNCIAS: LINEAR: ANGULAR:	ACABAMENTO:	REBARBAR E QUEBRAR ARESTAS AGUDAS	NÃO MUDAR ESCALA DO DESENHO / REVISÃO

	NOME	ASSINATURA	DATA
DESEN.			
VERIF.			
APROV.			
MANUF.			
QUALID.			

TÍTULO: Lateral

MATERIAL:

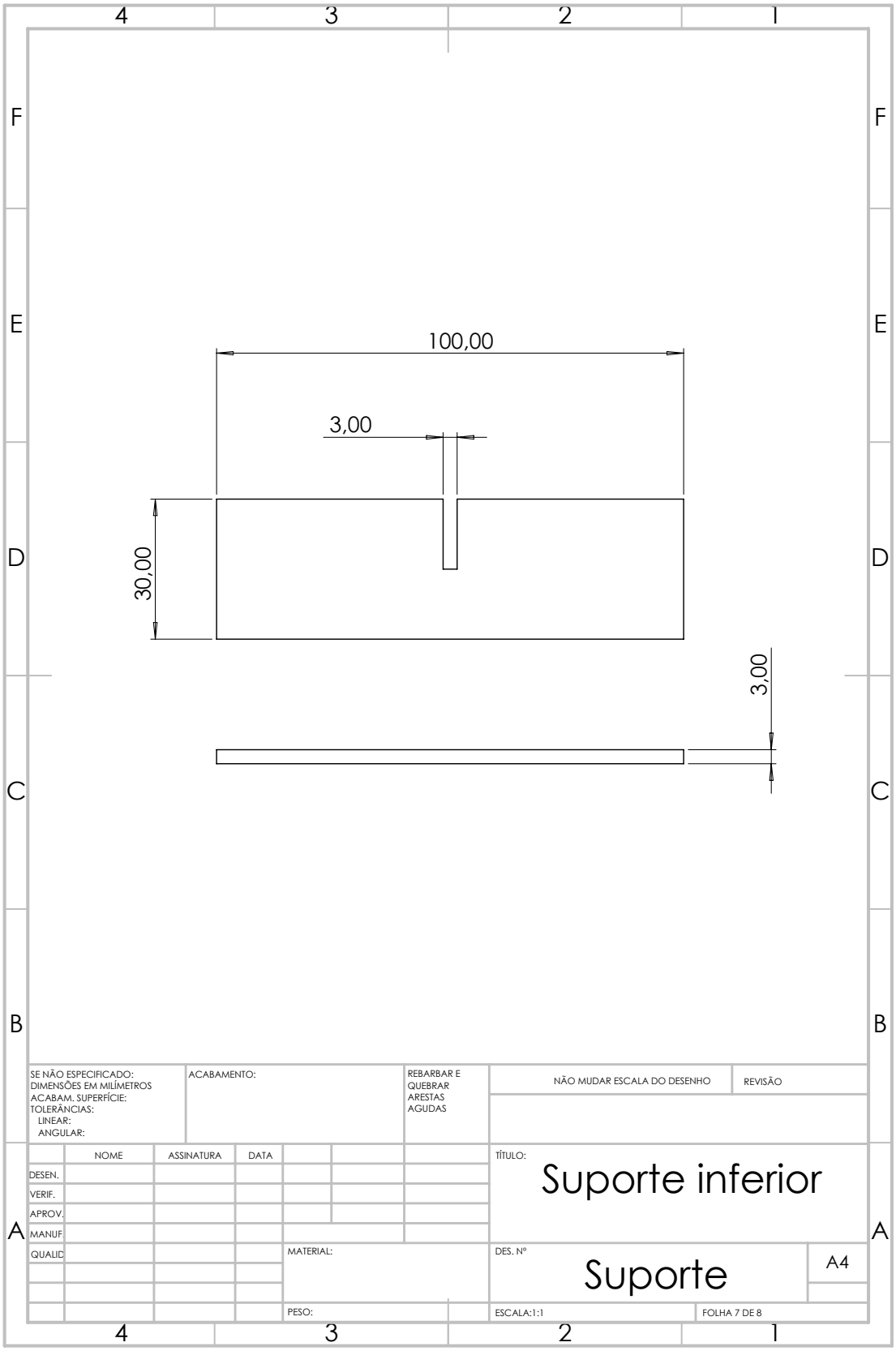
DES. Nº: Suporte

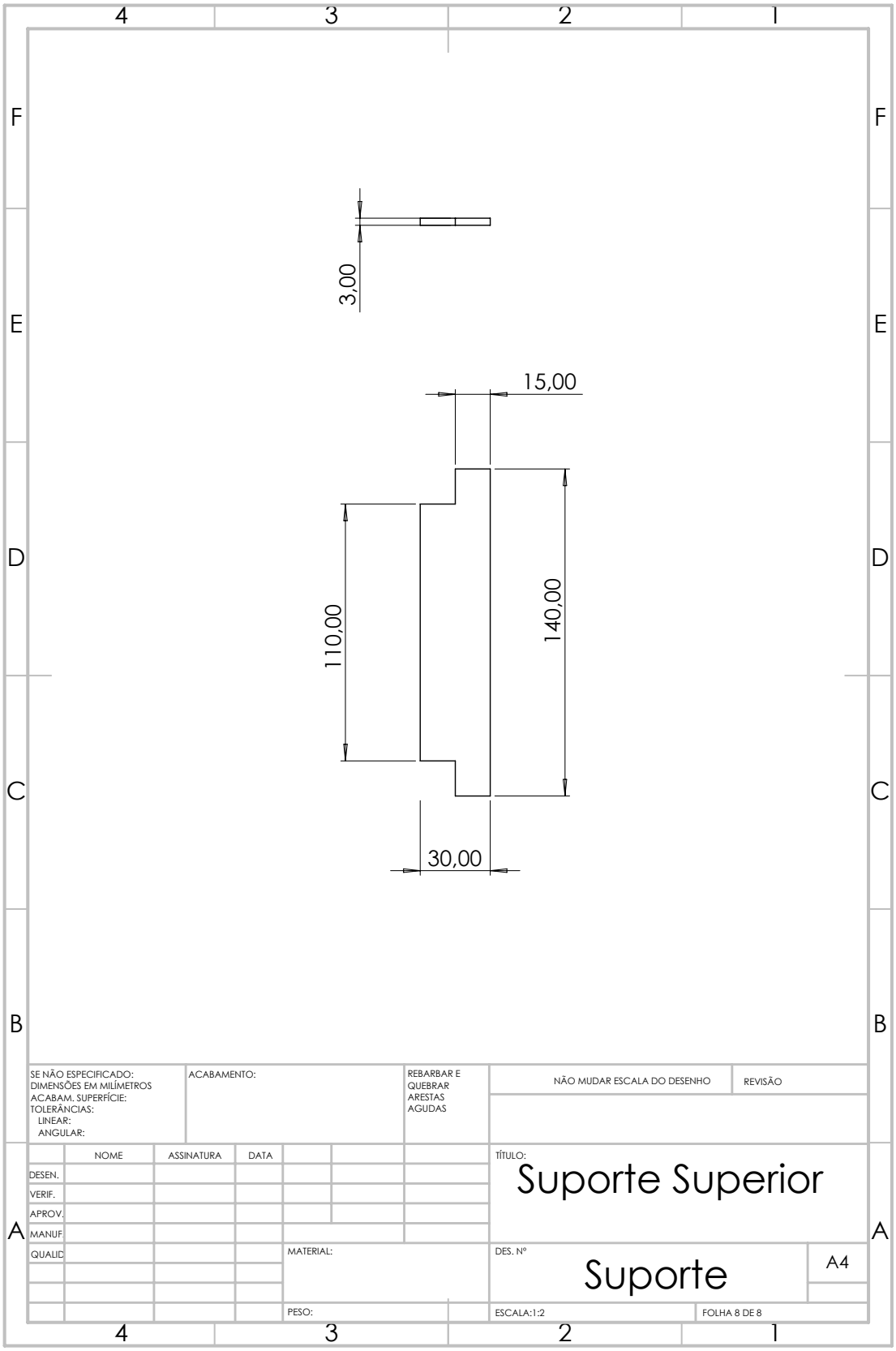
A4

PESO:

ESCALA:1:5

FOLHA 6 DE 8


4
3
2
1
F
E
D
C
B
A
100,00
3,00
30,00
3,00
SE NÃO ESPECIFICADO:
DIMENSÕES EM MILÍMETROS
ACABAM. SUPERFÍCIE:
TOLERÂNCIAS:
LINEAR:
ANGULAR:
ACABAMENTO:
REBARBAR E QUEBRAR ARESTAS AGUDAS
NÃO MUDAR ESCALA DO DESENHO
REVISÃO
NOME
ASSINATURA
DATA
DESEN.
VERIF.
APROV.
MANUF.
QUALID
TÍTULO:
Suporte inferior
MATERIAL:
DES. Nº
Suporte
A4
PESO:
ESCALA:1:1
FOLHA 7 DE 8


4
3
2
1
F
E
D
C
B
A
3,00
15,00
110,00
140,00
30,00
SE NÃO ESPECIFICADO:
DIMENSÕES EM MILÍMETROS
ACABAM. SUPERFÍCIE:
TOLERÂNCIAS:
LINEAR:
ANGULAR:
ACABAMENTO:
REBARBAR E QUEBRAR ARESTAS AGUDAS
NÃO MUDAR ESCALA DO DESENHO
REVISÃO
NOME
ASSINATURA
DATA
DESEN.
VERIF.
APROV.
MANUF.
QUALID
TÍTULO:
Suporte Superior
MATERIAL:
DES. Nº
Suporte
A4
PESO:
ESCALA:1:2
FOLHA 8 DE 8

Appendix C

Renaming Frames C++ Code

```c
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
FILE *file;

main()
{
    int i,a, fim, intervalo, ret;
    char s[200], oldname[200], newname[200];
    a=1;
    fim = 3500;
    intervalo = 10;

    for(i=0; i<=3500; i++)
    {
        sprintf(s, "C:/Users/elima/Documents/IPB/Thesis/Frames/1/Untitled_%06d.jpeg", i);
        if (a==1){
            printf(" ------> file Untitled_%06d.jpeg NOT deleted\n", i);
            a = 10;
        }
        else {
            if (remove(s) == 0) {
            printf(" ---> file Untitled_%06d.jpeg deleted.\n", i);
            }
            else {
                printf(" ------> Unable to delete the file Untitled_%06d.jpeg\n", i);
            }
            a = a-1;
        }
    }


    for(i=0; i<=fim/intervalo; i++)
    {
        sprintf(oldname, "Untitled_%06d.jpeg", i*intervalo);
        sprintf(newname, "Untitled_%05d.jpeg", i);
        ret = rename(oldname, newname);
        if(ret == 0) {
            printf("----> File Untitled_%06d.jpeg renamed successfully\n", i*intervalo);
        } else {
            printf("-------> Unable to rename the file Untitled_%06d.jpeg\n", i*intervalo);
        }
    }

    system("pause");
    return 0;
}

```

Appendix D

Test Results and R Code

```c
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
FILE *file;

main()
{
    int i,a, fim, intervalo, ret;
    char s[200], oldname[200], newname[200];
    a=1;
    fim = 3500;
    intervalo = 10;

    for(i=0; i<=3500; i++)
    {
        sprintf(s, "C:/Users/elima/Documents/IPB/Thesis/Frames/1/Untitled_%06d.jpeg", i);
        if (a==1){
            printf(" ------> file Untitled_%06d.jpeg NOT deleted\n", i);
            a = 10;
        }
        else {
            if (remove(s) == 0) {
            printf(" ---> file Untitled_%06d.jpeg deleted.\n", i);
            }
            else {
                printf(" ------> Unable to delete the file Untitled_%06d.jpeg\n", i);
            }
            a = a-1;
        }
    }


    for(i=0; i<=fim/intervalo; i++)
    {
        sprintf(oldname, "Untitled_%06d.jpeg", i*intervalo);
        sprintf(newname, "Untitled_%05d.jpeg", i);
        ret = rename(oldname, newname);
        if(ret == 0) {
            printf("----> File Untitled_%06d.jpeg renamed successfully\n", i*intervalo);
        } else {
            printf("-------> Unable to rename the file Untitled_%06d.jpeg\n", i*intervalo);
        }
    }

    system("pause");
    return 0;
}

```

```
# \\\\\\\\\\\\\\\\\\\\\\\\\\\\ Test 1 ////////////////////////////

# ----------- 1 - Leitura de dados (ANSYS) ---------------
setwd("C:/Users/elima/Documents/IPB/Thesis/Frames")
getwd()
rm(list=ls(all=TRUE))
library(reshape2)
library(ggplot2)
library(dplyr)
library(viridis)
nodes_coord <- read.csv("nodes_coordinates_1.csv", header = TRUE, sep = ";", dec = ",")
nodes_elem <- read.csv("nodes_elements_1.csv", header = TRUE, sep = ";", dec = ",")
elem_str_ogd <- read.csv("element_strain_ogden_1.csv", header = TRUE, sep = ";", dec = ",")
nodes_str_ogd <- read.csv("nodes_strain_ogden_1.csv", header = TRUE, sep = ";", dec = ",")
elem_str_moon <- read.csv("element_strain_mooney_1.csv", header = TRUE, sep = ";", dec = ",")
nodes_str_moon <- read.csv("nodes_strain_mooney_1.csv", header = TRUE, sep = ";", dec = ",")
elem_str_neo <- read.csv("element_strain_neohook_1.csv", header = TRUE, sep = ";", dec = ",")
nodes_str_neo <- read.csv("nodes_strain_neohook_1.csv", header = TRUE, sep = ";", dec = ",")




# ----------- 2 - Transformação de dados para obtenção de coordenadas dos Elementos (ANSYS)
---------------
# ------------2.1 - OGDEN ---------------

file_ogd_1 <- merge(elem_str_ogd, nodes_elem, all=TRUE, by.x = "ELEM", by.y = "ELEM")
file_ogd_1 <- file_ogd_1[,-3:-7]
for (i in 1:8) {
  file_ogd_1 <- merge(file_ogd_1, nodes_coord[,c(1,2)], all.x=TRUE,
                 by.x = paste("NODE",i, sep = "_", collapse = NULL), by.y = "NODE")
  colnames(file_ogd_1)[ncol(file_ogd_1)] <- paste("NODE",i,"X", sep = "_", collapse = NULL)
}
for (i in 1:8) {
  file_ogd_1 <- merge(file_ogd_1, nodes_coord[,c(1,3)], all.x=TRUE,
                 by.x = paste("NODE",i, sep = "_", collapse = NULL), by.y = "NODE")
  colnames(file_ogd_1)[ncol(file_ogd_1)] <- paste("NODE",i,"Y", sep = "_", collapse = NULL)
}
par(pty="s")
plot(nodes_coord$Y~nodes_coord$X, type="p",pch=".", asp=1)

file_ogd_2 <- file_ogd_1[,-10]
file_ogd_2 <- file_ogd_2[,-1:-8]
file_ogd_2 <- file_ogd_2[,-10:-17]
colnam <- colnames(file_ogd_2)
colnam <- colnam[-1]
file_ogd_2 <- melt(data = file_ogd_2, id.vars = "ELEM", measure.vars = colnam)

file_ogd_3 <- file_ogd_1[,-10]
file_ogd_3 <- file_ogd_3[,-1:-8]
file_ogd_3 <- file_ogd_3[,-2:-9]
colnam <- colnames(file_ogd_3)
colnam <- colnam[-1]
file_ogd_3 <- melt(data = file_ogd_3, id.vars = "ELEM", measure.vars = colnam)
file_ogd_4 <- cbind(file_ogd_2, file_ogd_3)
file_ogd_4 <- file_ogd_4[,-4]
colnames(file_ogd_4)[c(3,5)] <- c("X","Y")
file_ogd_5 <- file_ogd_1[,c("ELEM","EPTOY")]
mesh_ogd <- merge(file_ogd_5,file_ogd_4, by = c("ELEM"))
mesh_ogd <- mesh_ogd[order(mesh_ogd$ELEM,
                           atan2(mesh_ogd$X-mean(mesh_ogd$X),mesh_ogd$Y-mean(mesh_ogd$Y))),]
mesh_ogd_2 <- data.frame()
mesh_ogd_for <- data.frame()
for(i in unique(mesh_ogd$ELEM)){
  mesh_ogd_for <- filter(mesh_ogd, ELEM==i)
  mesh_ogd_for <- mesh_ogd_for[order(atan2(mesh_ogd_for$X-mean(mesh_ogd_for$X),mesh_ogd_for$Y-
mean(mesh_ogd_for$Y))),]
  mesh_ogd_2 <- rbind(mesh_ogd_2, mesh_ogd_for)
}
```

```
# ------------2.2 - Mooney_rivlin ---------------

file_moon_1 <- merge(elem_str_moon, nodes_elem, all=TRUE, by.x = "ELEM", by.y = "ELEM")
file_moon_1 <- file_moon_1[,-3:-7]
for (i in 1:8) {
  file_moon_1 <- merge(file_moon_1, nodes_coord[,c(1,2)], all.x=TRUE,
                       by.x = paste("NODE",i, sep = "_", collapse = NULL), by.y = "NODE")
  colnames(file_moon_1)[ncol(file_moon_1)] <- paste("NODE",i,"X", sep = "_", collapse = NULL)
}
for (i in 1:8) {
  file_moon_1 <- merge(file_moon_1, nodes_coord[,c(1,3)], all.x=TRUE,
                       by.x = paste("NODE",i, sep = "_", collapse = NULL), by.y = "NODE")
  colnames(file_moon_1)[ncol(file_moon_1)] <- paste("NODE",i,"Y", sep = "_", collapse = NULL)
}
par(pty="s")
plot(nodes_coord$Y~nodes_coord$X, type="p",pch=".",asp=1)

file_moon_2 <- file_moon_1[,-10]
file_moon_2 <- file_moon_2[,-1:-8]
file_moon_2 <- file_moon_2[,-10:-17]
colnam <- colnames(file_moon_2)
colnam <- colnam[-1]
file_moon_2 <- melt(data = file_moon_2, id.vars = "ELEM", measure.vars = colnam)

file_moon_3 <- file_moon_1[,-10]
file_moon_3 <- file_moon_3[,-1:-8]
file_moon_3 <- file_moon_3[,-2:-9]
colnam <- colnames(file_moon_3)
colnam <- colnam[-1]
file_moon_3 <- melt(data = file_moon_3, id.vars = "ELEM", measure.vars = colnam)
file_moon_4 <- cbind(file_moon_2, file_moon_3)
file_moon_4 <- file_moon_4[,-4]
colnames(file_moon_4)[c(3,5)] <- c("X","Y")
file_moon_5 <- file_moon_1[,c("ELEM","EPTOY")]
mesh_moon <- merge(file_moon_5,file_moon_4, by = c("ELEM"))
mesh_moon <- mesh_moon[order(mesh_moon$ELEM,
                             atan2(mesh_moon$X-mean(mesh_moon$X),mesh_moon$Y-mean(mesh_moon$Y))),]
mesh_moon_2 <- data.frame()
mesh_moon_for <- data.frame()
for(i in unique(mesh_moon$ELEM)){
  mesh_moon_for <- filter(mesh_moon, ELEM==i)
  mesh_moon_for <- mesh_moon_for[order(atan2(mesh_moon_for$X-mean(mesh_moon_for$X),mesh_moon_for$Y-
mean(mesh_moon_for$Y))),]
  mesh_moon_2 <- rbind(mesh_moon_2, mesh_moon_for)
}

# ------------2.3 - NEO_hookean ---------------

file_neo_1 <- merge(elem_str_neo, nodes_elem, all=TRUE, by.x = "ELEM", by.y = "ELEM")
file_neo_1 <- file_neo_1[,-3:-7]
for (i in 1:8) {
  file_neo_1 <- merge(file_neo_1, nodes_coord[,c(1,2)], all.x=TRUE,
                       by.x = paste("NODE",i, sep = "_", collapse = NULL), by.y = "NODE")
  colnames(file_neo_1)[ncol(file_neo_1)] <- paste("NODE",i,"X", sep = "_", collapse = NULL)
}
for (i in 1:8) {
  file_neo_1 <- merge(file_neo_1, nodes_coord[,c(1,3)], all.x=TRUE,
                       by.x = paste("NODE",i, sep = "_", collapse = NULL), by.y = "NODE")
  colnames(file_neo_1)[ncol(file_neo_1)] <- paste("NODE",i,"Y", sep = "_", collapse = NULL)
}
par(pty="s")
plot(nodes_coord$Y~nodes_coord$X, type="p",pch=".",asp=1)

file_neo_2 <- file_neo_1[,-10]
file_neo_2 <- file_neo_2[,-1:-8]
file_neo_2 <- file_neo_2[,-10:-17]
colnam <- colnames(file_neo_2)
colnam <- colnam[-1]
file_neo_2 <- melt(data = file_neo_2, id.vars = "ELEM", measure.vars = colnam)
```

```
file_neo_3 <- file_neo_1[,-10]
file_neo_3 <- file_neo_3[,-1:-8]
file_neo_3 <- file_neo_3[,-2:-9]
colnam <- colnames(file_neo_3)
colnam <- colnam[-1]
file_neo_3 <- melt(data = file_neo_3, id.vars = "ELEM", measure.vars = colnam)
file_neo_4 <- cbind(file_neo_2, file_neo_3)
file_neo_4 <- file_neo_4[,-4]
colnames(file_neo_4)[c(3,5)] <- c("X","Y")
file_neo_5 <- file_neo_1[,c("ELEM","EPTOY")]
mesh_neo <- merge(file_neo_5,file_neo_4, by = c("ELEM"))
mesh_neo <- mesh_neo[order(mesh_neo$ELEM,
                           atan2(mesh_neo$X-mean(mesh_neo$X),mesh_neo$Y-mean(mesh_neo$Y))),]
mesh_neo_2 <- data.frame()
mesh_neo_for <- data.frame()
for(i in unique(mesh_neo$ELEM)){
  mesh_neo_for <- filter(mesh_neo, ELEM==i)
  mesh_neo_for <- mesh_neo_for[order(atan2(mesh_neo_for$X-mean(mesh_neo_for$X),mesh_neo_for$Y-
mean(mesh_neo_for$Y))),]
  mesh_neo_2 <- rbind(mesh_neo_2, mesh_neo_for)
}

# ----------- 3 - Conversão de Polígonos Para identificação de pontos ---------------

# ----------- 3.1 - OGDEN -----------------

#ggplotly(p)
#install.packages("sfheaders")
library(sf)
library(sfheaders)

my_df_sf_ogd <- st_as_sf(mesh_ogd,
                         coords = c('X', 'Y')) %>%
  st_set_crs(26918) %>%
  group_by(ELEM) %>%
  summarise() %>%
  ungroup() %>%  # Just in case
  st_convex_hull()
my_df_sf_ogd <- merge(my_df_sf_ogd,file_ogd_5, by = c("ELEM"))

#install.packages("prevR")
library(prevR)
library(sp)
library(rgeos)
library(sp)
library(rgdal)
library(plotly)
part_ogd <- as(my_df_sf_ogd, 'Spatial')

setwd("C:/Users/elima/Documents/IPB/Thesis/Frames")
points_ogd <- read.csv("real_points_1_1.csv", header = TRUE, sep = ";", dec = ",")

#points <- points[c(-1,-2,-57:-61),]
dat2_ogd <- points_ogd
dat_ogd <- dat2_ogd
coordinates(dat_ogd) <- ~ x_real + y_real
proj4string(dat_ogd) <- proj4string(part_ogd)
ov_ogd <- over(dat_ogd, part_ogd)
my_df_sf_ogd_2 <- filter(my_df_sf_ogd, ELEM %in% ov_ogd$ELEM)


# ----------- 3.2 - Mooney-rivlin -----------------

#ggplotly(p)
#install.packages("sfheaders")
library(sf)
library(sfheaders)

my_df_sf_moon <- st_as_sf(mesh_moon,
                          coords = c('X', 'Y')) %>%
```

```
  st_set_crs(26918) %>%
  group_by(ELEM) %>%
  summarise() %>%
  ungroup() %>%  # Just in case
  st_convex_hull()
my_df_sf_moon <- merge(my_df_sf_moon,file_moon_5, by = c("ELEM"))

#install.packages("prevR")
library(prevR)
library(sp)
library(rgeos)
library(sp)
library(rgdal)
library(plotly)
part_moon <- as(my_df_sf_moon, 'Spatial')

setwd("C:/Users/elima/Documents/IPB/Thesis/Frames")
points_moon <- read.csv("real_points_1_1.csv", header = TRUE, sep = ";", dec = ",")

#points <- points[c(-1,-2,-57:-61),]
dat2_moon <- points_moon
dat_moon <- dat2_moon
coordinates(dat_moon) <- ~ x_real + y_real
proj4string(dat_moon) <- proj4string(part_moon)
ov_moon <- over(dat_moon, part_moon)
my_df_sf_moon_2 <- filter(my_df_sf_moon, ELEM %in% ov_moon$ELEM)
#ggplotly(p)

# ----------- 3.2 - NEO-hookean -----------------

#ggplotly(p)
#install.packages("sfheaders")
library(sf)
library(sfheaders)

my_df_sf_neo <- st_as_sf(mesh_neo,
                         coords = c('X', 'Y')) %>%
  st_set_crs(26918) %>%
  group_by(ELEM) %>%
  summarise() %>%
  ungroup() %>%  # Just in case
  st_convex_hull()
my_df_sf_neo <- merge(my_df_sf_neo,file_neo_5, by = c("ELEM"))

#install.packages("prevR")
library(prevR)
library(sp)
library(rgeos)
library(sp)
library(rgdal)
library(plotly)
part_neo <- as(my_df_sf_neo, 'Spatial')

setwd("C:/Users/elima/Documents/IPB/Thesis/Frames")
points_neo <- read.csv("real_points_1_1.csv", header = TRUE, sep = ";", dec = ",")

#points <- points[c(-1,-2,-57:-61),]
dat2_neo <- points_neo
dat_neo <- dat2_neo
coordinates(dat_neo) <- ~ x_real + y_real
proj4string(dat_neo) <- proj4string(part_neo)
ov_neo <- over(dat_neo, part_neo)
my_df_sf_neo_2 <- filter(my_df_sf_neo, ELEM %in% ov_neo$ELEM)
#ggplotly(p)

# ----------- 4 - Plotagem de gráfico ---------------


# p <- ggplot() +
#   geom_polygon(data=mesh_ogd_2, aes(x = X, y = Y, fill = EPTOY, group = ELEM)) +
```

```
#   scale_fill_gradient( low="white", high="black") +
#   geom_polygon(data=filter(mesh_ogd_2, ELEM %in% ov_neo$ELEM),
#                aes(x = X, y = Y, group = ELEM), fill = "orange")+ coord_fixed() +
#   geom_point(data=points, aes(x=x_real, y=y_real))
# p
#ggplotly(p)

# ---------- 5 - Leitura de dados de DIC ---------------

setwd("C:/Users/elima/Documents/IPB/Thesis/Frames")
test_1_1 <- read.csv("test_1_1.csv", header = TRUE, sep = ",", dec = ".")
test_1_2 <- read.csv("test_1_2.csv", header = TRUE, sep = ",", dec = ".")
points_1_1 <- read.csv("real_points_1_1.csv", header = TRUE, sep = ";", dec = ",")
points_1_2 <- read.csv("real_points_1_2.csv", header = TRUE, sep = ";", dec = ",")
test_1_1_2 <- melt(data=test_1_1, id.vars="Index", measure.vars = colnames(test_1_1[-1]))
test_1_2_2 <- melt(data=test_1_2, id.vars="Index", measure.vars = colnames(test_1_2[-1]))
last_data <- 10
p1 <- ggplot(data = test_1_1_2, aes(x = (Index-1)/2.5, y = value, group = variable, color =
variable)) +
  geom_line() + xlab("Seconds [s]") + ylab("Strain [mm/mm]") +
  geom_line(data = filter(test_1_1_2, Index >= max(test_1_1_2$Index)+1-last_data),
            aes(x = (Index-1)/2.5, y = value, group = variable), color = "red") +
  theme_classic() + theme(legend.position = "none") +
  scale_x_continuous(breaks=seq(0,ceiling(max((test_1_1_2$Index-1)/2.5)),length.out = 6)) +
  scale_y_continuous(breaks=seq(0,ceiling(max(test_1_1_2$value)),length.out = 6))
p1
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_1_points_strain_1.pdf", dpi=300)
#ggplotly(p1)

p2 <- ggplot(data = test_1_2_2, aes(x = (Index-1)/2.5, y = value, group = variable, color =
variable)) +
  geom_line() + xlab("Seconds [s]") + ylab("Strain [mm/mm]") +
  geom_line(data = filter(test_1_2_2, Index >= max(test_1_2_2$Index)+1-last_data),
            aes(x = (Index-1)/2.5, y = value, group = variable), color = "red") +
  theme_classic() + theme(legend.position = "none") +
  scale_x_continuous(breaks=seq(0,ceiling(max((test_1_2_2$Index-1)/2.5)),length.out = 6)) +
  scale_y_continuous(breaks=seq(0,ceiling(max(test_1_2_2$value)),length.out = 6))
p2
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_1_points_strain_2.pdf", dpi=300)
#ggplotly(p2)

test_final_1_1 <- test_1_1[(nrow(test_1_1)-last_data):(nrow(test_1_1)),]
media_1_1 <- cbind(points_1_1,
                   as.data.frame(colMeans(test_final_1_1[,2:ncol(test_final_1_1)])))
media_1_1 <- media_1_1[order(media_1_1$x_real, media_1_1$y_real),]
test_final_1_2 <- test_1_2[(nrow(test_1_2)-last_data):(nrow(test_1_2)),]
media_1_2 <- cbind(points_1_2,
                   as.data.frame(colMeans(test_final_1_2[,2:ncol(test_final_1_2)])))
media_1_2 <- media_1_2[order(media_1_2$x_real, media_1_2$y_real),]

#View(media_1_1)
#View(media_1_2)
media_1_ogd <- as.data.frame(cbind(media_1_1[,2], media_1_1[,3], media_1_1[,4]/100, media_1_2[,4]/
100,
                                   ((media_1_1[,4] + media_1_2[,4])/200), ov_ogd$EPTOY))
media_1_moon <- as.data.frame(cbind(media_1_1[,2], media_1_1[,3], media_1_1[,4]/100, media_1_2[,4]/
100,
                                    ((media_1_1[,4] + media_1_2[,4])/200), ov_moon$EPTOY))
media_1_neo <- as.data.frame(cbind(media_1_1[,2], media_1_1[,3], media_1_1[,4]/100, media_1_2[,4]/
100,
                                    ((media_1_1[,4] + media_1_2[,4])/200), ov_neo$EPTOY))

colnames(media_1_ogd) <- c("x", "y", "media_1_1", "media_1_2", "media", "EPTOY_ELEM")
colnames(media_1_moon) <- c("x", "y", "media_1_1", "media_1_2", "media", "EPTOY_ELEM")
colnames(media_1_neo) <- c("x", "y", "media_1_1", "media_1_2", "media", "EPTOY_ELEM")

EPTOY_2 <- NA
x0 <- NA
```

```
y0 <- NA
erro_1_1 <- NA
erro_1_2 <- NA
erro_1 <- NA

# ---------------- 5.1 - ogden ---------------

nodes <- merge(x = nodes_str_ogd, y = nodes_coord[,1:3], all.x = TRUE, by = "NODE")
pos <- as.matrix(nodes[,c(3,4)])
new.pos_tot <- as.matrix(media_1_ogd[,c("x","y")])
distance <- data.frame()

for(i in 1:nrow(media_1_ogd)){
  new.pos <- new.pos_tot[i,]
  distance <- sqrt((pos[,1] - new.pos[1])^2+(pos[,2] - new.pos[2])^2)
  nearest.idx <- which.min(distance)
  #nearest.idx <- which.min(colSums(sqrt((t(pos) - new.pos)^2)))
  media_1_ogd[i,"EPTOY_NODES"] = nodes[nearest.idx,"EPTOY"]
  media_1_ogd[i,"x0"] = nodes[nearest.idx,"X"]
  media_1_ogd[i,"y0"] = nodes[nearest.idx,"Y"]

}


erro_1_1_elem <- as.data.frame(cbind(100*sqrt((media_1_ogd$media_1_1-media_1_ogd$EPTOY_ELEM)^2)/
media_1_ogd$media_1_1))
erro_1_2_elem <- as.data.frame(cbind(100*sqrt((media_1_ogd$media_1_2-media_1_ogd$EPTOY_ELEM)^2)/
media_1_ogd$media_1_2))
erro_1_elem <- as.data.frame(cbind(100*sqrt((media_1_ogd$media-media_1_ogd$EPTOY_ELEM)^2)/
media_1_ogd$media))
colnames(erro_1_1_elem) <- "error_1_1_elem"
colnames(erro_1_2_elem) <- "error_1_2_elem"
colnames(erro_1_elem) <- "error_1_elem"
erro_1_1_node <- as.data.frame(cbind(100*sqrt((media_1_ogd$media_1_1-media_1_ogd$EPTOY_NODES)^2)/
media_1_ogd$media_1_1))
erro_1_2_node <- as.data.frame(cbind(100*sqrt((media_1_ogd$media_1_2-media_1_ogd$EPTOY_NODES)^2)/
media_1_ogd$media_1_2))
erro_1_node <- as.data.frame(cbind(100*sqrt((media_1_ogd$media-media_1_ogd$EPTOY_NODES)^2)/
media_1_ogd$media))
colnames(erro_1_1_node) <- "error_1_1_node"
colnames(erro_1_2_node) <- "error_1_2_node"
colnames(erro_1_node) <- "error_1_node"

media_1_ogd <- cbind(media_1_ogd, erro_1_1_elem, erro_1_2_elem, erro_1_elem, erro_1_1_node,
erro_1_2_node, erro_1_node)
plot(pos,asp=1)

# ---------------- 5.2 - Mooney-rivlin ---------------

nodes <- merge(x = nodes_str_moon, y = nodes_coord[,1:3], all.x = TRUE, by = "NODE")
pos <- as.matrix(nodes[,c(3,4)])
new.pos_tot <- as.matrix(media_1_moon[,c("x","y")])
distance <- data.frame()

for(i in 1:nrow(media_1_moon)){
  new.pos <- new.pos_tot[i,]
  distance <- sqrt((pos[,1] - new.pos[1])^2+(pos[,2] - new.pos[2])^2)
  nearest.idx <- which.min(distance)
  #nearest.idx <- which.min(colSums(sqrt((t(pos) - new.pos)^2)))
  media_1_moon[i,"EPTOY_NODES"] = nodes[nearest.idx,"EPTOY"]
  media_1_moon[i,"x0"] = nodes[nearest.idx,"X"]
  media_1_moon[i,"y0"] = nodes[nearest.idx,"Y"]

}


erro_1_1_node <- as.data.frame(cbind(100*sqrt((media_1_moon$media_1_1-media_1_moon$EPTOY_NODES)^2)/
media_1_moon$media_1_1))
erro_1_2_node <- as.data.frame(cbind(100*sqrt((media_1_moon$media_1_2-media_1_moon$EPTOY_NODES)^2)/
media_1_moon$media_1_2))
```

```
erro_1_node <- as.data.frame(cbind(100*sqrt((media_1_moon$media-media_1_moon$EPTOY_NODES)^2)/
media_1_moon$media))
colnames(erro_1_1_node) <- "error_1_1_node"
colnames(erro_1_2_node) <- "error_1_2_node"
colnames(erro_1_node) <- "error_1_node"
erro_1_1_elem <- as.data.frame(cbind(100*sqrt((media_1_moon$media_1_1-media_1_moon$EPTOY_ELEM)^2)/
media_1_moon$media_1_1))
erro_1_2_elem <- as.data.frame(cbind(100*sqrt((media_1_moon$media_1_2-media_1_moon$EPTOY_ELEM)^2)/
media_1_moon$media_1_2))
erro_1_elem <- as.data.frame(cbind(100*sqrt((media_1_moon$media-media_1_moon$EPTOY_ELEM)^2)/
media_1_moon$media))
colnames(erro_1_1_elem) <- "error_1_1_elem"
colnames(erro_1_2_elem) <- "error_1_2_elem"
colnames(erro_1_elem) <- "error_1_elem"

media_1_moon <- cbind(media_1_moon, erro_1_1_elem, erro_1_2_elem, erro_1_elem, erro_1_1_node,
erro_1_2_node, erro_1_node)
plot(pos, asp=1)


# ---------------- 5.3 - Neo-hookean ---------------

nodes <- merge(x = nodes_str_neo, y = nodes_coord[,1:3], all.x = TRUE, by = "NODE")
pos <- as.matrix(nodes[,c(3,4)])
new.pos_tot <- as.matrix(media_1_neo[,c("x","y")])
distance <- data.frame()

for(i in 1:nrow(media_1_neo)){
  new.pos <- new.pos_tot[i,]
  distance <- sqrt((pos[,1] - new.pos[1])^2+(pos[,2] - new.pos[2])^2)
  nearest.idx <- which.min(distance)
  #nearest.idx <- which.min(colSums(sqrt((t(pos) - new.pos)^2)))
  media_1_neo[i,"EPTOY_NODES"] = nodes[nearest.idx,"EPTOY"]
  media_1_neo[i,"x0"] = nodes[nearest.idx,"X"]
  media_1_neo[i,"y0"] = nodes[nearest.idx,"Y"]

}


erro_1_1_elem <- as.data.frame(cbind(100*sqrt((media_1_neo$media_1_1-media_1_neo$EPTOY_ELEM)^2)/
media_1_neo$media_1_1))
erro_1_2_elem <- as.data.frame(cbind(100*sqrt((media_1_neo$media_1_2-media_1_neo$EPTOY_ELEM)^2)/
media_1_neo$media_1_2))
erro_1_elem <- as.data.frame(cbind(100*sqrt((media_1_neo$media-media_1_neo$EPTOY_ELEM)^2)/
media_1_neo$media))
colnames(erro_1_1_elem) <- "error_1_1_elem"
colnames(erro_1_2_elem) <- "error_1_2_elem"
colnames(erro_1_elem) <- "error_1_elem"

erro_1_1_node <- as.data.frame(cbind(100*sqrt((media_1_neo$media_1_1-media_1_neo$EPTOY_NODES)^2)/
media_1_neo$media_1_1))
erro_1_2_node <- as.data.frame(cbind(100*sqrt((media_1_neo$media_1_2-media_1_neo$EPTOY_NODES)^2)/
media_1_neo$media_1_2))
erro_1_node <- as.data.frame(cbind(100*sqrt((media_1_neo$media-media_1_neo$EPTOY_NODES)^2)/
media_1_neo$media))
colnames(erro_1_1_node) <- "error_1_1_node"
colnames(erro_1_2_node) <- "error_1_2_node"
colnames(erro_1_node) <- "error_1_node"

media_1_neo <- cbind(media_1_neo, erro_1_1_node, erro_1_2_node, erro_1_node, erro_1_1_elem,
erro_1_2_elem, erro_1_elem)
plot(pos,asp=1)


# ----------- 6 - Plotagem Final ---------------

library(viridis)
library(plotly)

#gp2 <- ggplotly(p2)
```

```
##library(htmlwidgets)
#saveWidget(gp2, "p2.html", selfcontained = F, libdir = "lib")
#widget_file_size(p1)
# ggplotly(p1)
cons_models <- c("Ogden", "Mooney-Rivlyn", "Neo Hookean")
position_models_x <- c(12,42,72)
position_models_y <- c(-5,-5,-5)
label <- as.data.frame(cbind(cons_models, as.numeric(position_models_x),
as.numeric(position_models_y)))
p1 <- ggplot() +
  geom_polygon(data=mesh_ogd_2, aes(x = X, y = Y, fill = EPTOY, group = ELEM)) +
  geom_polygon(data=mesh_moon_2, aes(x = X + 30, y = Y, fill = EPTOY, group = ELEM)) +
  geom_polygon(data=mesh_neo_2, aes(x = X + 60, y = Y, fill = EPTOY, group = ELEM)) +
  scale_fill_gradient(low="white", high="black") + coord_fixed() +
  geom_point(data=media_1_ogd, aes(x=x, y=y, colour=error_1_node), size = 2) +
  geom_point(data=media_1_moon, aes(x=x + 30, y=y, colour=error_1_node), size = 2) +
  geom_point(data=media_1_neo, aes(x=x + 60, y=y, colour=error_1_node), size = 2) +
  scale_color_gradient(low="turquoise1", high="indianred1") +
  geom_point(data=media_1_ogd, aes(x=as.numeric(x0), y=as.numeric(y0)), colour="black", size = 1,
shape = "x") +
  geom_point(data=media_1_moon, aes(x=as.numeric(x0) + 30, y=as.numeric(y0)), colour="black", size
= 1, shape = "x") +
  geom_point(data=media_1_neo, aes(x=as.numeric(x0) + 60, y=as.numeric(y0)), colour="black", size =
1, shape = "x") +
  theme_classic() + theme_void() + labs(fill="ANSYS Strain", colour="Error") +
  geom_text(data=label, aes(x=position_models_x, y=position_models_y, label = cons_models),
hjust=0.5, vjust=0.5)
p1
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_1_comparison_node.pdf", dpi=300)
#gp1 <- ggplotly(p1)
# library(htmlwidgets)
# saveWidget(gp1, "p1.html", selfcontained = F, libdir = "lib")
# widget_file_size(p1)
# ggplotly(p1)

p2 <- ggplot() +
  geom_polygon(data=mesh_ogd_2, aes(x = X, y = Y, fill = EPTOY, group = ELEM)) +
  geom_polygon(data=mesh_moon_2, aes(x = X + 30, y = Y, fill = EPTOY, group = ELEM)) +
  geom_polygon(data=mesh_neo_2, aes(x = X + 60, y = Y, fill = EPTOY, group = ELEM)) +
  scale_fill_gradient(low="white", high="black") + coord_fixed() +
  geom_point(data=media_1_ogd, aes(x=x, y=y, colour=error_1_elem), size = 2) +
  geom_point(data=media_1_moon, aes(x=x + 30, y=y, colour=error_1_elem), size = 2) +
  geom_point(data=media_1_neo, aes(x=x + 60, y=y, colour=error_1_elem), size = 2) +
  scale_color_gradient(low="turquoise1", high="indianred1") +
  geom_point(data=media_1_ogd, aes(x=as.numeric(x0), y=as.numeric(y0)), colour="black", size = 1,
shape = "x") +
  geom_point(data=media_1_moon, aes(x=as.numeric(x0) + 30, y=as.numeric(y0)), colour="black", size
= 1, shape = "x") +
  geom_point(data=media_1_neo, aes(x=as.numeric(x0) + 60, y=as.numeric(y0)), colour="black", size =
1, shape = "x") +
  theme_classic() + theme_void() + labs(fill="ANSYS Strain", colour="Error") +
  geom_text(data=label, aes(x=position_models_x, y=position_models_y, label = cons_models),
hjust=0.5, vjust=0.5)
p2
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_1_comparison_elem.pdf", dpi=300)

export_elem <- as.data.frame(cbind(1:nrow(media_1_ogd), media_1_ogd$x, media_1_ogd$y,
media_1_ogd$media,
                             media_1_ogd$EPTOY_ELEM, media_1_ogd$error_1_elem,
                             media_1_moon$EPTOY_ELEM, media_1_moon$error_1_elem,
                             media_1_neo$EPTOY_ELEM, media_1_neo$error_1_elem))
colnames(export_elem) <- c("point", "point_x", "point_y", "mean",
                      "y_strain_ogden", "O",
                      "y_strain_mooney_riv", "M",
                      "y_strain_neo_hook", "N")
export_2_elem <- melt(data = export_elem[,c(1, 6, 8, 10)],
                  id.vars = "point", measure.vars = colnames(export_elem)[c(6,8,10)])
#export_2_elem <- filter(export_2_elem, point %in% setdiff(export_2_elem$point, c(5, 6, 48, 49)))
```

```
export_3_elem <- data_frame(
  factor = c("O", "M", "N"),
  mean = c(mean(export_elem[,6]), mean(export_elem[,8]), mean(export_elem[,10])),
  sd = c(sd(export_elem[,6]), sd(export_elem[,8]), sd(export_elem[,10]))
)
export_3_elem$factor <- factor(export_3_elem$factor, levels = levels(export_2_elem$variable))


export_node <- as.data.frame(cbind(1:nrow(media_1_ogd), media_1_ogd$x, media_1_ogd$y,
media_1_ogd$media,
                                   media_1_ogd$EPTOY_NODES, media_1_ogd$error_1_node,
                                   media_1_moon$EPTOY_NODES, media_1_moon$error_1_node,
                                   media_1_neo$EPTOY_NODES, media_1_neo$error_1_node))
colnames(export_node) <- c("point", "point_x", "point_y", "mean",
                           "y_strain_ogden", "O",
                           "y_strain_mooney_riv", "M",
                           "y_strain_neo_hook", "N")
export_2_node <- melt(data = export_node[,c(1, 6, 8, 10)],
                      id.vars = "point", measure.vars = colnames(export_node)[c(6,8,10)])
#export_2_node <- filter(export_2_node, point %in% setdiff(export_2_node$point, c(5, 6, 48, 49)))
export_3_node <- data_frame(
  factor = c("O", "M", "N"),
  mean = c(mean(export_node[,6]), mean(export_node[,8]), mean(export_node[,10])),
  sd = c(sd(export_node[,6]), sd(export_node[,8]), sd(export_node[,10]))
)
export_3_node$factor <- factor(export_3_node$factor, levels = levels(export_2_node$variable))
#install.packages("ggpubr")
library(ggpubr)
ggboxplot(export_2_elem, x = "variable", y = "value",
          color = "variable", fill = "lightgray", palette = c("dodgerblue4","green4",
"firebrick1"),
          ylab = "Errors [%]", xlab = "Constitutive Models")
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_1_boxplot_elem.pdf", dpi=300)

ggboxplot(export_2_node, x = "variable", y = "value",
          color = "variable", fill = "lightgray", palette = c("dodgerblue4","green4",
"firebrick1"),
          ylab = "Errors [%]", xlab = "Constitutive Models")
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_1_boxplot_node.pdf", dpi=300)
options(digits=10)

#anova for elem
aov_elem <- aov(value ~ variable, data = export_2_elem)
summary(aov_elem)
# library(gridExtra)
# capture.output(summary(aov_elem),file="test_1_anova.doc")
MSE <- summary(aov_elem)[[1]]["Residuals","Mean Sq"]
ks.test(aov_elem$residuals,"pnorm",0,sqrt(MSE),alternative = "t")
TukeyHSD(aov_elem)
pdf("test_1_tukey_elem.pdf", height=5, width=5)
plot(TukeyHSD(aov_elem), las = 1)# 2. Create a plot
dev.off() # Close the pdf file

#anova for node
aov_node <- aov(value ~ variable, data = export_2_node)
summary(aov_node)
MSE <- summary(aov_node)[[1]]["Residuals","Mean Sq"]
ks.test(aov_node$residuals,"pnorm",0,sqrt(MSE),alternative = "t")
TukeyHSD(aov_node)
pdf("test_1_tukey_node.pdf", height=5, width=5)
plot(TukeyHSD(aov_node), las = 1)# 2. Create a plot
dev.off() # Close the pdf file


p1 <- ggplot() +
  geom_bar(data=export_2_elem, aes(x=point, y=value, fill=factor(variable)),
           position=position_dodge(), stat="identity") +
  ggtitle("Errors and Points") +
```

```
  scale_fill_manual("legend", values = c("dodgerblue4","green4", "firebrick1"))
p1
#ggplotly(p1)


p2 <- ggplot() +
  geom_bar(data=export_2_node, aes(x=point, y=value, fill=factor(variable)),
           position=position_dodge(), stat="identity") +
  ggtitle("Errors and Points") +
  scale_fill_manual("legend", values = c("dodgerblue4","green4", "firebrick1"))
p2
#ggplotly(p2)

q <- ggplot() +
  geom_bar(data=export_3_elem, aes(x=factor, y=mean, fill=factor),
           position=position_dodge(),stat="identity") +
  ggtitle("Errors") +
  scale_fill_manual("legend", values = c("dodgerblue4","green4", "firebrick1"))
q
q <- ggplot() +
  ggtitle("Errors") +
  geom_bar(data=export_3_node, aes(x=factor, y=mean, fill=factor),
           position=position_dodge(),stat="identity") +
  ggtitle("Errors") +
  scale_fill_manual("legend", values = c("dodgerblue4","green4", "firebrick1"))
q
ggplotly(q)




p <- ggplot() +
  geom_bar(data=export_2_elem, aes(x=point, y=value, fill=factor(variable)),
           position=position_dodge(), stat="identity") +
  ggtitle("Errors and Points") +
  scale_fill_manual("legend", values = c("dodgerblue4","green4", "firebrick1"))
ggplotly(p)


q <- ggplot() +
  geom_bar(data=export_3_elem, aes(x=factor, y=mean, fill=factor),
           position=position_dodge(),stat="identity") +
  ggtitle("Errors") +
  scale_fill_manual("legend", values = c("dodgerblue4","green4", "firebrick1"))
q
ggplotly(q)

export1 <- export_elem
export2 <- export_node
colnames(export1) <- colnames(export2) <- c("Points","x","y","Mean DIC Strain",
                                            "Ogden Strain Error", "Ogden Strain",
                                            "Mooney-Rivlyn Strain Error", "Mooney-Rivlyn Strain",
                                            "Neo Hookean Strain Error", "Neo Hookean Strain")
setwd("C:/Users/elima/Documents/IPB/Thesis/Frames")
getwd()

library(gridExtra)
pdf("comparison_values_1_elem.pdf")
grid.table(export1)
dev.off()

library(gridExtra)
pdf("comparison_values_1_nodes.pdf")
grid.table(export2)
dev.off()

write.table(export1, file = "test_1_results_elem.txt", append = FALSE, quote = FALSE, sep = "\t",
dec = ",", row.names = FALSE, col.names = TRUE)
write.table(export2, file = "test_1_results_node.txt", append = FALSE, quote = FALSE, sep = "\t",
dec = ",", row.names = FALSE, col.names = TRUE)
```

```
# \\\\\\\\\\\\\\\\\\\\\\\\\\\\ Test 2 ////////////////////////////

# ----------- 1 - Leitura de dados (ANSYS) ---------------
setwd("C:/Users/elima/Documents/IPB/Thesis/Frames")
getwd()
rm(list=ls(all=TRUE))
library(reshape2)
library(ggplot2)
library(dplyr)
library(viridis)
nodes_coord <- read.csv("nodes_coordinates_2.csv", header = TRUE, sep = ";", dec = ",")
nodes_elem <- read.csv("nodes_elements_2.csv", header = TRUE, sep = ";", dec = ",")
elem_str_ogd <- read.csv("element_strain_ogden_2.csv", header = TRUE, sep = ";", dec = ",")
nodes_str_ogd <- read.csv("nodes_strain_ogden_2.csv", header = TRUE, sep = ";", dec = ",")
elem_str_moon <- read.csv("element_strain_mooney_2.csv", header = TRUE, sep = ";", dec = ",")
nodes_str_moon <- read.csv("nodes_strain_mooney_2.csv", header = TRUE, sep = ";", dec = ",")
elem_str_neo <- read.csv("element_strain_neohook_2.csv", header = TRUE, sep = ";", dec = ",")
nodes_str_neo <- read.csv("nodes_strain_neohook_2.csv", header = TRUE, sep = ";", dec = ",")



# ----------- 2 - Transformação de dados para obtenção de coordenadas dos Elementos (ANSYS)
---------------
# ------------2.1 - OGDEN ---------------

file_ogd_1 <- merge(elem_str_ogd, nodes_elem, all=TRUE, by.x = "ELEM", by.y = "ELEM")
file_ogd_1 <- file_ogd_1[,-3:-7]
for (i in 1:8) {
  file_ogd_1 <- merge(file_ogd_1, nodes_coord[,c(1,2)], all.x=TRUE,
                 by.x = paste("NODE",i, sep = "_", collapse = NULL), by.y = "NODE")
  colnames(file_ogd_1)[ncol(file_ogd_1)] <- paste("NODE",i,"X", sep = "_", collapse = NULL)
}
for (i in 1:8) {
  file_ogd_1 <- merge(file_ogd_1, nodes_coord[,c(1,3)], all.x=TRUE,
                 by.x = paste("NODE",i, sep = "_", collapse = NULL), by.y = "NODE")
  colnames(file_ogd_1)[ncol(file_ogd_1)] <- paste("NODE",i,"Y", sep = "_", collapse = NULL)
}
par(pty="s")
plot(nodes_coord$Y~nodes_coord$X, type="p",pch=".", asp=1)

file_ogd_2 <- file_ogd_1[,-10]
file_ogd_2 <- file_ogd_2[,-1:-8]
file_ogd_2 <- file_ogd_2[,-10:-17]
colnam <- colnames(file_ogd_2)
colnam <- colnam[-1]
file_ogd_2 <- melt(data = file_ogd_2, id.vars = "ELEM", measure.vars = colnam)

file_ogd_3 <- file_ogd_1[,-10]
file_ogd_3 <- file_ogd_3[,-1:-8]
file_ogd_3 <- file_ogd_3[,-2:-9]
colnam <- colnames(file_ogd_3)
colnam <- colnam[-1]
file_ogd_3 <- melt(data = file_ogd_3, id.vars = "ELEM", measure.vars = colnam)
file_ogd_4 <- cbind(file_ogd_2, file_ogd_3)
file_ogd_4 <- file_ogd_4[,-4]
colnames(file_ogd_4)[c(3,5)] <- c("X","Y")
file_ogd_5 <- file_ogd_1[,c("ELEM","EPTOY")]
mesh_ogd <- merge(file_ogd_5,file_ogd_4, by = c("ELEM"))
mesh_ogd <- mesh_ogd[order(mesh_ogd$ELEM,
                           atan2(mesh_ogd$X-mean(mesh_ogd$X),mesh_ogd$Y-mean(mesh_ogd$Y))),]
mesh_ogd_2 <- data.frame()
mesh_ogd_for <- data.frame()
for(i in unique(mesh_ogd$ELEM)){
  mesh_ogd_for <- filter(mesh_ogd, ELEM==i)
  mesh_ogd_for <- mesh_ogd_for[order(atan2(mesh_ogd_for$X-mean(mesh_ogd_for$X),mesh_ogd_for$Y-
mean(mesh_ogd_for$Y))),]
  mesh_ogd_2 <- rbind(mesh_ogd_2, mesh_ogd_for)
}
```

```
# ------------2.2 - Mooney_rivlin ---------------

file_moon_1 <- merge(elem_str_moon, nodes_elem, all=TRUE, by.x = "ELEM", by.y = "ELEM")
file_moon_1 <- file_moon_1[,-3:-7]
for (i in 1:8) {
  file_moon_1 <- merge(file_moon_1, nodes_coord[,c(1,2)], all.x=TRUE,
                       by.x = paste("NODE",i, sep = "_", collapse = NULL), by.y = "NODE")
  colnames(file_moon_1)[ncol(file_moon_1)] <- paste("NODE",i,"X", sep = "_", collapse = NULL)
}
for (i in 1:8) {
  file_moon_1 <- merge(file_moon_1, nodes_coord[,c(1,3)], all.x=TRUE,
                       by.x = paste("NODE",i, sep = "_", collapse = NULL), by.y = "NODE")
  colnames(file_moon_1)[ncol(file_moon_1)] <- paste("NODE",i,"Y", sep = "_", collapse = NULL)
}
par(pty="s")
plot(nodes_coord$Y~nodes_coord$X, type="p",pch=".",asp=1)

file_moon_2 <- file_moon_1[,-10]
file_moon_2 <- file_moon_2[,-1:-8]
file_moon_2 <- file_moon_2[,-10:-17]
colnam <- colnames(file_moon_2)
colnam <- colnam[-1]
file_moon_2 <- melt(data = file_moon_2, id.vars = "ELEM", measure.vars = colnam)

file_moon_3 <- file_moon_1[,-10]
file_moon_3 <- file_moon_3[,-1:-8]
file_moon_3 <- file_moon_3[,-2:-9]
colnam <- colnames(file_moon_3)
colnam <- colnam[-1]
file_moon_3 <- melt(data = file_moon_3, id.vars = "ELEM", measure.vars = colnam)
file_moon_4 <- cbind(file_moon_2, file_moon_3)
file_moon_4 <- file_moon_4[,-4]
colnames(file_moon_4)[c(3,5)] <- c("X","Y")
file_moon_5 <- file_moon_1[,c("ELEM","EPTOY")]
mesh_moon <- merge(file_moon_5,file_moon_4, by = c("ELEM"))
mesh_moon <- mesh_moon[order(mesh_moon$ELEM,
                             atan2(mesh_moon$X-mean(mesh_moon$X),mesh_moon$Y-mean(mesh_moon$Y))),]
mesh_moon_2 <- data.frame()
mesh_moon_for <- data.frame()
for(i in unique(mesh_moon$ELEM)){
  mesh_moon_for <- filter(mesh_moon, ELEM==i)
  mesh_moon_for <- mesh_moon_for[order(atan2(mesh_moon_for$X-mean(mesh_moon_for$X),mesh_moon_for$Y-
mean(mesh_moon_for$Y))),]
  mesh_moon_2 <- rbind(mesh_moon_2, mesh_moon_for)
}

# ------------2.3 - NEO_hookean ---------------

file_neo_1 <- merge(elem_str_neo, nodes_elem, all=TRUE, by.x = "ELEM", by.y = "ELEM")
file_neo_1 <- file_neo_1[,-3:-7]
for (i in 1:8) {
  file_neo_1 <- merge(file_neo_1, nodes_coord[,c(1,2)], all.x=TRUE,
                       by.x = paste("NODE",i, sep = "_", collapse = NULL), by.y = "NODE")
  colnames(file_neo_1)[ncol(file_neo_1)] <- paste("NODE",i,"X", sep = "_", collapse = NULL)
}
for (i in 1:8) {
  file_neo_1 <- merge(file_neo_1, nodes_coord[,c(1,3)], all.x=TRUE,
                       by.x = paste("NODE",i, sep = "_", collapse = NULL), by.y = "NODE")
  colnames(file_neo_1)[ncol(file_neo_1)] <- paste("NODE",i,"Y", sep = "_", collapse = NULL)
}
par(pty="s")
plot(nodes_coord$Y~nodes_coord$X, type="p",pch=".",asp=1)

file_neo_2 <- file_neo_1[,-10]
file_neo_2 <- file_neo_2[,-1:-8]
file_neo_2 <- file_neo_2[,-10:-17]
colnam <- colnames(file_neo_2)
colnam <- colnam[-1]
file_neo_2 <- melt(data = file_neo_2, id.vars = "ELEM", measure.vars = colnam)
```

```
file_neo_3 <- file_neo_1[,-10]
file_neo_3 <- file_neo_3[,-1:-8]
file_neo_3 <- file_neo_3[,-2:-9]
colnam <- colnames(file_neo_3)
colnam <- colnam[-1]
file_neo_3 <- melt(data = file_neo_3, id.vars = "ELEM", measure.vars = colnam)
file_neo_4 <- cbind(file_neo_2, file_neo_3)
file_neo_4 <- file_neo_4[,-4]
colnames(file_neo_4)[c(3,5)] <- c("X","Y")
file_neo_5 <- file_neo_1[,c("ELEM","EPTOY")]
mesh_neo <- merge(file_neo_5,file_neo_4, by = c("ELEM"))
mesh_neo <- mesh_neo[order(mesh_neo$ELEM,
                           atan2(mesh_neo$X-mean(mesh_neo$X),mesh_neo$Y-mean(mesh_neo$Y))),]
mesh_neo_2 <- data.frame()
mesh_neo_for <- data.frame()
for(i in unique(mesh_neo$ELEM)){
  mesh_neo_for <- filter(mesh_neo, ELEM==i)
  mesh_neo_for <- mesh_neo_for[order(atan2(mesh_neo_for$X-mean(mesh_neo_for$X),mesh_neo_for$Y-
mean(mesh_neo_for$Y))),]
  mesh_neo_2 <- rbind(mesh_neo_2, mesh_neo_for)
}

# ----------- 3 - Conversão de Polígonos Para identificação de pontos ---------------

# ----------- 3.1 - OGDEN -----------------

#ggplotly(p)
#install.packages("sfheaders")
library(sf)
library(sfheaders)

my_df_sf_ogd <- st_as_sf(mesh_ogd,
                         coords = c('X', 'Y')) %>%
  st_set_crs(26918) %>%
  group_by(ELEM) %>%
  summarise() %>%
  ungroup() %>%  # Just in case
  st_convex_hull()
my_df_sf_ogd <- merge(my_df_sf_ogd,file_ogd_5, by = c("ELEM"))

#install.packages("prevR")
library(prevR)
library(sp)
library(rgeos)
library(sp)
library(rgdal)
library(plotly)
part_ogd <- as(my_df_sf_ogd, 'Spatial')

setwd("C:/Users/elima/Documents/IPB/Thesis/Frames")
points_ogd <- read.csv("real_points_2_1.csv", header = TRUE, sep = ";", dec = ",")

#points <- points[c(-1,-2,-57:-61),]
dat2_ogd <- points_ogd
dat_ogd <- dat2_ogd
coordinates(dat_ogd) <- ~ x_real + y_real
proj4string(dat_ogd) <- proj4string(part_ogd)
ov_ogd <- over(dat_ogd, part_ogd)
my_df_sf_ogd_2 <- filter(my_df_sf_ogd, ELEM %in% ov_ogd$ELEM)


# ----------- 3.2 - Mooney-rivlin -----------------

#ggplotly(p)
#install.packages("sfheaders")
library(sf)
library(sfheaders)

my_df_sf_moon <- st_as_sf(mesh_moon,
                          coords = c('X', 'Y')) %>%
```

```
  st_set_crs(26918) %>%
  group_by(ELEM) %>%
  summarise() %>%
  ungroup() %>%  # Just in case
  st_convex_hull()
my_df_sf_moon <- merge(my_df_sf_moon,file_moon_5, by = c("ELEM"))

#install.packages("prevR")
library(prevR)
library(sp)
library(rgeos)
library(sp)
library(rgdal)
library(plotly)
part_moon <- as(my_df_sf_moon, 'Spatial')

setwd("C:/Users/elima/Documents/IPB/Thesis/Frames")
points_moon <- read.csv("real_points_2_1.csv", header = TRUE, sep = ";", dec = ",")

#points <- points[c(-1,-2,-57:-61),]
dat2_moon <- points_moon
dat_moon <- dat2_moon
coordinates(dat_moon) <- ~ x_real + y_real
proj4string(dat_moon) <- proj4string(part_moon)
ov_moon <- over(dat_moon, part_moon)
my_df_sf_moon_2 <- filter(my_df_sf_moon, ELEM %in% ov_moon$ELEM)
#ggplotly(p)

# ----------- 3.3 - NEO-hookean -----------------

#ggplotly(p)
#install.packages("sfheaders")
library(sf)
library(sfheaders)

my_df_sf_neo <- st_as_sf(mesh_neo,
                         coords = c('X', 'Y')) %>%
  st_set_crs(26918) %>%
  group_by(ELEM) %>%
  summarise() %>%
  ungroup() %>%  # Just in case
  st_convex_hull()
my_df_sf_neo <- merge(my_df_sf_neo,file_neo_5, by = c("ELEM"))

#install.packages("prevR")
library(prevR)
library(sp)
library(rgeos)
library(sp)
library(rgdal)
library(plotly)
part_neo <- as(my_df_sf_neo, 'Spatial')

setwd("C:/Users/elima/Documents/IPB/Thesis/Frames")
points_neo <- read.csv("real_points_2_1.csv", header = TRUE, sep = ";", dec = ",")

#points <- points[c(-1,-2,-57:-61),]
dat2_neo <- points_neo
dat_neo <- dat2_neo
coordinates(dat_neo) <- ~ x_real + y_real
proj4string(dat_neo) <- proj4string(part_neo)
ov_neo <- over(dat_neo, part_neo)
my_df_sf_neo_2 <- filter(my_df_sf_neo, ELEM %in% ov_neo$ELEM)
#ggplotly(p)

# ----------- 4 - Plotagem de gráfico ---------------


# p <- ggplot() +
#   geom_polygon(data=mesh_ogd_2, aes(x = X, y = Y, fill = EPTOY, group = ELEM)) +
```

```
#   scale_fill_gradient( low="white", high="black") +
#   geom_polygon(data=filter(mesh_ogd_2, ELEM %in% ov_neo$ELEM),
#                aes(x = X, y = Y, group = ELEM), fill = "orange")+ coord_fixed() +
#   geom_point(data=points, aes(x=x_real, y=y_real))
# p
#ggplotly(p)

# ---------- 5 - Leitura de dados de DIC ---------------

setwd("C:/Users/elima/Documents/IPB/Thesis/Frames")
test_1_1 <- read.csv("test_2_1.csv", header = TRUE, sep = ",", dec = ".")
test_1_2 <- read.csv("test_2_2.csv", header = TRUE, sep = ",", dec = ".")
test_1_3 <- read.csv("test_2_3.csv", header = TRUE, sep = ",", dec = ".")
points_1_1 <- read.csv("real_points_2_1.csv", header = TRUE, sep = ";", dec = ",")
points_1_2 <- read.csv("real_points_2_2.csv", header = TRUE, sep = ";", dec = ",")
points_1_3 <- read.csv("real_points_2_3.csv", header = TRUE, sep = ";", dec = ",")
test_1_1_2 <- melt(data=test_1_1, id.vars="Index", measure.vars = colnames(test_1_1[-1]))
test_1_2_2 <- melt(data=test_1_2, id.vars="Index", measure.vars = colnames(test_1_2[-1]))
test_1_3_2 <- melt(data=test_1_3, id.vars="Index", measure.vars = colnames(test_1_3[-1]))

last_data <- 10
p1 <- ggplot(data = test_1_1_2, aes(x = (Index-1)/2.5, y = value, group = variable, color =
variable)) +
  geom_line() + xlab("Seconds [s]") + ylab("Strain [mm/mm]") +
  geom_line(data = filter(test_1_1_2, Index >= max(test_1_1_2$Index)+1-last_data),
            aes(x = (Index-1)/2.5, y = value, group = variable), color = "red") +
  theme_classic() + theme(legend.position = "none") +
  scale_x_continuous(breaks=seq(0,ceiling(max((test_1_1_2$Index-1)/2.5)),length.out = 6)) +
  scale_y_continuous(breaks=seq(0,ceiling(max(test_1_1_2$value)),length.out = 6))
p1
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_2_points_strain_1.pdf", dpi=300)
#ggplotly(p1)

p2 <- ggplot(data = test_1_2_2, aes(x = (Index-1)/2.5, y = value, group = variable, color =
variable)) +
  geom_line() + xlab("Seconds [s]") + ylab("Strain [mm/mm]") +
  geom_line(data = filter(test_1_2_2, Index >= max(test_1_2_2$Index)+1-last_data),
            aes(x = (Index-1)/2.5, y = value, group = variable), color = "red") +
  theme_classic() + theme(legend.position = "none") +
  scale_x_continuous(breaks=seq(0,ceiling(max((test_1_2_2$Index-1)/2.5)),length.out = 6)) +
  scale_y_continuous(breaks=seq(0,ceiling(max(test_1_2_2$value)),length.out = 6))
p2
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_2_points_strain_2.pdf", dpi=300)
#ggplotly(p2)

p3 <- ggplot(data = test_1_3_2, aes(x = (Index-1)/2.5, y = value, group = variable, color =
variable)) +
  geom_line() + xlab("Seconds [s]") + ylab("Strain [mm/mm]") +
  geom_line(data = filter(test_1_3_2, Index >= max(test_1_3_2$Index)+1-last_data),
            aes(x = (Index-1)/2.5, y = value, group = variable), color = "red") +
  theme_classic() + theme(legend.position = "none") +
  scale_x_continuous(breaks=seq(0,ceiling(max((test_1_3_2$Index-1)/2.5)),length.out = 6)) +
  scale_y_continuous(breaks=seq(0,ceiling(max(test_1_3_2$value)),length.out = 6))
p3
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_2_points_strain_3.pdf", dpi=300)
#ggplotly(p3)

last_data <- 10
test_final_1_1 <- test_1_1[(nrow(test_1_1)-last_data):(nrow(test_1_1)),]
media_1_1 <- cbind(points_1_1,
                   as.data.frame(colMeans(test_final_1_1[,2:ncol(test_final_1_1)])))
media_1_1 <- media_1_1[order(media_1_1$x_real, media_1_1$y_real),]

test_final_1_2 <- test_1_2[(nrow(test_1_2)-last_data):(nrow(test_1_2)),]
media_1_2 <- cbind(points_1_2,
                   as.data.frame(colMeans(test_final_1_2[,2:ncol(test_final_1_2)])))
media_1_2 <- media_1_2[order(media_1_2$x_real, media_1_2$y_real),]
```

```
test_final_1_3 <- test_1_3[(nrow(test_1_3)-last_data):(nrow(test_1_3)),]
media_1_3 <- cbind(points_1_3,
                   as.data.frame(colMeans(test_final_1_3[,2:ncol(test_final_1_3)])))
media_1_3 <- media_1_3[order(media_1_3$x_real, media_1_3$y_real),]



#View(media_1_1)
#View(media_1_2)
media_1_ogd <- as.data.frame(cbind(media_1_1[,2], media_1_1[,3],
                                   media_1_1[,4]/100, media_1_2[,4]/100, media_1_3[,4]/100,
                                   ((media_1_1[,4] + media_1_2[,4] + media_1_3[,4])/300),
ov_ogd$EPTOY))
media_1_moon <- as.data.frame(cbind(media_1_1[,2], media_1_1[,3],
                                    media_1_1[,4]/100, media_1_2[,4]/100, media_1_3[,4]/100,
                                   ((media_1_1[,4] + media_1_2[,4] + media_1_3[,4])/300),
ov_moon$EPTOY))
media_1_neo <- as.data.frame(cbind(media_1_1[,2], media_1_1[,3],
                                   media_1_1[,4]/100, media_1_2[,4]/100, media_1_3[,4]/100,
                                   ((media_1_1[,4] + media_1_2[,4] + media_1_3[,4])/300),
ov_neo$EPTOY))

colnames(media_1_ogd) <- c("x", "y", "media_1_1", "media_1_2", "media_1_3", "media", "EPTOY_ELEM")
colnames(media_1_moon) <- c("x", "y", "media_1_1", "media_1_2", "media_1_3", "media", "EPTOY_ELEM")
colnames(media_1_neo) <- c("x", "y", "media_1_1", "media_1_2", "media_1_3", "media", "EPTOY_ELEM")

EPTOY_2 <- NA
x0 <- NA
y0 <- NA
erro_1_1 <- NA
erro_1_2 <- NA
erro_1 <- NA

# ---------------- 5.1 - ogden ---------------

nodes <- merge(x = nodes_str_ogd, y = nodes_coord[,1:3], all.x = TRUE, by = "NODE")
pos <- as.matrix(nodes[,c(3,4)])
new.pos_tot <- as.matrix(media_1_ogd[,c("x","y")])
distance <- data.frame()

for(i in 1:nrow(media_1_ogd)){
  new.pos <- new.pos_tot[i,]
  distance <- sqrt((pos[,1] - new.pos[1])^2+(pos[,2] - new.pos[2])^2)
  nearest.idx <- which.min(distance)
  #nearest.idx <- which.min(colSums(sqrt((t(pos) - new.pos)^2)))
  media_1_ogd[i,"EPTOY_NODES"] = nodes[nearest.idx,"EPTOY"]
  media_1_ogd[i,"x0"] = nodes[nearest.idx,"X"]
  media_1_ogd[i,"y0"] = nodes[nearest.idx,"Y"]

}


erro_1_1_elem <- as.data.frame(cbind(100*sqrt((media_1_ogd$media_1_1-media_1_ogd$EPTOY_ELEM)^2)/
media_1_ogd$media_1_1))
erro_1_2_elem <- as.data.frame(cbind(100*sqrt((media_1_ogd$media_1_2-media_1_ogd$EPTOY_ELEM)^2)/
media_1_ogd$media_1_2))
erro_1_3_elem <- as.data.frame(cbind(100*sqrt((media_1_ogd$media_1_3-media_1_ogd$EPTOY_ELEM)^2)/
media_1_ogd$media_1_3))
erro_1_elem <- as.data.frame(cbind(100*sqrt((media_1_ogd$media-media_1_ogd$EPTOY_ELEM)^2)/
media_1_ogd$media))
colnames(erro_1_1_elem) <- "error_1_1_elem"
colnames(erro_1_2_elem) <- "error_1_2_elem"
colnames(erro_1_3_elem) <- "error_1_3_elem"
colnames(erro_1_elem) <- "error_1_elem"
erro_1_1_node <- as.data.frame(cbind(100*sqrt((media_1_ogd$media_1_1-media_1_ogd$EPTOY_NODES)^2)/
media_1_ogd$media_1_1))
erro_1_2_node <- as.data.frame(cbind(100*sqrt((media_1_ogd$media_1_2-media_1_ogd$EPTOY_NODES)^2)/
media_1_ogd$media_1_2))
```

```
erro_1_3_node <- as.data.frame(cbind(100*sqrt((media_1_ogd$media_1_3-media_1_ogd$EPTOY_NODES)^2)/
media_1_ogd$media_1_3))
erro_1_node <- as.data.frame(cbind(100*sqrt((media_1_ogd$media-media_1_ogd$EPTOY_NODES)^2)/
media_1_ogd$media))
colnames(erro_1_1_node) <- "error_1_1_node"
colnames(erro_1_2_node) <- "error_1_2_node"
colnames(erro_1_3_node) <- "error_1_3_node"
colnames(erro_1_node) <- "error_1_node"

media_1_ogd <- cbind(media_1_ogd, erro_1_1_elem, erro_1_2_elem, erro_1_3_elem, erro_1_elem,
erro_1_1_node, erro_1_2_node, erro_1_3_node, erro_1_node)
plot(pos,asp=1)

# ---------------- 5.2 - Mooney-rivlin ---------------

nodes <- merge(x = nodes_str_moon, y = nodes_coord[,1:3], all.x = TRUE, by = "NODE")
pos <- as.matrix(nodes[,c(3,4)])
new.pos_tot <- as.matrix(media_1_moon[,c("x","y")])
distance <- data.frame()

for(i in 1:nrow(media_1_moon)){
  new.pos <- new.pos_tot[i,]
  distance <- sqrt((pos[,1] - new.pos[1])^2+(pos[,2] - new.pos[2])^2)
  nearest.idx <- which.min(distance)
  #nearest.idx <- which.min(colSums(sqrt((t(pos) - new.pos)^2)))
  media_1_moon[i,"EPTOY_NODES"] = nodes[nearest.idx,"EPTOY"]
  media_1_moon[i,"x0"] = nodes[nearest.idx,"X"]
  media_1_moon[i,"y0"] = nodes[nearest.idx,"Y"]

}


erro_1_1_node <- as.data.frame(cbind(100*sqrt((media_1_moon$media_1_1-media_1_moon$EPTOY_NODES)^2)/
media_1_moon$media_1_1))
erro_1_2_node <- as.data.frame(cbind(100*sqrt((media_1_moon$media_1_2-media_1_moon$EPTOY_NODES)^2)/
media_1_moon$media_1_2))
erro_1_3_node <- as.data.frame(cbind(100*sqrt((media_1_moon$media_1_3-media_1_moon$EPTOY_NODES)^2)/
media_1_moon$media_1_3))
erro_1_node <- as.data.frame(cbind(100*sqrt((media_1_moon$media-media_1_moon$EPTOY_NODES)^2)/
media_1_moon$media))
colnames(erro_1_1_node) <- "error_1_1_node"
colnames(erro_1_2_node) <- "error_1_2_node"
colnames(erro_1_3_node) <- "error_1_3_node"
colnames(erro_1_node) <- "error_1_node"
erro_1_1_elem <- as.data.frame(cbind(100*sqrt((media_1_moon$media_1_1-media_1_moon$EPTOY_ELEM)^2)/
media_1_moon$media_1_1))
erro_1_2_elem <- as.data.frame(cbind(100*sqrt((media_1_moon$media_1_2-media_1_moon$EPTOY_ELEM)^2)/
media_1_moon$media_1_2))
erro_1_3_elem <- as.data.frame(cbind(100*sqrt((media_1_moon$media_1_3-media_1_moon$EPTOY_ELEM)^2)/
media_1_moon$media_1_3))
erro_1_elem <- as.data.frame(cbind(100*sqrt((media_1_moon$media-media_1_moon$EPTOY_ELEM)^2)/
media_1_moon$media))
colnames(erro_1_1_elem) <- "error_1_1_elem"
colnames(erro_1_2_elem) <- "error_1_2_elem"
colnames(erro_1_3_elem) <- "error_1_3_elem"
colnames(erro_1_elem) <- "error_1_elem"

media_1_moon <- cbind(media_1_moon, erro_1_1_elem, erro_1_2_elem, erro_1_3_elem, erro_1_elem,
                      erro_1_1_node, erro_1_2_node, erro_1_3_node, erro_1_node)
plot(pos, asp=1)


# ---------------- 5.3 - Neo-hookean ---------------

nodes <- merge(x = nodes_str_neo, y = nodes_coord[,1:3], all.x = TRUE, by = "NODE")
pos <- as.matrix(nodes[,c(3,4)])
new.pos_tot <- as.matrix(media_1_neo[,c("x","y")])
distance <- data.frame()

for(i in 1:nrow(media_1_neo)){
```

```
  new.pos <- new.pos_tot[i,]
  distance <- sqrt((pos[,1] - new.pos[1])^2+(pos[,2] - new.pos[2])^2)
  nearest.idx <- which.min(distance)
  #nearest.idx <- which.min(colSums(sqrt((t(pos) - new.pos)^2)))
  media_1_neo[i,"EPTOY_NODES"] = nodes[nearest.idx,"EPTOY"]
  media_1_neo[i,"x0"] = nodes[nearest.idx,"X"]
  media_1_neo[i,"y0"] = nodes[nearest.idx,"Y"]

}


erro_1_1_elem <- as.data.frame(cbind(100*sqrt((media_1_neo$media_1_1-media_1_neo$EPTOY_ELEM)^2)/
media_1_neo$media_1_1))
erro_1_2_elem <- as.data.frame(cbind(100*sqrt((media_1_neo$media_1_2-media_1_neo$EPTOY_ELEM)^2)/
media_1_neo$media_1_2))
erro_1_3_elem <- as.data.frame(cbind(100*sqrt((media_1_neo$media_1_3-media_1_neo$EPTOY_ELEM)^2)/
media_1_neo$media_1_3))
erro_1_elem <- as.data.frame(cbind(100*sqrt((media_1_neo$media-media_1_neo$EPTOY_ELEM)^2)/
media_1_neo$media))
colnames(erro_1_1_elem) <- "error_1_1_elem"
colnames(erro_1_2_elem) <- "error_1_2_elem"
colnames(erro_1_3_elem) <- "error_1_3_elem"
colnames(erro_1_elem) <- "error_1_elem"

erro_1_1_node <- as.data.frame(cbind(100*sqrt((media_1_neo$media_1_1-media_1_neo$EPTOY_NODES)^2)/
media_1_neo$media_1_1))
erro_1_2_node <- as.data.frame(cbind(100*sqrt((media_1_neo$media_1_2-media_1_neo$EPTOY_NODES)^2)/
media_1_neo$media_1_2))
erro_1_3_node <- as.data.frame(cbind(100*sqrt((media_1_neo$media_1_3-media_1_neo$EPTOY_NODES)^2)/
media_1_neo$media_1_3))
erro_1_node <- as.data.frame(cbind(100*sqrt((media_1_neo$media-media_1_neo$EPTOY_NODES)^2)/
media_1_neo$media))
colnames(erro_1_1_node) <- "error_1_1_node"
colnames(erro_1_2_node) <- "error_1_2_node"
colnames(erro_1_3_node) <- "error_1_3_node"
colnames(erro_1_node) <- "error_1_node"

media_1_neo <- cbind(media_1_neo, erro_1_1_node, erro_1_2_node, erro_1_3_node, erro_1_node,
                     erro_1_1_elem, erro_1_2_elem, erro_1_3_elem, erro_1_elem)
plot(pos,asp=1)


# ----------- 6 - Plotagem Final ---------------

library(viridis)
library(plotly)

#gp2 <- ggplotly(p2)
##library(htmlwidgets)
#saveWidget(gp2, "p2.html", selfcontained = F, libdir = "lib")
#widget_file_size(p1)
# ggplotly(p1)
cons_models <- c("Ogden", "Mooney-Rivlyn", "Neo Hookean")
position_models_x <- c(12,42,72)
position_models_y <- c(-5,-5,-5)
label <- as.data.frame(cbind(cons_models, as.numeric(position_models_x),
as.numeric(position_models_y)))
p1 <- ggplot() +
  geom_polygon(data=mesh_ogd_2, aes(x = X, y = Y, fill = EPTOY, group = ELEM)) +
  geom_polygon(data=mesh_moon_2, aes(x = X + 30, y = Y, fill = EPTOY, group = ELEM)) +
  geom_polygon(data=mesh_neo_2, aes(x = X + 60, y = Y, fill = EPTOY, group = ELEM)) +
  scale_fill_gradient(low="white", high="black") + coord_fixed() +
  geom_point(data=media_1_ogd, aes(x=x, y=y, colour=error_1_node), size = 2) +
  geom_point(data=media_1_moon, aes(x=x + 30, y=y, colour=error_1_node), size = 2) +
  geom_point(data=media_1_neo, aes(x=x + 60, y=y, colour=error_1_node), size = 2) +
  scale_color_gradient(low="turquoise1", high="indianred1") +
  geom_point(data=media_1_ogd, aes(x=as.numeric(x0), y=as.numeric(y0)), colour="black", size = 1,
shape = "x") +
  geom_point(data=media_1_moon, aes(x=as.numeric(x0) + 30, y=as.numeric(y0)), colour="black", size
= 1, shape = "x") +
```

```
  geom_point(data=media_1_neo, aes(x=as.numeric(x0) + 60, y=as.numeric(y0)), colour="black", size =
1, shape = "x") +
  theme_classic() + theme_void() + labs(fill="ANSYS Strain", colour="Error") +
  geom_text(data=label, aes(x=position_models_x, y=position_models_y, label = cons_models),
hjust=0.5, vjust=0.5)
p1
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_2_comparison_node.pdf", dpi=300)
#gp1 <- ggplotly(p1)
# library(htmlwidgets)
# saveWidget(gp1, "p1.html", selfcontained = F, libdir = "lib")
# widget_file_size(p1)
# ggplotly(p1)

p2 <- ggplot() +
  geom_polygon(data=mesh_ogd_2, aes(x = X, y = Y, fill = EPTOY, group = ELEM)) +
  geom_polygon(data=mesh_moon_2, aes(x = X + 30, y = Y, fill = EPTOY, group = ELEM)) +
  geom_polygon(data=mesh_neo_2, aes(x = X + 60, y = Y, fill = EPTOY, group = ELEM)) +
  scale_fill_gradient(low="white", high="black") + coord_fixed() +
  geom_point(data=media_1_ogd, aes(x=x, y=y, colour=error_1_elem), size = 2) +
  geom_point(data=media_1_moon, aes(x=x + 30, y=y, colour=error_1_elem), size = 2) +
  geom_point(data=media_1_neo, aes(x=x + 60, y=y, colour=error_1_elem), size = 2) +
  scale_color_gradient(low="turquoise1", high="indianred1") +
  geom_point(data=media_1_ogd, aes(x=as.numeric(x0), y=as.numeric(y0)), colour="black", size = 1,
shape = "x") +
  geom_point(data=media_1_moon, aes(x=as.numeric(x0) + 30, y=as.numeric(y0)), colour="black", size
= 1, shape = "x") +
  geom_point(data=media_1_neo, aes(x=as.numeric(x0) + 60, y=as.numeric(y0)), colour="black", size =
1, shape = "x") +
  theme_classic() + theme_void() + labs(fill="ANSYS Strain", colour="Error") +
  geom_text(data=label, aes(x=position_models_x, y=position_models_y, label = cons_models),
hjust=0.5, vjust=0.5)
p2
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_2_comparison_elem.pdf", dpi=300)

export_elem <- as.data.frame(cbind(1:nrow(media_1_ogd), media_1_ogd$x, media_1_ogd$y,
media_1_ogd$media,
                                   media_1_ogd$EPTOY_ELEM, media_1_ogd$error_1_elem,
                                   media_1_moon$EPTOY_ELEM, media_1_moon$error_1_elem,
                                   media_1_neo$EPTOY_ELEM, media_1_neo$error_1_elem))
colnames(export_elem) <- c("point", "point_x", "point_y", "mean",
                           "y_strain_ogden", "O",
                           "y_strain_mooney_riv", "M",
                           "y_strain_neo_hook", "N")
export_2_elem <- melt(data = export_elem[,c(1, 6, 8, 10)],
                      id.vars = "point", measure.vars = colnames(export_elem)[c(6,8,10)])
#export_2_elem <- filter(export_2_elem, point %in% setdiff(export_2_elem$point, c(5, 6, 48, 49)))
export_3_elem <- data_frame(
  factor = c("O", "M", "N"),
  mean = c(mean(export_elem[,6]), mean(export_elem[,8]), mean(export_elem[,10])),
  sd = c(sd(export_elem[,6]), sd(export_elem[,8]), sd(export_elem[,10]))
)
export_3_elem$factor <- factor(export_3_elem$factor, levels = levels(export_2_elem$variable))


export_node <- as.data.frame(cbind(1:nrow(media_1_ogd), media_1_ogd$x, media_1_ogd$y,
media_1_ogd$media,
                                   media_1_ogd$EPTOY_NODES, media_1_ogd$error_1_node,
                                   media_1_moon$EPTOY_NODES, media_1_moon$error_1_node,
                                   media_1_neo$EPTOY_NODES, media_1_neo$error_1_node))
colnames(export_node) <- c("point", "point_x", "point_y", "mean",
                           "y_strain_ogden", "O",
                           "y_strain_mooney_riv", "M",
                           "y_strain_neo_hook", "N")
export_2_node <- melt(data = export_node[,c(1, 6, 8, 10)],
                      id.vars = "point", measure.vars = colnames(export_node)[c(6,8,10)])
#export_2_node <- filter(export_2_node, point %in% setdiff(export_2_node$point, c(5, 6, 48, 49)))
export_3_node <- data_frame(
  factor = c("O", "M", "N"),
```

```
  mean = c(mean(export_node[,6]), mean(export_node[,8]), mean(export_node[,10])),
  sd = c(sd(export_node[,6]), sd(export_node[,8]), sd(export_node[,10]))
)
export_3_node$factor <- factor(export_3_node$factor, levels = levels(export_2_node$variable))
#install.packages("ggpubr")
library(ggpubr)
ggboxplot(export_2_elem, x = "variable", y = "value",
          color = "variable", fill = "lightgray", palette = c("dodgerblue4","green4",
"firebrick1"),
          ylab = "Errors [%]", xlab = "Constitutive Models")
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_2_boxplot_elem.pdf", dpi=300)

ggboxplot(export_2_node, x = "variable", y = "value",
          color = "variable", fill = "lightgray", palette = c("dodgerblue4","green4",
"firebrick1"),
          ylab = "Errors [%]", xlab = "Constitutive Models")
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_2_boxplot_node.pdf", dpi=300)
options(digits=10)

#anova for elem
aov_elem <- aov(value ~ variable, data = export_2_elem)
summary(aov_elem)
# library(gridExtra)
# capture.output(summary(aov_elem),file="test_1_anova.doc")
MSE <- summary(aov_elem)[[1]]["Residuals","Mean Sq"]
ks.test(aov_elem$residuals,"pnorm",0,sqrt(MSE),alternative = "t")
TukeyHSD(aov_elem)
pdf("test_2_tukey_elem.pdf", height=5, width=5)
plot(TukeyHSD(aov_elem), las = 1)# 2. Create a plot
dev.off() # Close the pdf file

#anova for node
aov_node <- aov(value ~ variable, data = export_2_node)
summary(aov_node)
MSE <- summary(aov_node)[[1]]["Residuals","Mean Sq"]
ks.test(aov_node$residuals,"pnorm",0,sqrt(MSE),alternative = "t")
TukeyHSD(aov_node)
pdf("test_2_tukey_node.pdf", height=5, width=5)
plot(TukeyHSD(aov_node), las = 1)# 2. Create a plot
dev.off() # Close the pdf file


p1 <- ggplot() +
  geom_bar(data=export_2_elem, aes(x=point, y=value, fill=factor(variable)),
           position=position_dodge(), stat="identity") +
  ggtitle("Errors and Points") +
  scale_fill_manual("legend", values = c("dodgerblue4","green4", "firebrick1"))
p1
#ggplotly(p1)


p2 <- ggplot() +
  geom_bar(data=export_2_node, aes(x=point, y=value, fill=factor(variable)),
           position=position_dodge(), stat="identity") +
  ggtitle("Errors and Points") +
  scale_fill_manual("legend", values = c("dodgerblue4","green4", "firebrick1"))
p2
#ggplotly(p2)

q <- ggplot() +
  geom_bar(data=export_3_elem, aes(x=factor, y=mean, fill=factor),
           position=position_dodge(),stat="identity") +
  ggtitle("Errors") +
  scale_fill_manual("legend", values = c("dodgerblue4","green4", "firebrick1"))
q
q <- ggplot() +
  ggtitle("Errors") +
  geom_bar(data=export_3_node, aes(x=factor, y=mean, fill=factor),
```

```
          position=position_dodge(),stat="identity") +
  ggtitle("Errors") +
  scale_fill_manual("legend", values = c("dodgerblue4","green4", "firebrick1"))
q
ggplotly(q)



p <- ggplot() +
  geom_bar(data=export_2_elem, aes(x=point, y=value, fill=factor(variable)),
          position=position_dodge(), stat="identity") +
  ggtitle("Errors and Points") +
  scale_fill_manual("legend", values = c("dodgerblue4","green4", "firebrick1"))
ggplotly(p)


q <- ggplot() +
  geom_bar(data=export_3_elem, aes(x=factor, y=mean, fill=factor),
          position=position_dodge(),stat="identity") +
  ggtitle("Errors") +
  scale_fill_manual("legend", values = c("dodgerblue4","green4", "firebrick1"))
q
ggplotly(q)

export1 <- export_elem
export2 <- export_node
colnames(export1) <- colnames(export2) <- c("Points","x","y","Mean DIC Strain",
                                            "Ogden Strain", "Ogden Strain Error",
                                            "Mooney-Rivlyn Strain", "Mooney-Rivlyn Strain Error",
                                            "Neo Hookean Strain", "Neo Hookean Strain Error")
setwd("C:/Users/elima/Documents/IPB/Thesis/Frames")
getwd()
write.table(export1, file = "test_2_results_elem.txt", append = FALSE, quote = FALSE, sep = "\t",
dec = ",", row.names = FALSE, col.names = TRUE)
write.table(export2, file = "test_2_results_node.txt", append = FALSE, quote = FALSE, sep = "\t",
dec = ",", row.names = FALSE, col.names = TRUE)
```

```
# \\\\\\\\\\\\\\\\\\\\\\\\\\\\ Test 3 ////////////////////////////

# ----------- 1 - Leitura de dados (ANSYS) ---------------
setwd("C:/Users/elima/Documents/IPB/Thesis/Frames")
getwd()
rm(list=ls(all=TRUE))
library(reshape2)
library(ggplot2)
library(dplyr)
library(viridis)
nodes_coord <- read.csv("nodes_coordinates_3.csv", header = TRUE, sep = ";", dec = ",")
nodes_elem <- read.csv("nodes_elements_3.csv", header = TRUE, sep = ";", dec = ",")
elem_str_ogd <- read.csv("element_strain_ogden_3.csv", header = TRUE, sep = ";", dec = ",")
nodes_str_ogd <- read.csv("nodes_strain_ogden_3.csv", header = TRUE, sep = ";", dec = ",")
elem_str_moon <- read.csv("element_strain_mooney_3.csv", header = TRUE, sep = ";", dec = ",")
nodes_str_moon <- read.csv("nodes_strain_mooney_3.csv", header = TRUE, sep = ";", dec = ",")
elem_str_neo <- read.csv("element_strain_neohook_3.csv", header = TRUE, sep = ";", dec = ",")
nodes_str_neo <- read.csv("nodes_strain_neohook_3.csv", header = TRUE, sep = ";", dec = ",")




# ----------- 2 - Transformação de dados para obtenção de coordenadas dos Elementos (ANSYS)
---------------
# ------------2.1 - OGDEN ---------------

file_ogd_1 <- merge(elem_str_ogd, nodes_elem, all=TRUE, by.x = "ELEM", by.y = "ELEM")
file_ogd_1 <- file_ogd_1[,-3:-7]
for (i in 1:8) {
  file_ogd_1 <- merge(file_ogd_1, nodes_coord[,c(1,2)], all.x=TRUE,
                 by.x = paste("NODE",i, sep = "_", collapse = NULL), by.y = "NODE")
  colnames(file_ogd_1)[ncol(file_ogd_1)] <- paste("NODE",i,"X", sep = "_", collapse = NULL)
}
for (i in 1:8) {
  file_ogd_1 <- merge(file_ogd_1, nodes_coord[,c(1,3)], all.x=TRUE,
                 by.x = paste("NODE",i, sep = "_", collapse = NULL), by.y = "NODE")
  colnames(file_ogd_1)[ncol(file_ogd_1)] <- paste("NODE",i,"Y", sep = "_", collapse = NULL)
}
par(pty="s")
plot(nodes_coord$Y~nodes_coord$X, type="p",pch=".", asp=1)

file_ogd_2 <- file_ogd_1[,-10]
file_ogd_2 <- file_ogd_2[,-1:-8]
file_ogd_2 <- file_ogd_2[,-10:-17]
colnam <- colnames(file_ogd_2)
colnam <- colnam[-1]
file_ogd_2 <- melt(data = file_ogd_2, id.vars = "ELEM", measure.vars = colnam)

file_ogd_3 <- file_ogd_1[,-10]
file_ogd_3 <- file_ogd_3[,-1:-8]
file_ogd_3 <- file_ogd_3[,-2:-9]
colnam <- colnames(file_ogd_3)
colnam <- colnam[-1]
file_ogd_3 <- melt(data = file_ogd_3, id.vars = "ELEM", measure.vars = colnam)
file_ogd_4 <- cbind(file_ogd_2, file_ogd_3)
file_ogd_4 <- file_ogd_4[,-4]
colnames(file_ogd_4)[c(3,5)] <- c("X","Y")
file_ogd_5 <- file_ogd_1[,c("ELEM","EPTOY")]
mesh_ogd <- merge(file_ogd_5,file_ogd_4, by = c("ELEM"))
mesh_ogd <- mesh_ogd[order(mesh_ogd$ELEM,
                           atan2(mesh_ogd$X-mean(mesh_ogd$X),mesh_ogd$Y-mean(mesh_ogd$Y))),]
mesh_ogd_2 <- data.frame()
mesh_ogd_for <- data.frame()
for(i in unique(mesh_ogd$ELEM)){
  mesh_ogd_for <- filter(mesh_ogd, ELEM==i)
  mesh_ogd_for <- mesh_ogd_for[order(atan2(mesh_ogd_for$X-mean(mesh_ogd_for$X),mesh_ogd_for$Y-
mean(mesh_ogd_for$Y))),]
  mesh_ogd_2 <- rbind(mesh_ogd_2, mesh_ogd_for)
}
```

```
# ------------2.2 - Mooney_rivlin ---------------

file_moon_1 <- merge(elem_str_moon, nodes_elem, all=TRUE, by.x = "ELEM", by.y = "ELEM")
file_moon_1 <- file_moon_1[,-3:-7]
for (i in 1:8) {
  file_moon_1 <- merge(file_moon_1, nodes_coord[,c(1,2)], all.x=TRUE,
                       by.x = paste("NODE",i, sep = "_", collapse = NULL), by.y = "NODE")
  colnames(file_moon_1)[ncol(file_moon_1)] <- paste("NODE",i,"X", sep = "_", collapse = NULL)
}
for (i in 1:8) {
  file_moon_1 <- merge(file_moon_1, nodes_coord[,c(1,3)], all.x=TRUE,
                       by.x = paste("NODE",i, sep = "_", collapse = NULL), by.y = "NODE")
  colnames(file_moon_1)[ncol(file_moon_1)] <- paste("NODE",i,"Y", sep = "_", collapse = NULL)
}
par(pty="s")
plot(nodes_coord$Y~nodes_coord$X, type="p",pch=".",asp=1)

file_moon_2 <- file_moon_1[,-10]
file_moon_2 <- file_moon_2[,-1:-8]
file_moon_2 <- file_moon_2[,-10:-17]
colnam <- colnames(file_moon_2)
colnam <- colnam[-1]
file_moon_2 <- melt(data = file_moon_2, id.vars = "ELEM", measure.vars = colnam)

file_moon_3 <- file_moon_1[,-10]
file_moon_3 <- file_moon_3[,-1:-8]
file_moon_3 <- file_moon_3[,-2:-9]
colnam <- colnames(file_moon_3)
colnam <- colnam[-1]
file_moon_3 <- melt(data = file_moon_3, id.vars = "ELEM", measure.vars = colnam)
file_moon_4 <- cbind(file_moon_2, file_moon_3)
file_moon_4 <- file_moon_4[,-4]
colnames(file_moon_4)[c(3,5)] <- c("X","Y")
file_moon_5 <- file_moon_1[,c("ELEM","EPTOY")]
mesh_moon <- merge(file_moon_5,file_moon_4, by = c("ELEM"))
mesh_moon <- mesh_moon[order(mesh_moon$ELEM,
                             atan2(mesh_moon$X-mean(mesh_moon$X),mesh_moon$Y-mean(mesh_moon$Y))),]
mesh_moon_2 <- data.frame()
mesh_moon_for <- data.frame()
for(i in unique(mesh_moon$ELEM)){
  mesh_moon_for <- filter(mesh_moon, ELEM==i)
  mesh_moon_for <- mesh_moon_for[order(atan2(mesh_moon_for$X-mean(mesh_moon_for$X),mesh_moon_for$Y-
mean(mesh_moon_for$Y))),]
  mesh_moon_2 <- rbind(mesh_moon_2, mesh_moon_for)
}

# ------------2.3 - NEO_hookean ---------------

file_neo_1 <- merge(elem_str_neo, nodes_elem, all=TRUE, by.x = "ELEM", by.y = "ELEM")
file_neo_1 <- file_neo_1[,-3:-7]
for (i in 1:8) {
  file_neo_1 <- merge(file_neo_1, nodes_coord[,c(1,2)], all.x=TRUE,
                       by.x = paste("NODE",i, sep = "_", collapse = NULL), by.y = "NODE")
  colnames(file_neo_1)[ncol(file_neo_1)] <- paste("NODE",i,"X", sep = "_", collapse = NULL)
}
for (i in 1:8) {
  file_neo_1 <- merge(file_neo_1, nodes_coord[,c(1,3)], all.x=TRUE,
                       by.x = paste("NODE",i, sep = "_", collapse = NULL), by.y = "NODE")
  colnames(file_neo_1)[ncol(file_neo_1)] <- paste("NODE",i,"Y", sep = "_", collapse = NULL)
}
par(pty="s")
plot(nodes_coord$Y~nodes_coord$X, type="p",pch=".",asp=1)

file_neo_2 <- file_neo_1[,-10]
file_neo_2 <- file_neo_2[,-1:-8]
file_neo_2 <- file_neo_2[,-10:-17]
colnam <- colnames(file_neo_2)
colnam <- colnam[-1]
file_neo_2 <- melt(data = file_neo_2, id.vars = "ELEM", measure.vars = colnam)
```

```
file_neo_3 <- file_neo_1[,-10]
file_neo_3 <- file_neo_3[,-1:-8]
file_neo_3 <- file_neo_3[,-2:-9]
colnam <- colnames(file_neo_3)
colnam <- colnam[-1]
file_neo_3 <- melt(data = file_neo_3, id.vars = "ELEM", measure.vars = colnam)
file_neo_4 <- cbind(file_neo_2, file_neo_3)
file_neo_4 <- file_neo_4[,-4]
colnames(file_neo_4)[c(3,5)] <- c("X","Y")
file_neo_5 <- file_neo_1[,c("ELEM","EPTOY")]
mesh_neo <- merge(file_neo_5,file_neo_4, by = c("ELEM"))
mesh_neo <- mesh_neo[order(mesh_neo$ELEM,
                           atan2(mesh_neo$X-mean(mesh_neo$X),mesh_neo$Y-mean(mesh_neo$Y))),]
mesh_neo_2 <- data.frame()
mesh_neo_for <- data.frame()
for(i in unique(mesh_neo$ELEM)){
  mesh_neo_for <- filter(mesh_neo, ELEM==i)
  mesh_neo_for <- mesh_neo_for[order(atan2(mesh_neo_for$X-mean(mesh_neo_for$X),mesh_neo_for$Y-
mean(mesh_neo_for$Y))),]
  mesh_neo_2 <- rbind(mesh_neo_2, mesh_neo_for)
}

# ----------- 3 - Conversão de Polígonos Para identificação de pontos ---------------

# ----------- 3.1 - OGDEN -----------------

#ggplotly(p)
#install.packages("sfheaders")
library(sf)
library(sfheaders)

my_df_sf_ogd <- st_as_sf(mesh_ogd,
                         coords = c('X', 'Y')) %>%
  st_set_crs(26918) %>%
  group_by(ELEM) %>%
  summarise() %>%
  ungroup() %>%  # Just in case
  st_convex_hull()
my_df_sf_ogd <- merge(my_df_sf_ogd,file_ogd_5, by = c("ELEM"))

#install.packages("prevR")
library(prevR)
library(sp)
library(rgeos)
library(sp)
library(rgdal)
library(plotly)
part_ogd <- as(my_df_sf_ogd, 'Spatial')

setwd("C:/Users/elima/Documents/IPB/Thesis/Frames")
points_ogd <- read.csv("real_points_3_1.csv", header = TRUE, sep = ";", dec = ",")

#points <- points[c(-1,-2,-57:-61),]
dat2_ogd <- points_ogd
dat_ogd <- dat2_ogd
coordinates(dat_ogd) <- ~ x_real + y_real
proj4string(dat_ogd) <- proj4string(part_ogd)
ov_ogd <- over(dat_ogd, part_ogd)
my_df_sf_ogd_2 <- filter(my_df_sf_ogd, ELEM %in% ov_ogd$ELEM)


# ----------- 3.2 - Mooney-rivlin -----------------

#ggplotly(p)
#install.packages("sfheaders")
library(sf)
library(sfheaders)

my_df_sf_moon <- st_as_sf(mesh_moon,
                          coords = c('X', 'Y')) %>%
```

```
  st_set_crs(26918) %>%
  group_by(ELEM) %>%
  summarise() %>%
  ungroup() %>%  # Just in case
  st_convex_hull()
my_df_sf_moon <- merge(my_df_sf_moon,file_moon_5, by = c("ELEM"))

#install.packages("prevR")
library(prevR)
library(sp)
library(rgeos)
library(sp)
library(rgdal)
library(plotly)
part_moon <- as(my_df_sf_moon, 'Spatial')

setwd("C:/Users/elima/Documents/IPB/Thesis/Frames")
points_moon <- read.csv("real_points_3_1.csv", header = TRUE, sep = ";", dec = ",")

#points <- points[c(-1,-2,-57:-61),]
dat2_moon <- points_moon
dat_moon <- dat2_moon
coordinates(dat_moon) <- ~ x_real + y_real
proj4string(dat_moon) <- proj4string(part_moon)
ov_moon <- over(dat_moon, part_moon)
my_df_sf_moon_2 <- filter(my_df_sf_moon, ELEM %in% ov_moon$ELEM)
#ggplotly(p)

# ----------- 3.2 - NEO-hookean -----------------

#ggplotly(p)
#install.packages("sfheaders")
library(sf)
library(sfheaders)

my_df_sf_neo <- st_as_sf(mesh_neo,
                         coords = c('X', 'Y')) %>%
  st_set_crs(26918) %>%
  group_by(ELEM) %>%
  summarise() %>%
  ungroup() %>%  # Just in case
  st_convex_hull()
my_df_sf_neo <- merge(my_df_sf_neo,file_neo_5, by = c("ELEM"))

#install.packages("prevR")
library(prevR)
library(sp)
library(rgeos)
library(sp)
library(rgdal)
library(plotly)
part_neo <- as(my_df_sf_neo, 'Spatial')

setwd("C:/Users/elima/Documents/IPB/Thesis/Frames")
points_neo <- read.csv("real_points_3_1.csv", header = TRUE, sep = ";", dec = ",")

#points <- points[c(-1,-2,-57:-61),]
dat2_neo <- points_neo
dat_neo <- dat2_neo
coordinates(dat_neo) <- ~ x_real + y_real
proj4string(dat_neo) <- proj4string(part_neo)
ov_neo <- over(dat_neo, part_neo)
my_df_sf_neo_2 <- filter(my_df_sf_neo, ELEM %in% ov_neo$ELEM)
#ggplotly(p)

# ----------- 4 - Plotagem de gráfico ---------------


# p <- ggplot() +
#   geom_polygon(data=mesh_ogd_2, aes(x = X, y = Y, fill = EPTOY, group = ELEM)) +
```

```
#   scale_fill_gradient( low="white", high="black") +
#   geom_polygon(data=filter(mesh_ogd_2, ELEM %in% ov_neo$ELEM),
#                aes(x = X, y = Y, group = ELEM), fill = "orange")+ coord_fixed() +
#   geom_point(data=points, aes(x=x_real, y=y_real))
# p
#ggplotly(p)

# ----------- 5 - Leitura de dados de DIC ---------------
setwd("C:/Users/elima/Documents/IPB/Thesis/Frames")
test_1_1 <- read.csv("test_3_1.csv", header = TRUE, sep = ",", dec = ".")
test_1_2 <- read.csv("test_3_2.csv", header = TRUE, sep = ",", dec = ".")
test_1_3 <- read.csv("test_3_3.csv", header = TRUE, sep = ",", dec = ".")

plot(nodes_coord$Y~nodes_coord$X, type="p",pch=".",asp=1)
points_1_1 <- read.csv("real_points_3_1.csv", header = TRUE, sep = ";", dec = ",")

points(points_1_1$x_real, points_1_1$y_real, asp=1, col=2)
points_1_2 <- read.csv("real_points_3_2.csv", header = TRUE, sep = ";", dec = ",")
points_1_3 <- read.csv("real_points_3_3.csv", header = TRUE, sep = ";", dec = ",")
test_1_1_2 <- melt(data=test_1_1, id.vars="Index", measure.vars = colnames(test_1_1[-1]))
test_1_2_2 <- melt(data=test_1_2, id.vars="Index", measure.vars = colnames(test_1_2[-1]))
test_1_3_2 <- melt(data=test_1_3, id.vars="Index", measure.vars = colnames(test_1_3[-1]))
last_data <- 10

p1 <- ggplot(data = test_1_1_2, aes(x = (Index-1)/2.5, y = value, group = variable, color =
variable)) +
  geom_line() + xlab("Seconds [s]") + ylab("Strain [mm/mm]") +
  geom_line(data = filter(test_1_1_2, Index >= max(test_1_1_2$Index)+1-last_data),
            aes(x = (Index-1)/2.5, y = value, group = variable), color = "red") +
  theme_classic() + theme(legend.position = "none") +
  scale_x_continuous(breaks=seq(0,ceiling(max((test_1_1_2$Index-1)/2.5)),length.out = 6)) +
  scale_y_continuous(breaks=seq(0,ceiling(max(test_1_1_2$value)),length.out = 6))
p1
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_3_points_strain_1.pdf", dpi=300)
#ggplotly(p1)

p2 <- ggplot(data = test_1_2_2, aes(x = (Index-1)/2.5, y = value, group = variable, color =
variable)) +
  geom_line() + xlab("Seconds [s]") + ylab("Strain [mm/mm]") +
  geom_line(data = filter(test_1_2_2, Index >= max(test_1_2_2$Index)+1-last_data),
            aes(x = (Index-1)/2.5, y = value, group = variable), color = "red") +
  theme_classic() + theme(legend.position = "none") +
  scale_x_continuous(breaks=seq(0,ceiling(max((test_1_2_2$Index-1)/2.5)),length.out = 6)) +
  scale_y_continuous(breaks=seq(0,ceiling(max(test_1_2_2$value)),length.out = 6))
p2
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_3_points_strain_2.pdf", dpi=300)
#ggplotly(p2)

p3 <- ggplot(data = test_1_3_2, aes(x = (Index-1)/2.5, y = value, group = variable, color =
variable)) +
  geom_line() + xlab("Seconds [s]") + ylab("Strain [mm/mm]") +
  geom_line(data = filter(test_1_3_2, Index >= max(test_1_3_2$Index)+1-last_data),
            aes(x = (Index-1)/2.5, y = value, group = variable), color = "red") +
  theme_classic() + theme(legend.position = "none") +
  scale_x_continuous(breaks=seq(0,ceiling(max((test_1_3_2$Index-1)/2.5)),length.out = 6)) +
  scale_y_continuous(breaks=seq(0,ceiling(max(test_1_3_2$value)),length.out = 6))
p3
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_3_points_strain_3.pdf", dpi=300)
#ggplotly(p3)

last_data <- 10
test_final_1_1 <- test_1_1[(nrow(test_1_1)-last_data):(nrow(test_1_1)),]
media_1_1 <- cbind(points_1_1,
                   as.data.frame(colMeans(test_final_1_1[,2:ncol(test_final_1_1)])))
media_1_1 <- media_1_1[order(media_1_1$x_real, media_1_1$y_real),]

test_final_1_2 <- test_1_2[(nrow(test_1_2)-last_data):(nrow(test_1_2)),]
```

```
media_1_2 <- cbind(points_1_2,
                   as.data.frame(colMeans(test_final_1_2[,2:ncol(test_final_1_2)])))
media_1_2 <- media_1_2[order(media_1_2$x_real, media_1_2$y_real),]

test_final_1_3 <- test_1_3[(nrow(test_1_3)-last_data):(nrow(test_1_3)),]
media_1_3 <- cbind(points_1_3,
                   as.data.frame(colMeans(test_final_1_3[,2:ncol(test_final_1_3)])))
media_1_3 <- media_1_3[order(media_1_3$x_real, media_1_3$y_real),]



#View(media_1_1)
#View(media_1_2)
media_1_ogd <- as.data.frame(cbind(media_1_1[,2], media_1_1[,3],
                                   media_1_1[,4]/100, media_1_2[,4]/100, media_1_3[,4]/100,
                                   ((media_1_1[,4] + media_1_2[,4] + media_1_3[,4])/300),
ov_ogd$EPTOY))
media_1_moon <- as.data.frame(cbind(media_1_1[,2], media_1_1[,3],
                                    media_1_1[,4]/100, media_1_2[,4]/100, media_1_3[,4]/100,
                                    ((media_1_1[,4] + media_1_2[,4] + media_1_3[,4])/300),
ov_moon$EPTOY))
media_1_neo <- as.data.frame(cbind(media_1_1[,2], media_1_1[,3],
                                   media_1_1[,4]/100, media_1_2[,4]/100, media_1_3[,4]/100,
                                   ((media_1_1[,4] + media_1_2[,4] + media_1_3[,4])/300),
ov_neo$EPTOY))

colnames(media_1_ogd) <- c("x", "y", "media_1_1", "media_1_2", "media_1_3", "media", "EPTOY_ELEM")
colnames(media_1_moon) <- c("x", "y", "media_1_1", "media_1_2", "media_1_3", "media", "EPTOY_ELEM")
colnames(media_1_neo) <- c("x", "y", "media_1_1", "media_1_2", "media_1_3", "media", "EPTOY_ELEM")

EPTOY_2 <- NA
x0 <- NA
y0 <- NA
erro_1_1 <- NA
erro_1_2 <- NA
erro_1 <- NA

# ---------------- 5.1 - ogden ---------------

nodes <- merge(x = nodes_str_ogd, y = nodes_coord[,1:3], all.x = TRUE, by = "NODE")
pos <- as.matrix(nodes[,c(3,4)])
new.pos_tot <- as.matrix(media_1_ogd[,c("x","y")])
distance <- data.frame()

for(i in 1:nrow(media_1_ogd)){
  new.pos <- new.pos_tot[i,]
  distance <- sqrt((pos[,1] - new.pos[1])^2+(pos[,2] - new.pos[2])^2)
  nearest.idx <- which.min(distance)
  #nearest.idx <- which.min(colSums(sqrt((t(pos) - new.pos)^2)))
  media_1_ogd[i,"EPTOY_NODES"] = nodes[nearest.idx,"EPTOY"]
  media_1_ogd[i,"x0"] = nodes[nearest.idx,"X"]
  media_1_ogd[i,"y0"] = nodes[nearest.idx,"Y"]

}


erro_1_1_elem <- as.data.frame(cbind(100*sqrt((media_1_ogd$media_1_1-media_1_ogd$EPTOY_ELEM)^2)/
media_1_ogd$media_1_1))
erro_1_2_elem <- as.data.frame(cbind(100*sqrt((media_1_ogd$media_1_2-media_1_ogd$EPTOY_ELEM)^2)/
media_1_ogd$media_1_2))
erro_1_3_elem <- as.data.frame(cbind(100*sqrt((media_1_ogd$media_1_3-media_1_ogd$EPTOY_ELEM)^2)/
media_1_ogd$media_1_3))
erro_1_elem <- as.data.frame(cbind(100*sqrt((media_1_ogd$media-media_1_ogd$EPTOY_ELEM)^2)/
media_1_ogd$media))
colnames(erro_1_1_elem) <- "error_1_1_elem"
colnames(erro_1_2_elem) <- "error_1_2_elem"
colnames(erro_1_3_elem) <- "error_1_3_elem"
colnames(erro_1_elem) <- "error_1_elem"
erro_1_1_node <- as.data.frame(cbind(100*sqrt((media_1_ogd$media_1_1-media_1_ogd$EPTOY_NODES)^2)/
media_1_ogd$media_1_1))
```

```
erro_1_2_node <- as.data.frame(cbind(100*sqrt((media_1_ogd$media_1_2-media_1_ogd$EPTOY_NODES)^2)/
media_1_ogd$media_1_2))
erro_1_3_node <- as.data.frame(cbind(100*sqrt((media_1_ogd$media_1_3-media_1_ogd$EPTOY_NODES)^2)/
media_1_ogd$media_1_3))
erro_1_node <- as.data.frame(cbind(100*sqrt((media_1_ogd$media-media_1_ogd$EPTOY_NODES)^2)/
media_1_ogd$media))
colnames(erro_1_1_node) <- "error_1_1_node"
colnames(erro_1_2_node) <- "error_1_2_node"
colnames(erro_1_3_node) <- "error_1_3_node"
colnames(erro_1_node) <- "error_1_node"

media_1_ogd <- cbind(media_1_ogd, erro_1_1_elem, erro_1_2_elem, erro_1_3_elem, erro_1_elem,
erro_1_1_node, erro_1_2_node, erro_1_3_node, erro_1_node)
plot(pos,asp=1)

# ---------------- 5.2 - Mooney-rivlin ---------------

nodes <- merge(x = nodes_str_moon, y = nodes_coord[,1:3], all.x = TRUE, by = "NODE")
pos <- as.matrix(nodes[,c(3,4)])
new.pos_tot <- as.matrix(media_1_moon[,c("x","y")])
distance <- data.frame()

for(i in 1:nrow(media_1_moon)){
  new.pos <- new.pos_tot[i,]
  distance <- sqrt((pos[,1] - new.pos[1])^2+(pos[,2] - new.pos[2])^2)
  nearest.idx <- which.min(distance)
  #nearest.idx <- which.min(colSums(sqrt((t(pos) - new.pos)^2)))
  media_1_moon[i,"EPTOY_NODES"] = nodes[nearest.idx,"EPTOY"]
  media_1_moon[i,"x0"] = nodes[nearest.idx,"X"]
  media_1_moon[i,"y0"] = nodes[nearest.idx,"Y"]

}


erro_1_1_node <- as.data.frame(cbind(100*sqrt((media_1_moon$media_1_1-media_1_moon$EPTOY_NODES)^2)/
media_1_moon$media_1_1))
erro_1_2_node <- as.data.frame(cbind(100*sqrt((media_1_moon$media_1_2-media_1_moon$EPTOY_NODES)^2)/
media_1_moon$media_1_2))
erro_1_3_node <- as.data.frame(cbind(100*sqrt((media_1_moon$media_1_3-media_1_moon$EPTOY_NODES)^2)/
media_1_moon$media_1_3))
erro_1_node <- as.data.frame(cbind(100*sqrt((media_1_moon$media-media_1_moon$EPTOY_NODES)^2)/
media_1_moon$media))
colnames(erro_1_1_node) <- "error_1_1_node"
colnames(erro_1_2_node) <- "error_1_2_node"
colnames(erro_1_3_node) <- "error_1_3_node"
colnames(erro_1_node) <- "error_1_node"
erro_1_1_elem <- as.data.frame(cbind(100*sqrt((media_1_moon$media_1_1-media_1_moon$EPTOY_ELEM)^2)/
media_1_moon$media_1_1))
erro_1_2_elem <- as.data.frame(cbind(100*sqrt((media_1_moon$media_1_2-media_1_moon$EPTOY_ELEM)^2)/
media_1_moon$media_1_2))
erro_1_3_elem <- as.data.frame(cbind(100*sqrt((media_1_moon$media_1_3-media_1_moon$EPTOY_ELEM)^2)/
media_1_moon$media_1_3))
erro_1_elem <- as.data.frame(cbind(100*sqrt((media_1_moon$media-media_1_moon$EPTOY_ELEM)^2)/
media_1_moon$media))
colnames(erro_1_1_elem) <- "error_1_1_elem"
colnames(erro_1_2_elem) <- "error_1_2_elem"
colnames(erro_1_3_elem) <- "error_1_3_elem"
colnames(erro_1_elem) <- "error_1_elem"

media_1_moon <- cbind(media_1_moon, erro_1_1_elem, erro_1_2_elem, erro_1_3_elem, erro_1_elem,
                      erro_1_1_node, erro_1_2_node, erro_1_3_node, erro_1_node)
plot(pos, asp=1)


# ---------------- 5.3 - Neo-hookean ---------------

nodes <- merge(x = nodes_str_neo, y = nodes_coord[,1:3], all.x = TRUE, by = "NODE")
pos <- as.matrix(nodes[,c(3,4)])
new.pos_tot <- as.matrix(media_1_neo[,c("x","y")])
distance <- data.frame()
```

```
for(i in 1:nrow(media_1_neo)){
  new.pos <- new.pos_tot[i,]
  distance <- sqrt((pos[,1] - new.pos[1])^2+(pos[,2] - new.pos[2])^2)
  nearest.idx <- which.min(distance)
  #nearest.idx <- which.min(colSums(sqrt((t(pos) - new.pos)^2)))
  media_1_neo[i,"EPTOY_NODES"] = nodes[nearest.idx,"EPTOY"]
  media_1_neo[i,"x0"] = nodes[nearest.idx,"X"]
  media_1_neo[i,"y0"] = nodes[nearest.idx,"Y"]

}


erro_1_1_elem <- as.data.frame(cbind(100*sqrt((media_1_neo$media_1_1-media_1_neo$EPTOY_ELEM)^2)/
media_1_neo$media_1_1))
erro_1_2_elem <- as.data.frame(cbind(100*sqrt((media_1_neo$media_1_2-media_1_neo$EPTOY_ELEM)^2)/
media_1_neo$media_1_2))
erro_1_3_elem <- as.data.frame(cbind(100*sqrt((media_1_neo$media_1_3-media_1_neo$EPTOY_ELEM)^2)/
media_1_neo$media_1_3))
erro_1_elem <- as.data.frame(cbind(100*sqrt((media_1_neo$media-media_1_neo$EPTOY_ELEM)^2)/
media_1_neo$media))
colnames(erro_1_1_elem) <- "error_1_1_elem"
colnames(erro_1_2_elem) <- "error_1_2_elem"
colnames(erro_1_3_elem) <- "error_1_3_elem"
colnames(erro_1_elem) <- "error_1_elem"

erro_1_1_node <- as.data.frame(cbind(100*sqrt((media_1_neo$media_1_1-media_1_neo$EPTOY_NODES)^2)/
media_1_neo$media_1_1))
erro_1_2_node <- as.data.frame(cbind(100*sqrt((media_1_neo$media_1_2-media_1_neo$EPTOY_NODES)^2)/
media_1_neo$media_1_2))
erro_1_3_node <- as.data.frame(cbind(100*sqrt((media_1_neo$media_1_3-media_1_neo$EPTOY_NODES)^2)/
media_1_neo$media_1_3))
erro_1_node <- as.data.frame(cbind(100*sqrt((media_1_neo$media-media_1_neo$EPTOY_NODES)^2)/
media_1_neo$media))
colnames(erro_1_1_node) <- "error_1_1_node"
colnames(erro_1_2_node) <- "error_1_2_node"
colnames(erro_1_3_node) <- "error_1_3_node"
colnames(erro_1_node) <- "error_1_node"

media_1_neo <- cbind(media_1_neo, erro_1_1_node, erro_1_2_node, erro_1_3_node, erro_1_node,
                     erro_1_1_elem, erro_1_2_elem, erro_1_3_elem, erro_1_elem)
plot(pos,asp=1)


# ----------- 6 - Plotagem Final ---------------

library(viridis)
library(plotly)

#gp2 <- ggplotly(p2)
##library(htmlwidgets)
#saveWidget(gp2, "p2.html", selfcontained = F, libdir = "lib")
#widget_file_size(p1)
# ggplotly(p1)
cons_models <- c("Ogden", "Mooney-Rivlyn", "Neo Hookean")
position_models_x <- c(12,42,72)
position_models_y <- c(-5,-5,-5)
label <- as.data.frame(cbind(cons_models, as.numeric(position_models_x),
as.numeric(position_models_y)))
p1 <- ggplot() +
  geom_polygon(data=mesh_ogd_2, aes(x = X, y = Y, fill = EPTOY, group = ELEM)) +
  geom_polygon(data=mesh_moon_2, aes(x = X + 30, y = Y, fill = EPTOY, group = ELEM)) +
  geom_polygon(data=mesh_neo_2, aes(x = X + 60, y = Y, fill = EPTOY, group = ELEM)) +
  scale_fill_gradient(low="white", high="black") + coord_fixed() +
  geom_point(data=media_1_ogd, aes(x=x, y=y, colour=error_1_node), size = 2) +
  geom_point(data=media_1_moon, aes(x=x + 30, y=y, colour=error_1_node), size = 2) +
  geom_point(data=media_1_neo, aes(x=x + 60, y=y, colour=error_1_node), size = 2) +
  scale_color_gradient(low="turquoise1", high="indianred1") +
  geom_point(data=media_1_ogd, aes(x=as.numeric(x0), y=as.numeric(y0)), colour="black", size = 1,
shape = "x") +
```

```
  geom_point(data=media_1_moon, aes(x=as.numeric(x0) + 30, y=as.numeric(y0)), colour="black", size
= 1, shape = "x") +
  geom_point(data=media_1_neo, aes(x=as.numeric(x0) + 60, y=as.numeric(y0)), colour="black", size =
1, shape = "x") +
  theme_classic() + theme_void() + labs(fill="ANSYS Strain", colour="Error") +
  geom_text(data=label, aes(x=position_models_x, y=position_models_y, label = cons_models),
hjust=0.5, vjust=0.5)
p1
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_3_comparison_node.pdf", dpi=300)
#gp1 <- ggplotly(p1)
# library(htmlwidgets)
# saveWidget(gp1, "p1.html", selfcontained = F, libdir = "lib")
# widget_file_size(p1)
# ggplotly(p1)

p2 <- ggplot() +
  geom_polygon(data=mesh_ogd_2, aes(x = X, y = Y, fill = EPTOY, group = ELEM)) +
  geom_polygon(data=mesh_moon_2, aes(x = X + 30, y = Y, fill = EPTOY, group = ELEM)) +
  geom_polygon(data=mesh_neo_2, aes(x = X + 60, y = Y, fill = EPTOY, group = ELEM)) +
  scale_fill_gradient(low="white", high="black") + coord_fixed() +
  geom_point(data=media_1_ogd, aes(x=x, y=y, colour=error_1_elem), size = 2) +
  geom_point(data=media_1_moon, aes(x=x + 30, y=y, colour=error_1_elem), size = 2) +
  geom_point(data=media_1_neo, aes(x=x + 60, y=y, colour=error_1_elem), size = 2) +
  scale_color_gradient(low="turquoise1", high="indianred1") +
  geom_point(data=media_1_ogd, aes(x=as.numeric(x0), y=as.numeric(y0)), colour="black", size = 1,
shape = "x") +
  geom_point(data=media_1_moon, aes(x=as.numeric(x0) + 30, y=as.numeric(y0)), colour="black", size
= 1, shape = "x") +
  geom_point(data=media_1_neo, aes(x=as.numeric(x0) + 60, y=as.numeric(y0)), colour="black", size =
1, shape = "x") +
  theme_classic() + theme_void() + labs(fill="ANSYS Strain", colour="Error") +
  geom_text(data=label, aes(x=position_models_x, y=position_models_y, label = cons_models),
hjust=0.5, vjust=0.5)
p2
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_3_comparison_elem.pdf", dpi=300)

export_elem <- as.data.frame(cbind(1:nrow(media_1_ogd), media_1_ogd$x, media_1_ogd$y,
media_1_ogd$media,
                                   media_1_ogd$EPTOY_ELEM, media_1_ogd$error_1_elem,
                                   media_1_moon$EPTOY_ELEM, media_1_moon$error_1_elem,
                                   media_1_neo$EPTOY_ELEM, media_1_neo$error_1_elem))
colnames(export_elem) <- c("point", "point_x", "point_y", "mean",
                           "y_strain_ogden", "O",
                           "y_strain_mooney_riv", "M",
                           "y_strain_neo_hook", "N")
export_2_elem <- melt(data = export_elem[,c(1, 6, 8, 10)],
                      id.vars = "point", measure.vars = colnames(export_elem)[c(6,8,10)])
#export_2_elem <- filter(export_2_elem, point %in% setdiff(export_2_elem$point, c(5, 6, 48, 49)))
export_3_elem <- data_frame(
  factor = c("O", "M", "N"),
  mean = c(mean(export_elem[,6]), mean(export_elem[,8]), mean(export_elem[,10])),
  sd = c(sd(export_elem[,6]), sd(export_elem[,8]), sd(export_elem[,10]))
)
export_3_elem$factor <- factor(export_3_elem$factor, levels = levels(export_2_elem$variable))


export_node <- as.data.frame(cbind(1:nrow(media_1_ogd), media_1_ogd$x, media_1_ogd$y,
media_1_ogd$media,
                                   media_1_ogd$EPTOY_NODES, media_1_ogd$error_1_node,
                                   media_1_moon$EPTOY_NODES, media_1_moon$error_1_node,
                                   media_1_neo$EPTOY_NODES, media_1_neo$error_1_node))
colnames(export_node) <- c("point", "point_x", "point_y", "mean",
                           "y_strain_ogden", "O",
                           "y_strain_mooney_riv", "M",
                           "y_strain_neo_hook", "N")
export_2_node <- melt(data = export_node[,c(1, 6, 8, 10)],
                      id.vars = "point", measure.vars = colnames(export_node)[c(6,8,10)])
#export_2_node <- filter(export_2_node, point %in% setdiff(export_2_node$point, c(5, 6, 48, 49)))
```

```
export_3_node <- data_frame(
  factor = c("O", "M", "N"),
  mean = c(mean(export_node[,6]), mean(export_node[,8]), mean(export_node[,10])),
  sd = c(sd(export_node[,6]), sd(export_node[,8]), sd(export_node[,10]))
)
export_3_node$factor <- factor(export_3_node$factor, levels = levels(export_2_node$variable))
#install.packages("ggpubr")
library(ggpubr)
ggboxplot(export_2_elem, x = "variable", y = "value",
          color = "variable", fill = "lightgray", palette = c("dodgerblue4","green4",
"firebrick1"),
          ylab = "Errors [%]", xlab = "Constitutive Models")
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_3_boxplot_elem.pdf", dpi=300)

ggboxplot(export_2_node, x = "variable", y = "value",
          color = "variable", fill = "lightgray", palette = c("dodgerblue4","green4",
"firebrick1"),
          ylab = "Errors [%]", xlab = "Constitutive Models")
setwd("C:/Users/elima/Documents/IPB/Thesis/Figures")
ggsave("test_3_boxplot_node.pdf", dpi=300)
options(digits=10)

#anova for elem
aov_elem <- aov(value ~ variable, data = export_2_elem)
summary(aov_elem)
# library(gridExtra)
# capture.output(summary(aov_elem),file="test_1_anova.doc")
MSE <- summary(aov_elem)[[1]]["Residuals","Mean Sq"]
ks.test(aov_elem$residuals,"pnorm",0,sqrt(MSE),alternative = "t")
TukeyHSD(aov_elem)
pdf("test_3_tukey_elem.pdf", height=5, width=5)
plot(TukeyHSD(aov_elem), las = 1)# 2. Create a plot
dev.off() # Close the pdf file

#anova for node
aov_node <- aov(value ~ variable, data = export_2_node)
summary(aov_node)
MSE <- summary(aov_node)[[1]]["Residuals","Mean Sq"]
ks.test(aov_node$residuals,"pnorm",0,sqrt(MSE),alternative = "t")
TukeyHSD(aov_node)
pdf("test_3_tukey_node.pdf", height=5, width=5)
plot(TukeyHSD(aov_node), las = 1)# 2. Create a plot
dev.off() # Close the pdf file


p1 <- ggplot() +
  geom_bar(data=export_2_elem, aes(x=point, y=value, fill=factor(variable)),
           position=position_dodge(), stat="identity") +
  ggtitle("Errors and Points") +
  scale_fill_manual("legend", values = c("dodgerblue4","green4", "firebrick1"))
p1
#ggplotly(p1)


p2 <- ggplot() +
  geom_bar(data=export_2_node, aes(x=point, y=value, fill=factor(variable)),
           position=position_dodge(), stat="identity") +
  ggtitle("Errors and Points") +
  scale_fill_manual("legend", values = c("dodgerblue4","green4", "firebrick1"))
p2
#ggplotly(p2)

q <- ggplot() +
  geom_bar(data=export_3_elem, aes(x=factor, y=mean, fill=factor),
           position=position_dodge(),stat="identity") +
  ggtitle("Errors") +
  scale_fill_manual("legend", values = c("dodgerblue4","green4", "firebrick1"))
q
q <- ggplot() +
```

```
  ggtitle("Errors") +
  geom_bar(data=export_3_node, aes(x=factor, y=mean, fill=factor),
           position=position_dodge(),stat="identity") +
  ggtitle("Errors") +
  scale_fill_manual("legend", values = c("dodgerblue4","green4", "firebrick1"))
q
ggplotly(q)



p <- ggplot() +
  geom_bar(data=export_2_elem, aes(x=point, y=value, fill=factor(variable)),
           position=position_dodge(), stat="identity") +
  ggtitle("Errors and Points") +
  scale_fill_manual("legend", values = c("dodgerblue4","green4", "firebrick1"))
ggplotly(p)


q <- ggplot() +
  geom_bar(data=export_3_elem, aes(x=factor, y=mean, fill=factor),
           position=position_dodge(),stat="identity") +
  ggtitle("Errors") +
  scale_fill_manual("legend", values = c("dodgerblue4","green4", "firebrick1"))
q
ggplotly(q)

export1 <- export_elem
export2 <- export_node
colnames(export1) <- colnames(export2) <- c("Points","x","y","Mean DIC Strain",
                                            "Ogden Strain", "Ogden Strain Error",
                                            "Mooney-Rivlyn Strain", "Mooney-Rivlyn Strain Error",
                                            "Neo Hookean Strain", "Neo Hookean Strain Error")
setwd("C:/Users/elima/Documents/IPB/Thesis/Frames")
getwd()
write.table(export1, file = "test_3_results_elem.txt", append = FALSE, quote = FALSE, sep = "\t",
dec = ",", row.names = FALSE, col.names = TRUE)
write.table(export2, file = "test_3_results_node.txt", append = FALSE, quote = FALSE, sep = "\t",
dec = ",", row.names = FALSE, col.names = TRUE)
```