UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

BACHELOR'S FINAL THESIS

Study of material interpolation for 3D lightweight structures

Document:

Thesis Report

Author:

Geli I Cerdan, Jofre

Director:

Ferrer Ferre, Alex

Degree:

Bachelor's Degree in Aerospace Technology Engineering

Examination session:

Spring, 2022.

"The art of structure is where to put the holes."

Robert Le Ricolais, architect and engineer, 1894-1977

Resum

L'ús d'eines informàtiques s'ha tornat essencial en els enginyers d'avui en dia, des de la utilització de software específics fins a la programació dels mateixos. Es per això, que aquest treball busca donar les eines suficients per comprendre els coneixements essencials de programació per un enginyer modern, els quals permetin una fàcil correcció, enteniment i eficiència del codi. A més d'això, la tesis buscar donar solució a un dels problemes heurístics de l'optimització topològica. Aquest és un camp de recerca que permet trobar estructures de disseny òptimes en termes d'ús de volum. L'enfocament més popular, el mètode SIMP, una funció d'interpolació auto-penalitzada en un problema d'optimització topològica relaxada, no sempre proposa solucions possibles a la realitat. Per aquesta raó, es pretén trobar una solució alternativa al mètode SIMP per qualsevol cas. Finalment, es pretén aplicar ambdós coneixements en reescriure part del codi Swan.

A partir de reescriure un codi prèviament programat a l'assignatura d'estructures aeroespacials, anomenat "Codi Cante", es desenvolupen els apartats de programació basada en objectes, treball amb copies de seguretat al núvol, desenvolupament a partir de tests i codi net. Els quals, son eines utilitzades pels enginyers de software moderns. Referent a l'optimització topològica es treballa amb un mètode alternatiu anomenat SIMP-All per donar solució als desavantatges del mètode SIMP.

Utilitzant els mètodes prèviament descrits, per una banda, s'adquireixen els coneixements necessaris per programar de manera eficient i estructurada. En aquest punt es tracta la programació basada en objectes, treball amb copies de seguretat al núvol, desenvolupament a partir de tests i codi net. D'altra banda, els resultats referents a l'optimització topològica permeten obtenir estructures òptimes i possibles a la realitat. En primer lloc, es troba la solució del mètode SIMP-All per casos de tres dimensions. Llavors, es demostra que la solució forma part d'uns límits físics anomenats "Hashin-Shtrikman bounds" i es proposa una solució integral del mètode SIMP-All. Aquests resultats es comparen amb el mètode SIMP en dos i tres dimensions. Finalment, s'apliquen tots els nous coneixements per reescriure part del codi Swan.

En conclusió, a partir de la refactorització del "Codi Cante", s'han desenvolupat les eines suficients per comprendre els coneixements essencials de programació per un enginyer modern. Altrament, ha sigut possible obtenir una equació general pel mètode SIMP-ALL. La qual, no només permet obtenir resultats òptims, sinó que les estructures que genera seran sempre possible a la realitat. A més, l'equació s'ha implementa al programa Swan utilitzant els nous coneixements de programació, permetent així, una fàcil correcció, enteniment i eficiència del codi.

Abstract

The use of computer tools has become essential in modern engineering, from the use of specific software to programming it. That is why this work seeks to provide the tools to understand the essential knowledge of programming for a modern engineer, which allows the easy correction, understanding and efficiency of the code. In addition, the thesis seeks to solve one of the heuristic problems of topology optimization. It is a research field that obtains optimal design structures in terms of volume usage. The most popular approach, the SIMP method, an auto-penalized interpolation function in a relaxed topology optimization problem, does not always propose actual solutions. For this reason, an alternative solution to the SIMP method is to be found for any case scenario. Finally, it is intended to apply these knowledge by rewriting part of the Swan code.

By refactoring a previously programmed code from the subject 'aerospace structures', called "Codi Cante", the sections object oriented programming, work with cloud backups, test driven development and clean code are developed. These are tools used by modern software engineers. Regarding topology optimization, an alternative method called SIMP-All is used to solve the drawbacks of the SIMP method.

From the previously defined methods, knowledge is acquired to program efficiently and structurally. At this point, object oriented programming, cloud-backed work, test driven development and clean code are discussed. On the other hand, the results concerning topology optimization allow the obtaining of optimal and actual structures. Firstly, the SIMP-All method solution for three-dimensional cases is found. The solution is then shown to be part of the physical limits called "Hashin-Shtrikman bounds" and a comprehensive solution for the SIMP-All method is proposed. These results are compared with the SIMP method in two and three dimensions. Finally, all the new knowledge is applied to rewrite part of the Swan code.

In conclusion, from the refactoring of the "Codi Cante", sufficient tools have been developed to understand essential knowledge of programming by a modern engineer. Besides that, it has been possible to obtain a general equation for the SIMP-ALL method. Not only does the generalized equation allow to find optimal results, but the generated structures are also possible in the reality. Furthermore, the equation has been implemented in the Swan program using the new programming knowledge, thus allowing an easy correction, understanding and efficiency of the code.

Acknowledgments

I would like to thank my parents and friends for supporting me with this endeavour and always cheering me up to improve myself everyday. I will always be indebted to them and feel blessed to have them as my family.

A big thanks to the faculty of Aerospace Engineering at the ESEIAAT, which has been part of my life for 4 years, in which I have learned countless concepts and have allowed me to achieve astonishing goals.

Finally, the author wishes to express his deepest gratitude to his supervisor Dr. Alex Ferrer Ferre for his guidance, encouragements, criticism, advice and insight throughout the research. I could not have accomplished what I have in the time I did without our many hours of meetings and discussions. Your guidance and support has been essential to the success of this work.

Contents

List of Figures

List of Tables

Chapter I. Introduction

1 Introduction

1.1 Aim

The objective of this project is to extend the capabilities of the topology optimization interpolation method "SIMP-ALL" to "N" dimensions, in order to design lightweight structures with any isotropic material. Furthermore, it is also intended to implement these results in the "SWAN" program [1] using modern programming tools.

1.2 Scope

The scope of the project includes:

- **Collaborative programming.** Tools used to allow several contributors of a coding project to program changes and updates without generating conflicts in the code. Within this point, it will be explained how to use "Git" and "GitHub" with their corresponding commands.
- **TDD.** It will be discussed how to create a code that verifies the correct functioning of the main program while using refactoring techniques.
- **Clean code.** Within this point, it will be explained how to write a code following a certain style.
- **Refactoring.** It will be discussed how to restructure the source code to make it more efficient without changing its external behaviour.
- **OOP.** The structure of a class, how a class interacts with others and its objects and methods will be explained.
- **Topology optimization.** It will be discussed what topological optimization is, what methods it has and how it works.
- **SIMP-All formulation in 'N' dimensions.** The necessary mathematical expressions will be developed in order to expand the capabilities of the SIMP-ALL method to 'N' dimensions cases.
- **Analysis of results.** The results obtained by the SIMP and SIMP-ALL methods for the 2 and 3 dimensional cases will be compared.
- **SWAN modification.** New expressions will be incorporated into the SWAN code to allow solving 3D cases with the SIMP-ALL method.
- **Hashin-Shtrikman Bounds.** A qualitative and quantitative description of these bound will be carried out.

The scope of the project does NOT include:

- Explaining concepts of topology optimization that are not directly related to the development of this work.
- Writing a complete topology optimization code. The "SWAN" code will be used, and modified by adding new functions where necessary.

1.3 Requirements

Since this project combines programming tools and computational methods, it is necessary to use the following programs to carry it out.

- GitHub in order to update the versions of the code.
- MATLAB to develop and run the codes.
- Overleaf so as to write the final report of the project.
- Word to write the Project charter.
- Dropbox in order to store the required information.
- Lucid so as to create UML.
- GiD to create 2 and 3 dimensional meshes and analyse the results.

1.4 Justification

Programming skills.

The development of this TFG allows whoever may be interested, to be trained and acquire knowledge in relation to the modern programming tools that are currently used in the software industry. These tools are procedures that provide an increment in the efficiency of teamwork, using for example: GitHub; allowing to detect errors more quickly, using among others TDD; and, these facilitate the structuring of the code in a more optimal way, with OOP. Moreover, these programming tools are independent of programming languages, making their scope wider.

Topology optimization.

Topology optimization is an outstanding mathematical method used to obtain the optimal structure, in terms of geometry, subjected to boundary conditions, whose capabilities are enough to satisfy the constraints. It can be applied in some industries, such as aerospace or automotive, that aim to reduce the weight of their components in order to have a better performance over-all.

SIMP interpolation.

In recent years, the most widely used interpolation method of topology optimization has been the so-called SIMP. Nevertheless, it has been seen that its use is limited, since convergence to an exact geometry is not always guaranteed. This method is based on approximating the interpolation curve to a third-degree penalty function. Unfortunately, as previously mentioned, this method presents problems, mainly when the curve exceeds the "Hashin-Shtrikman" limits [2]. Consequently, the optimal geometry presents intermediate values resulting in a non-clear boundary.

SIMP-All interpolation.

To improve these deficiencies, the supervisor of this project has proposed an alternative method [3]. This new method is called SIMP-ALL, and is based on the concept of topological derivative. Specifically, it approximates a tangent curve at the extremes of the "Hashin-Shtrikman" bounds. This improvement avoids the SIMP's problem, since the curve always falls between the "Hashin-Shtrikman" limits. However, so far this method can only be applied in two-dimensional cases, and what this project allows is to give scope to its application in 3D dimensions. Thus, topology optimization can be then used in new study cases.

Chapter II. Modern programming tools

The second chapter is composed of sections 2, 3, 4 and 5. These have been developed in parallel to the "CodiCante" 's refactoring. Concepts of object oriented programming, cloud-backed work, test driven development and clean code have been applied and discussed in these sections. The chapter aims to provide the tools to be able to modify the Swan code professionally at a later stage.

2 Git

During the development of a program it is possible that a programmer may need to use a part of code previously discarded, or needs the approval of senior programmers for a new code. Moreover, one might face problems that other engineers have already encountered. Is it possible to have control over the changes and versions that have been done during development? The answer is yes, Git, it is an extension that allows the user to solve all of them [4]. In this section the main functionalities of Git will be presented.

2.1 Repository

Before using the Git tool, it is important to understand where files will be stored. The repository is defined as a place or receptacle where things are deposited or stored.[5] For the final application of this project, it will refer where the files will be saved or backed up.

Types of repositories

Repositories can be classified according to their use.

- **Bare Repositories**. These repositories are used to share the changes that are made by different developers. The users can neither create new files nor modify the original files of these repositories.
- **Non-bare Repositories**. These type of repositories have the opposite behavior to bare repositories. The developers can create or modify the files within them.

A distinction between remote and local repositories will be made, in order to give an easier explanation of the following sections.

- **Cloud repository**. Any file or folder stored in a repository that requires the Internet to access to it. These repositories are typically Bare repositories for non owners. Nevertheless, these can be non-bare repositories for owners of the project.
- **Local repositories**. Any file or folder stored in a repository that do not require remote access to access it. By definition, these are non-bare repositories.

Stages of a file in a repository

Once a group of files and folders are added to a repository, these documents can have different stages.

- **Untracked**: Any file that is not under Git's knowledge.
- **Tracked**: Opposite definition of untracked. This file can be a modified version or the latest version of a file.
- **Staged**: A file that is added to a repository or is modified, and is waiting to be committed.
- **Committed**: Files whose changes have been saved via commitment.

Cloning repositories

It is the process to copy the content from an existing repository (e.g. cloud repository) to a new one (e.g. local repository). By default, cloning a repository creates a non-bare repository. In

order to use the cloning action, some Git commands are required. The actual aim of cloning a repository is to enable the user to not be dependent on the internet while working on the files.

2.2 Git

It has been presented how the data will be stored, but not the management of it through the repositories. Git is a command line tool, which allows the user to have a better control over the files via 'code snapshots'. It also provides a distributed revision control system to use previous versions of the same code [4]. To run this tool, some commands will be necessary. These commands will be explained in section 2.4. An installation of Git is also needed.

Git scheme of communication

The following figure shows how the files are moved between and within repositories. Looking at the figure, the usual file direction is clockwise. It is relevant to mention that Git doesn't only track files, but also moves them between repositories.

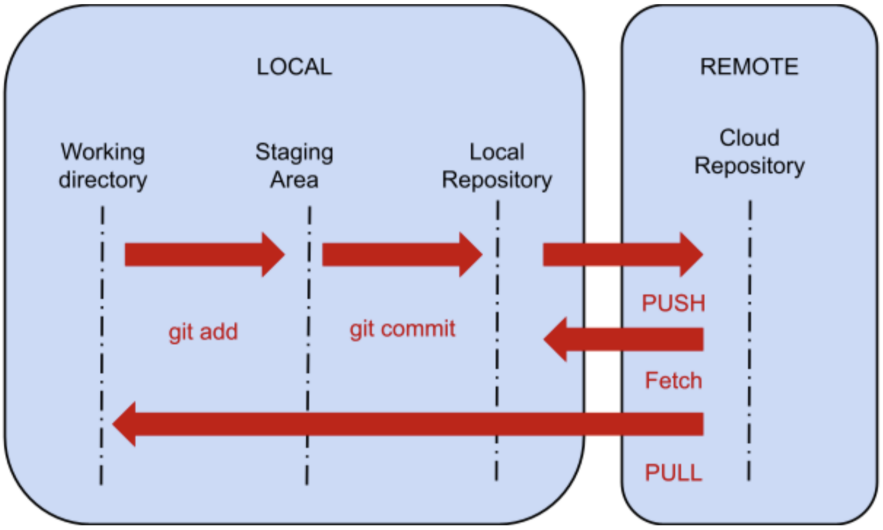

Figure 1: Git working tree

The main actions and the possible file status are explained below.

- **Tracked file**. File whose current version is controlled by Git.
- **Working directory**. Any document in the computer not tracked by Git.
- **Commit**. Action carried out to save the current state of a file. The file will be storaged in the local repository.
- **Staging Area**. Opposite definition of 'Working directory'.
- **Push**. Action that creates a copy of the local repository to the cloud repository.
- **Fetch**. Comparison of files and folders between the cloud and local repositories.
- **Pull**. Action that downloads a copy of the files and folders from the 'cloud repository' to the 'Working directory'.

How does it work?

Git automatically checks the tracked files for changes. If a change is detected, the modified file will be able to be committed. But before a file can be committed, a tracking has to be made [4]. It means that the file is moved from the 'Working directory' to the 'Staging Area', see Figure 1. This action is shown on the left side of Figure 2, where files 'A' to 'D' are added to the staging area. From this point, any change to the documents will be recorded by Git.

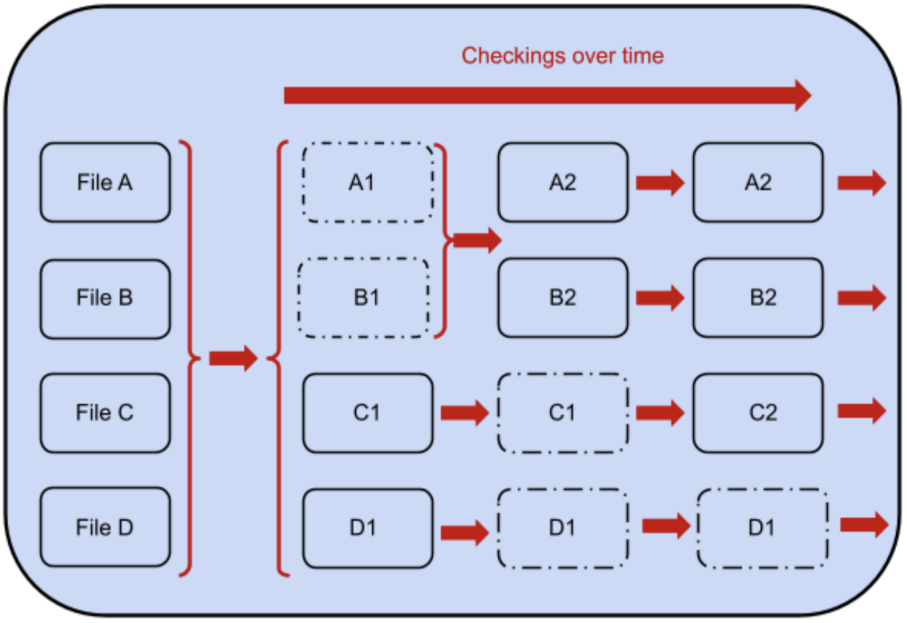

Figure 2: Storing data as snapshots of the project over time

Furthermore, a distinction of two cases can be made. First case, when files 'A' and 'B' are committed at the same time. This is the common case using the command "*Git commit -m 'XXX'*", this command will be explained in section 2.4. Using it, Git basically takes a picture of what all your files look like at that moment and stores a reference to that snapshot. To be efficient, if files have not changed, Git doesn't store the file again. Second case, where file 'C' is committed but the other files aren't. This case is possible with programs which has Git integrated e.g. Matlab. Git only takes an snapshot of the file you have previously selected.

2.3 GitHub

GitHub is a company that offers a cloud-based Git repository hosting service. Not only does it provide access control and several collaborative features for developers, but it also allows the communication with local repositories by means of using Git commands [6] [7].

GitHub's features

This service provides many features, but the most important ones are explained below.

- **Source code distribution**. The programmers can share their code with the community to be evaluated. These reviewers can also download the source code and experiment freely without modifying the original project (Forking a project).

- **Pull request**. Once some changes have been made to the code and approval is desired, it is possible to request it via GitHub. This request is sent to the project owner, who can approve it.
- **Merging**. If approval is accepted, the branch of the new code can be merged with the original.
- **Review of commits**. It is possible to track all the revisions that have been made to a file. Not only will it show the date, but also the sentence that describes the commit.

Even though GitHub offers a range of possibilities, it will only be used as a cloud repository. It will provide us with a copy of the final application of this project, in case of local repository malfunction.

Figure 3: GitHub company logo
Source: GitHub logo [8].

GitHub's interface

The following figures show the website GitHub interfaces that are mainly used to carry out the programming projects.

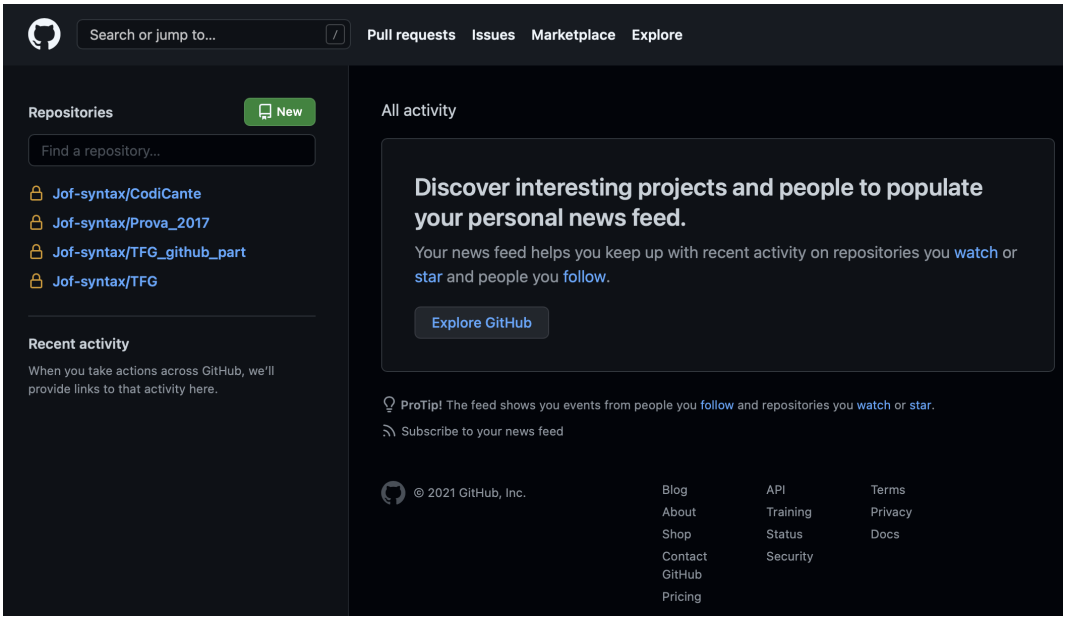

Figure 4: Main GitHub interface

On the left side of the Figure 4, it can be seen how the user's cloud repository is classified. It is also possible to use the 'New' button to create a new repository. As mentioned above, a 'Pull request' action can be applied, this utility can be seen at the top of the same picture.

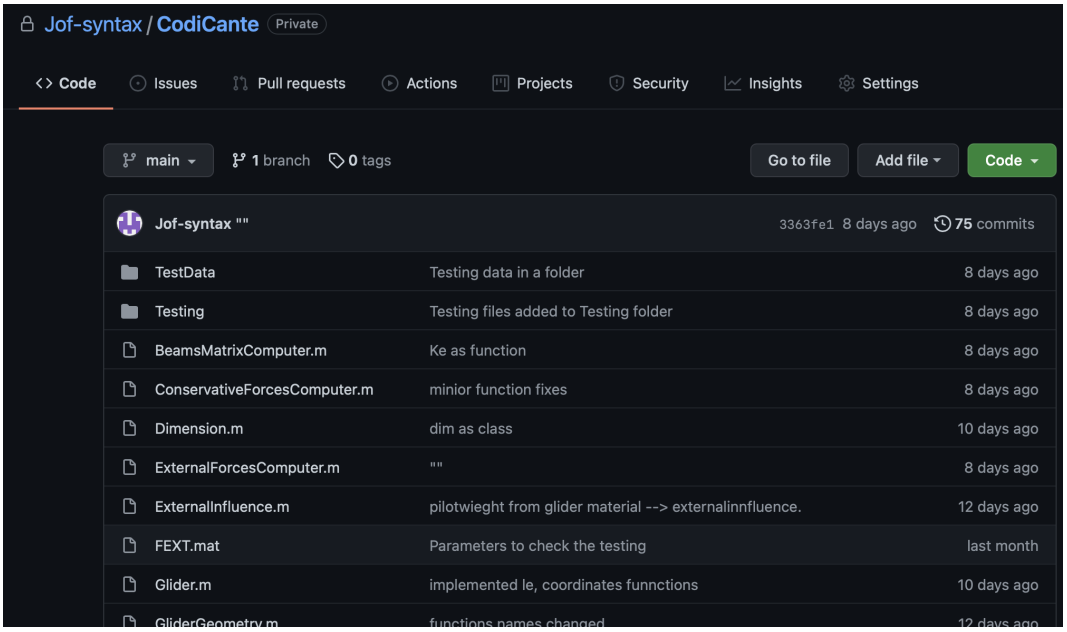

Figure 5: List of files inside a GitHub repository

Inside one of the repositories mentioned before, Figure 5 can be obtained. Not only is it possible to visualize the documents included in the repository, but also the commits made during development. On the other hand, the features from GitHub's website are also available throughout GitHub Desktop [9].

2.4 Git and GitHub commands

The following commands are used in a terminal, these allows the user to download, save, record files and their versions and save them. It's possible to manipulate the files locally or locally-remotely, that is why a distinction will be made between them [10] [4].

Local management

The following terminal instructions can be used to manipulate the files locally, in order to start the Git commands, stored revisions, review them or other local usage. This file management is carried out between the staging area and the local repository, as it can be seen in Figure 1.

- **Git init**: This command creates an empty Git repository, where files can be added or a re-initialization of an existing one can be made.
- **Git add .'XXX'**: Add file contents to the staging area, to prepare the content staged for the next commit. Once it is there, the file can be committed. In this instance 'XXX' represents the name of the file.
- **Git status**: Displays the working tree status. This command not only shows the differences between the index file and the current branch that the user is working on, but also paths in the working tree that are not tracked by Git.
- **Git commit -m 'XXX'**: Save the changes to the local repository, it contains the current contents of the file and the given log message describing the changes ('XXX').

- **Git log**: Displays the commit log and its ID.
- **Git reset –mixed 'XXX'**: Resets the current state of the file to another specified state. 'XXX' is the ID to identify the version that the user wants to change to.

Local-Remote management

The next commands are used to interact between the local repository and the cloud (GitHub). In general, these inputs are used to create a back-up of the existing local files.

- **Git clone 'URL'**: Clones a repository, normally a remote one, into a newly created local directory. It also creates remote-tracking branches for each branch in the cloned repository. In this case 'URL' is the link where the remote repository is stored.
- **Git fetch**: Compares objects and references from a remote repository to the local repository. If there is a difference, a Git pull will be requested.
- **Git push**: Updates the files in the remote repository from the files in the local repository. It also sends the commit log and its versions.
- **Git pull**: Downloads the remote files to the working directory.

The following figure, is a sample of how to use these commands in 'Terminal'. The keyboard inputs, 'Git init' and 'Git status', are underlined for a better understanding.

Figure 6: Example of using 'Git init' and 'Git status' in terminal

The description of these commands has the purpose to demonstrate the scope of possibilities offered by this extension. However, these commands will not be used in the final application of this project. Instead, the Matlab and GitHub Desktop implementation of them will be employed.

2.5 Branches

Another way to utilise the collaborative programming tools is by means of branches. These are incorporated within the Git tool, and they allow the commitment of different changes made by different developers into the main code. The aim of the branching is to allow each programmer to branch out from the original code base and isolate their work from others [11] [12].

For instance, during the development of a program, the team of programmers have a chain of commitments denominated master branch (code base), see the first row in Figure 7. Nevertheless, the code base has to be modified in order to implement new functionalities by the developers. Furthermore, the code can not be modified by all the team simultaneously and the external behaviour

of the master branch is not intended to be changed during the modifications. Branching can be used to solve it. Using the Git tool, each of the programmers can work with a copy of the main branch. In the Figure 7, only one copy of the master branch is represented, named 'branch A'. The developers can either work with this new branch or with another branch, e.g. branch 'B'. Once the modifications have been made, tested and committed, these changes can be implemented in the master branch by a merging command, as seen in Figure 7. As a result of branching, the master branch has been updated without changing its external behaviour and different developers have worked in it at the same time.

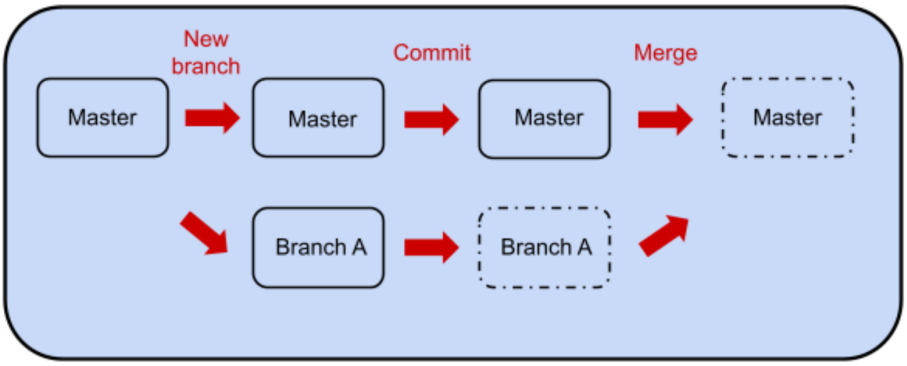

Figure 7: Sample of the creation, modification and merging of a branch without conflicts

It is important to note that neither the number of branches nor the number of developers that can be working on each branch are limited. Even though, each programmer can modify his branch without affecting the other branches, only one programmer can be working on specific line in a branch at the same time. Besides that, if many branches are being modified and then merged, conflicts may occur during the merging process. See the conflicts subsection for a better understanding of the issue.

Commands

The next terminal Git commands can be used to create, manipulate or review the branches of the project's code. [12]

- **Git branch 'XXX'**: Creates another branch with the name defined in the quotation mark.
- **Git branch**: Shows the current branch name.
- **Git checkout 'XXX'**: To switch from the current branch to another, the other is chosen by 'XXX'. The programmer only modifies the branch where he is.
- **Git merge 'XXX'**: Updates the current branch, normally the master branch, by adding the changes made in the 'XXX' branch.
- **Git branch -d 'XXX'**: Deletes the branch selected within the quotation mark.

Conflicts

During the merging of different branches some conflicts may occur because of the differences between them. The Figure 8 shows a basic conflict generated by the divergences of the 'Master' and the 'A' branches.

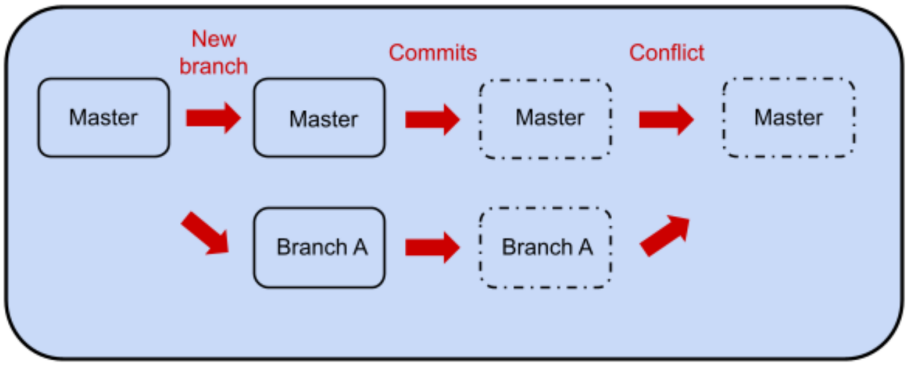

Figure 8: Schematic example of a conflict case

In this case, a branch 'A' has been created from the master branch. However, both branches have been modified in the same lines before the merging. Since Git uses snapshots of the code, it will not know what to do with the divergence. Git will have paused the merging process and will be asking the user to solve the conflict. If the programmer writes his codes using a code editor software, e.g. Visual Studio Code, the software will show the options to choose between the code from branch 'A' or from the 'master' branch. On the other hand, if the developer uses Terminal to apply the Git tool, the programmer must change the parts that are different between the codes, and then merge again creating a new commitment during the process.

2.6 Git and GitHub using Matlab

As previously explained in the branches conflicts, depending on the code editor software the Git experience may vary. Using MATLAB is an example of this, because the commands do not need to be typed by the user. In this subsection, the main characteristics of Git within MATLAB will be shown.

Source Control

Git and MATLAB are related by the source control menu. It can be accessed by a right-click over the script name and then a click over the 'Source Control' option. The following figure ought to appear.

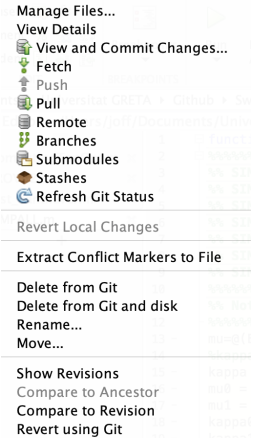

Figure 9: Source control options for a file tracked by Git

By means of these options, the commands explained in subsections (2.4) and (2.5) can be used in a more simplified way. For instance, the commands can be executed by a left click over the option, instead of writing the command in Terminal. An example showing the use of this menu can be seen at the end of this subsection.

File status

The following table, sum up the different types of status a file can have. The status can be seen in the 'current folder' section of the main MATLAB window. On the other hand, each of these status can be modified using the source-control option and applying the right option.

Sign type	.	+	■	●
Meaning	File not added by git	File added by git but not commited	File commited with modifications	File commited without modifications

Table 1: Possible Status of a file within MATLAB

Example of Git within Matlab

In order to show all the characteristics used during the development of this project, an example of them has been made, see Appendix section 1.

3 Object Oriented Programming

Object oriented programming (OOP) is a computer programming model that organizes software design around data, known as objects, rather than functions and logic. Each object can have different attributes and external behaviour, which allows the programmer to create unique structures with an increased performance [13].

Looking at Figure 10, it is possible to see how the structure from an OOP and no-OOP code differs. Firstly, object oriented programming allows the grouping of actions by importance and similarity, whereas a function scheme is organized as a consecutive sequence of tasks without any hierarchy. While in a simple scheme the information is moved only forward, OOP permits the movement of information to higher or lower levels in the hierarchy. Besides that, a normal scheme has to compile and execute all the files associated to the process, however, object oriented programming only compiles the necessary function for each case of execution. Furthermore, since the functions are grouped by types, it is easier to find a malfunction in the code.

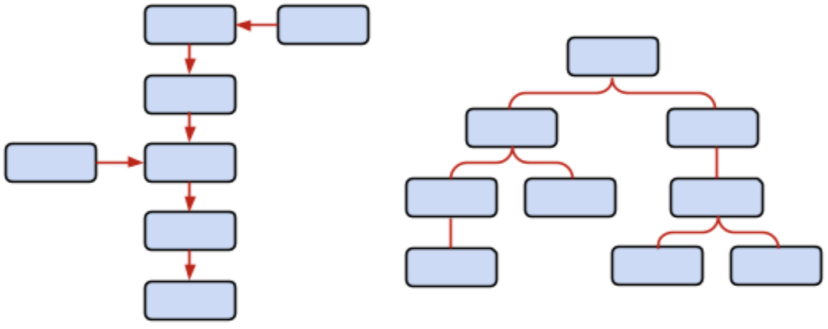

Figure 10: Functions scheme vs OOP scheme

3.1 How does it work?

Globally, OOP organizes the blocks from Figure 10, known as classes, as a hierarchy. However, each of these classes are composed by objects, which at the same time have attributes, see Figure 11.

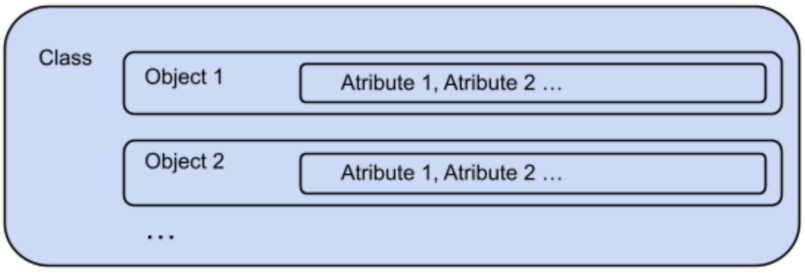

Figure 11: Composition of a class

Class

A class is defined as an extensible program-code-template composed of functions and parameters, which at the same time is composed of objects. It is possible to define different types of classes as described in subsection (3.3). Moreover, neither the number objects nor the number of attributes are limited in a class.

Objects

The objects can correspond to a real-world or an abstract entity. An object can have the form of a function, a parameter or another class. The main ones are described below.

- **Properties:** In the same class, these allow the definition of workable data. These can be other classes or parameters. Furthermore, objects can only be used in non-static methods, see subsection (3.3).
- **Methods:** Functions in a class are defined in methods, which describe the behaviors of an object and the capabilities of a class. Each method can contain different functions, which share the same attributes.

Attributes

Attributes describe the range of accessibility that a property, method or user may hold. For instance, if a method is wanted to be kept private respect a user, by means of the attributes it is possible to impose this restriction.

3.2 Four principles

The pillars of object oriented programming can be summarized in four principles, see following figure.

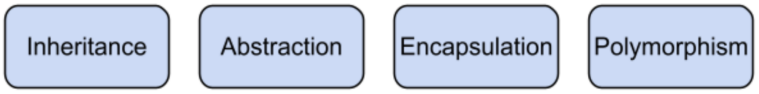

Figure 12: Principles of OOP

On the one hand, the **inheritance** principle allow the creation of relation between classes [14]. The main relation is the parent and child relation, where the child subclasses can inherit the features from the parent class. For instance, the parent class can describe methods and properties that can be further used by its child classes. It allows the programmer to have a method used in different child classes centralized in a parent class.

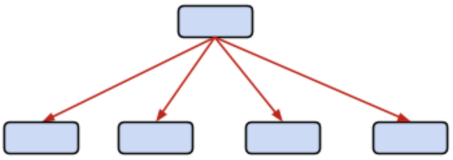

Figure 13: Inheritance case, where the parent class (top rectangle) has a method and/or property used in its child classes (button rectangles). Since the method and/or property is centralized, the programmer only has to edit one script instead of four.

On the other hand, the **abstraction** allows the objects only to reveal internal mechanisms that are relevant for the use of other objects, hiding any unnecessary implementation code. This concept can help developers make more easily additional changes or additions over time [13].

The **encapsulation** principle permits the programmer to show selected information to the user and contain the information in an object [14]. For instance, if a specific formula is necessary to compute a result but is not intended to be shown, the formula can be set as private. It means that the user can use the formula, but does not know its expression. These attributes are set in the definition of the class.

Finally, **polymorphism** means designing objects to share behaviors [15]. It can be understood as the program adapts the scripts compilation for each execution case. Using this principle, it is possible to use an abstract method in a parent class, but describe it in a child class. It allows the creation of a different method for each subclass, which can be executed by the parent's class. The aim of this principle is the avoidance of the compilation of unnecessary scripts, since only one or few child classes will be executed.

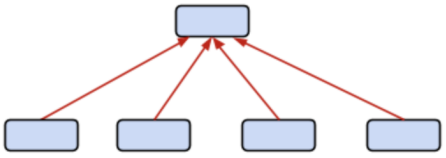

Figure 14: Polymorphism case, where the parent class (top rectangle) has a method to be executed but the definition is made in the child classes (bottom rectangles). Depending on the inputs of the problem, it will be compiled only one subclass. Since not all the subclasses have been compiled, the program has become more efficient and has adapted to the specific scenario.

3.3 OOP in Matlab

In this subsection the main characteristics and structure of a class in Matlab will be discussed. It starts with the general structure used in Matlab so as to define a class. Afterwards, the main attributes and features that can be added to the class are explained.

General class structure in Matlab

A generalized class structure for Matlab is provided below.

```
classdef ClassName % Handle, value or inheritance

   properties % (Attribute1 = value1, Attribute2 = value2,...)
      PropertyName
   end

   methods % (Attribute1 = value1, Attribute2 = value2,...)

      % Constructor
      function obj = ClassName(arg1,...)
         obj.PropertyName = arg1;
         ...
      end

      % Functions dependent on objects
      function ordinaryMethod(obj,arg1,...)
         ...
      end

   end

   methods (Static)
      % Functions not dependent on objects
      function staticMethod(arg1,...)
         ...
      end

   end

end
```

Looking at the previous script, the code begins with the definition of the class type and name, followed by the properties and finally the methods. It is noteworthy to mention that the structure does not have to be followed strictly. Nevertheless, since the objective of OOP is to reorganize the program, it is interesting to keep it.

Handle, Value and Inheritance Classes

The handle and value attributes are used to specify the type of object returned by the class [16]. A value class constructor returns an object that is associated with the variable to which it is assigned. Basically, if the returned variable is further modified in another script, Matlab will keep the original return stored in a copy. On the other hand, a handle class constructor returns a handle object that is a reference to the object created. In this case the data from executed classes will not be kept in memory after modification [17]. Furthermore, it is possible to stipulate the relation of child and parent by means of 'SubClassName' < 'Parent Name' attribute.

Properties

The next Matlab property attributes can be used to create privacy roles, enable the development of more advanced relations between classes and generate constant values among others.

- **Access:** It allows the creation of different privacy levels between the properties and other classes. This attribute is followed by public (unrestricted access), protected (access from class or subclasses) or private (access by class members only) options. If no access is defined, Matlab understands the property as public access.
- **Constant:** An invariant property in Matlab is a parameter that can not change its value during the execution of the program.
- **Abstract:** It is a parameter that can be used in a class, but its value is not defined on the same script. Otherwise, it must be defined on a child class.

The property attributes in Matlab can be combined with other attributes. For instance, a constant value can have a private, protected or public access. In order to allow all the possibilities, Matlab allows the definition of different types of properties within the same script. On the other hand, other attributes can be used in Matlab, a complete list can be seen in [18].

Methods

Similarly to the properties, the methods can have attributes and they can be used in different combinations creating different methods in the same class. The following list consist of the main attributes used in the development of this thesis.

- **Static:** When a method is defined as static, it means that no objects will be necessary so as to execute it.
- **Access:** It allows the creation of different privacy levels between the methods and other classes. This attribute is followed by public (unrestricted access), protected (access from class or subclasses) or private (access by class members only) options. If no access is defined, Matlab understands the method as public access.
- **Abstract:** It is a type of method that can be used in a class, but its value is not defined on the same script. Otherwise, it must be defined on a child class.

It is remarkable that the previous list is composed by the main attributes used in this thesis. A complete list containing all the possibilities for methods attributes in Matlab can be seen in [19].

Constructor

By default, Matlab executes a public method named as the class, e.g. 'ClassName(cParams)'. The method is known as the constructor and is the first part of the script that uses the inputs (cParams). If no constructor is defined, the class must be called as a struct. For instance, a class named 'ClassName()' has a method called 'init()' but it does not have a constructor, it should be called as 'ClassName.init()'.

Example of object oriented programming in Matlab

In order to show all the OOP characteristics used during the development of this project, an example of them has been developed and presented in Section 2 of the Appendix.

4 Test Driven Development

Test Driven Development (TDD) is a technique created to avoid problems during the development of a program. It reduces the number of bugs, implements only a sufficient number of functions and structures the code in a way that allows for future improvements [20]. The process of this method can be summarized in the following figure.

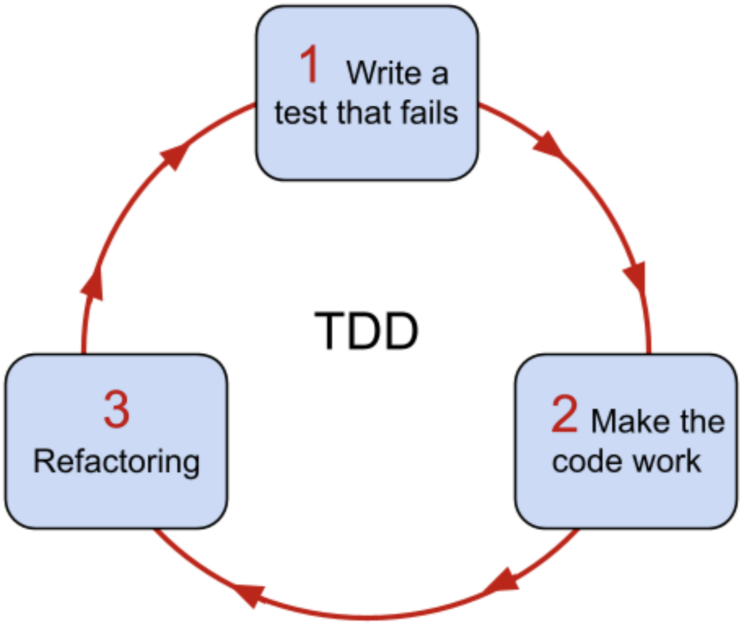

Figure 15: Test Driven Development process

The technique can be divided into two phases, the first one concerning the correct functionality of the code, and the second one focus on the minimization of unnecessary implementations.

Test First Development

- *1. Write a test that fails.* The first step is write tests that check the correct functionality of the inputs and outputs. Since the scripts that transform the inputs into the outputs have not been created yet, the tests will fail. It is remarkable that this step is called 'red', because a failed test is usually shown as red (See the following figure).

```
TestGliderAnalyser -->  FAILED.
```

Figure 16: Result of testing 'GliderAnalyser.m' as failed test

- *2. Make the code work.* Once the required tests have been created, the necessary scripts which have to transform the inputs into outputs and pass the previous tests, are created. Similarly to the first step, this step is named 'green', because a passed test is usually shown as green, see the following figure.

```
TestGliderAnalyser -->  PASSED.
```

Figure 17: Result of testing 'GliderAnalyser.m' as passed test

Refactoring

- *3. Refactoring.* Finally, this step is focus on reducing the redundancies of the code, where the source code is improved without changing its external behaviour. It is further discussed in the subsection (4.3).

Remark: It is noteworthy to mention that, since most of the code used during the development of this thesis was already created, in other subjects or by other developers, TDD can not be applied as previously described. Instead, a variation of it has been used. In this modified development technique, the test has been generated once the code was functional. Furthermore, after each refactoring improvement, the code has been tested in order to check the correct functionality.

4.1 Types of test

It this subsection, the two main types of testing schemes will be discussed.

Static Stored Data

It is the basic scheme used to verify the correct functionality of some scripts. It basically executes them using a known set of inputs. The scripts generate some results (outputs) and these are compared with stored expected results. Finally, the outcome of this comparison can be successful or unsuccessful. The process is sum up in the following figure.

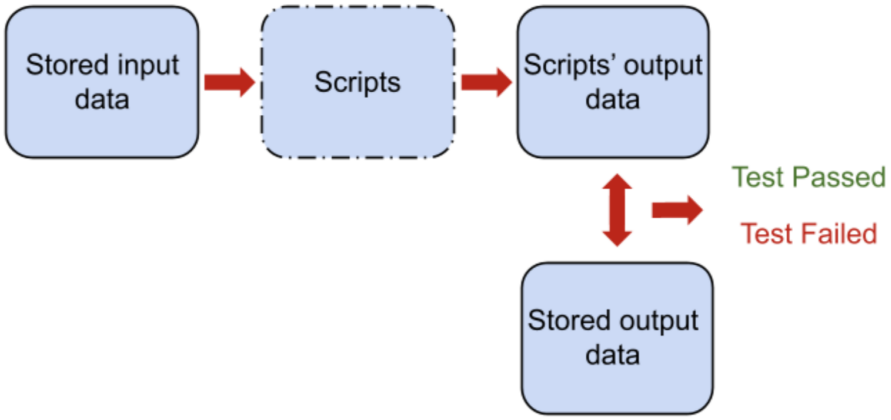

Figure 18: Static stored data scheme

It is remarkable that the stored output data can be a set of results created to be the target of the TDD or the first result obtained once the script is considered functional but has not been refactored yet. On the other hand, the tests are considered passed only if the outputs match. Contrary, the tests are considered failed if the outputs do not match. These cases are recovered in the following table.

Test Passed	Test Failed
The test has been satisfactory because both outputs are the same.	The scripts haven't passed the test because the outputs do not match.

Table 2: Possible outputs for the static stored data scheme, where the green and red indicate successful and unsuccessful tests, respectively

Dynamic Stored Data

The application of this scheme is mainly in databases development [21]. Not only has it the same capabilities as the static stored data scheme, but it also allows to identify if the scripts are outdated. Similarly to the previous scheme, this one has the same aim to analyse the correct functionality of scripts. Nevertheless, it also compares the inputs data, which means that the test checks if other parts of the database have been improved, e.g. by removing the definition of a parameters in a table.

The dynamic stored data scheme is summarized in the following figure.

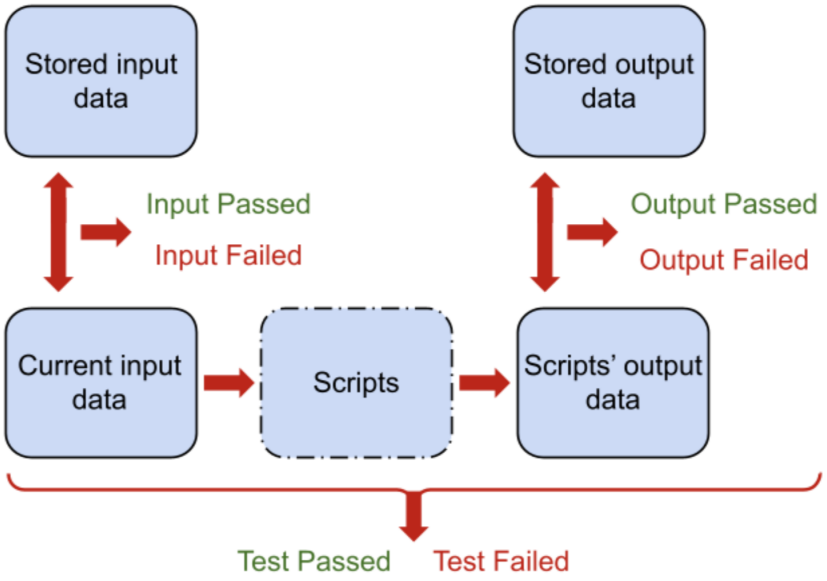

Figure 19: Dynamic stored data scheme

The tests are considered passed only if the inputs and outputs sub-tests are satisfactory. On the other hand, the test are considered failed if one or both of the sub-tests are unsuccessful. Moreover, the tests specify in which sub-test the code has failed. These possible cases are recovered in the following table.

Input sub-test Output sub-test	Input sub-test Output sub-test	Input sub-test Output sub-test
Both the inputs and outputs are the same compared to the stored ones. Therefore, the test is satisfactory.	The test has failed because the scripts have a malfunction that cause the unsatisfactory result of the outputs.	The failed test is caused by the unsatisfactory comparison of the inputs.

Table 3: Possible outputs for the dynamic stored data scheme, where the green and red indicate successful and unsuccessful sub-tests, respectively

It is remarkable that the last case from Table 3 is mainly caused by the addition or remove of extra parameters in the database. In any case, it means that the scripts work wrongly, but are possibly outdated.

In contrast with the other scheme, this one allows more versatility. For instance, these tests usually come with the option to modify the input and output stored data, whereas the static stored

data scheme maintains it constant. This upgrade is necessary in cases that the input sub-test has failed, e.g. the ResultSet [22] has lost a parameter that used to be, and the input, output data and scripts need to be modified.

4.2 Code coverage

The question that tries to satisfy the code coverage is: 'Do the tests take into account all the possibilities?'. Basically, it measures how many lines are executed (covered) during the run of a test respect to the total number of executable lines in a code.

How does it work?

Code coverage runs the test scripts, these at the same time run the codes that are going to be covered. For instance, one of these codes can be summarized as the figure below. Before the code is executed the code coverage only knows the existence of the file and the number of executable lines (6 in this case). After the execution of this script, the code coverage knows which lines have been executed. In this example, 4 executed and 2 not executed lines, therefore the coverage is 4/6 $\Rightarrow$ 67 % of code coverage.

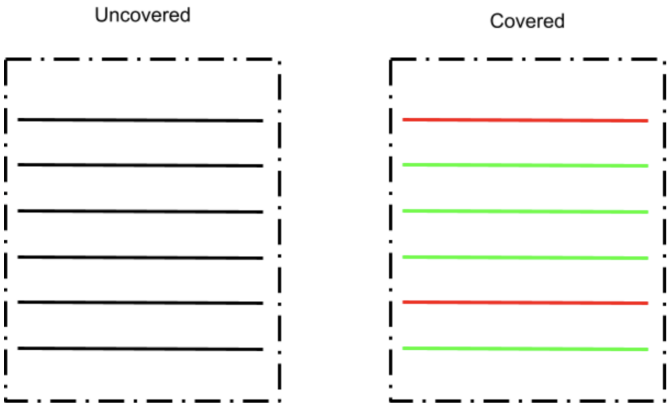

Figure 20: Script before and after code coverage

In the end, the code coverage generates a report with the percentage of executed lines and a pop-up in red and green of the executed and not executed lines. It can be seen in subsection (4.5).

How to compute it using Matlab?

In same cases, e.g. Matlab, the results are provided as a summarized table with a percentage of the covered code. Nevertheless, the computation of this table is obtained by means of external imports. The following functions are provided as a sample of how to structure the code in order to generate the code coverage in Matlab [23].

The following function generates a plugin with the report of the covered functions. This functions call the 'Solvertest.m' class and executes its test functions.

```
function TestRunner()
    % Import of TestSuite, TestRunner and CodeCoveragePlugin
    import matlab.unittest.TestSuite
    import matlab.unittest.TestRunner
```

```
    import matlab.unittest.plugins.CodeCoveragePlugin
    % Execute each test
    suite = TestSuite.fromClass(?SolverTest);
    runner = TestRunner.withTextOutput;
    runner.addPlugin(CodeCoveragePlugin.forFolder(pwd));
    result = runner.run(suite);
    % Show results
    table(result)
end
```

On the other hand, the following script calls the functions that will be tracked by the code coverage. It is remarkable that the methods used in this class are of the type 'Test', and the class itself is a subclass of the 'matlab.unittest.TestCase'. Furthermore, the structure inside the called function is the same as the one explained in subsection (3.1) (Appendix).

```
classdef SolverTest < matlab.unittest.TestCase

    methods (Test)

        function testGliderAnalyser(testCase)
                % Obtain current results
                InputData = load('TestData/Input data/TestClassDataGliderAnalyser.
    mat').cParams;
                test = GliderAnalyser(InputData);
                test.compute();
                test.plot();
                close all;
                actSolution = test;
                % Load stored results (expected)
                expSolution =     load('TestData/Outputdata/
    ResultTestGliderAnalyser.mat').test;
                % Verify the expected and current results
                testCase.verifyEqual(actSolution,expSolution);
        end

    end

end
```

It is noteworthy to mention that the number of covered functions is not limited. In the methods sections of the previous code different test functions can be covered at the same time. However, if a tested functions is part of a larger tested function and both of them are in the methods section of the 'SolverTest.m' class, only the larger influences.

4.3 Refactoring

Refactoring is the process used to simplify a code without adding, removing or changing its external capabilities. This process is not as precisely defined as the TDD, where the programmer has to follow preset steps. Instead, in this process different approaches can be made. Nevertheless, the most relevant approach is taking one step at a time and testing after each change. These steps can include but are not limited to the remove of duplication, simplify conditional expressions, move features between objects in order to better distribute functionality among classes and reduce the actions of each function to just one [24], in short, follow the rules of clean code.

Using the refactoring process, the redundancies of the code are minimized. Other developers

can easily understand what the code does. Since the functions are specialized for a unique task, these can be recycled in other projects. Besides that, it can help in the maintaining of the program. Finally, not only is the code reliable because it must have passed the tests, but also because unnecessary features are not implemented.

4.4 UML

Unified Modeling Language (UML) is a graphic scheme used to easy understand the structure of a code. It is mainly used during a refactoring, not only does it allow all classes involved to be detected at a glance, but also shows their hierarchy [25].

The scheme is composed of class-boxes and the relations between them, where the lowest level of shown information is the 'class'. The basic block can be seen in Figure 21. Inside the box, the public properties and methods are written. Furthermore, around the box there's an area designated for the union symbols, which indicate the relation of other classes to the current box. On the other hand, these boxes are sorted following a hierarchy, where the top one is the main class of the program. The pattern follows with the constructed and associated classes, followed by their constructed and associated classes, and so on.

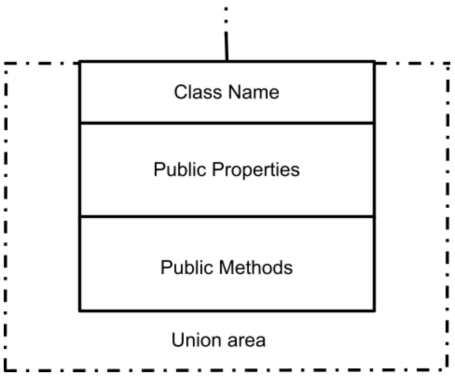

Figure 21: Class box scheme for UML

The union between the classes can be done by means of three different symbols. The composition symbol, black rhombus, is used when one class is constructed in a higher hierarchy class. The association symbol, white rhombus, is used when the methods of the class box are used in another class but it is not constructed there. Finally, the white triangle indicates the relation between child-parent. The following image shows the symbol for each case, respectively.

Figure 22: Types of union for the UML box

By definition, the composition symbol can only be used once per class box, whereas the other symbols more than once. It is because a class can only be constructed one time, but it can have different parents and its methods can be used in different classes.

4.5 Example of testing within Matlab

In order to show all the characteristics used during the development of this project, an example of them has been made, see Appendix section 3.

5 Clean Code

Clean code is a philosophy created by Robert C. Martin and followed by software engineers so as to avoid structural software problems. These troubles are described in the following subsection [26] [27] [28]. Even though these problems are generic, the cause can be minimized by Martin's philosophy.

Symptoms of a bad code

- **Fragility.** It refers to software that tends to crash in many places every time it is changed. One consequence is that the software is almost impossible to maintain because the origin of the problem is hard to find and any modification adds new bugs.
- **Immobility.** This is the inability to reuse code in other parts of a project because the code in question has too much redundancies.
- **Rigidity.** It happens when a change made to the code causes a cascade of subsequent changes in dependent modules.
- **Viscosity.** Two cases are differentiated. The viscosity of the design, when the original structure of the code is lost by the ease of an inefficient solution (that works). On the other hand, the environment viscosity happens when the programmers are inefficient developing their tasks (e.g. checking few files takes hours).

Philosophy

In order to outweigh these inconveniences, Robert C. Martin proposes a series of norms and standardisations. The idea behind it, is to create code with readability, changeability, extensibility and maintainability by any member of the team.

A complete discussion in this topic can be seen in [29]. Nevertheless, the most relevant rules followed in this thesis are provided below.

- Reduce complexity as much as possible.
- Indentation and tabulation when necessary.
- Keep configurable data at high levels.
- The name of a class has to be a noun and the first letter is uppercase. (e.g. GliderAnalyser)
- Function's name has to be a verb and the first letter is lowercase. (e.g. computeDynamicSolver)
- Maximum 120 lines per class.
- Words in the same string are separated by uppercase letter.(e.g. helloWorld)
- Choose representative names. (computeCoordinates - ~~computeXYZ~~)
- Functions only do one thing.

Note: The clean code practise can be seen in any line of code in this work.

Chapter III. Topology optimization

The chapter begins with section 6, which is an introduction to the topology optimization and its concepts. Then, in section 7, a simplified version of the topology optimization problem is formally formulated. Next, in section 8, the topological derivative for the problem is found. On the other hand, the main methods used to define an interpolation function (e.g. SIMP) and the shape of these functions are discussed in section 9. This section is followed by section 10, where the (HS) bounds are discussed and also formally defined in the Annex section 4. Next, in section 11, the core of the thesis starts with the application of the SIMP-All method in two and three dimensions. It is proven that these results are part of the previously defined HS bounds, which allow to express a generalized expression for the SIMP-All method in section 12. The results are implemented on the Swan code in section 13. Finally, the resultant refactored method (SIMP-All) is compared with the SIMP method in two and three dimensions in section 14.

6 Topology optimization introduction

Topology optimization (TO) is a mathematical method used to optimize material layout within a given design space. In the thesis, the optimization is mainly focused on structures. Nevertheless, the method can be applied in other fields, such as fluid, heat or geologist problems. TO allows the creation of new design structures that can be obtained by means of removing material.

The following figure shows an example of a cantilever beam problem in three dimensions, before and after the topology optimization method.

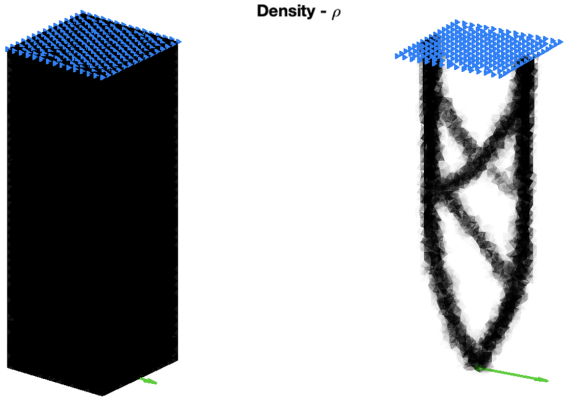

Figure 23: Topology optimization for a Cantilever beam problem in 3D, before TO method (left) and after TO method (right)

The subsections below aim to explain important concepts and parameters in order to comprehend the following sections.

Lamé parameters: Lamé's first parameter, Poisson's ratio, Young, Shear and Bulk modulus

The shear (μ) and bulk (κ) modulus are used to measure the elastic shear stiffness of a material and measure of how resistant to compression that material is, respectively. See the following figure.

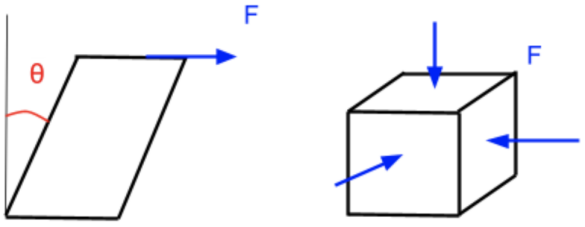

Figure 24: Representation of the shear mesure (left) and bulk measure (right)

These measures are related with the Lamé's first parameter (λ) as follows.

$$\lambda = \kappa - \frac{2}{N}\mu \tag{1}$$

where 'N' denotes the problem dimension. Besides that, the Lamé's first parameter (λ) is mainly used to describe the compliance of a structure, as it is shown in the following sections.

On the other hand, the Young modulus (E) and the Poisson's ratio (ν) are important parameters to be highlighted. While the Young modulus is a mechanical property that measures the tensile of a solid material, the Poisson's ratio measures the expansion or contraction of a material. Besides that, the Poisson ratio has values between [-1,0.5], whereas the Young modulus from 0 to infinity.

Furthermore, the following relations are used in the development of the thesis.

$$\begin{aligned} \mu &= \frac{E}{2\,(1+\nu)} \\ \kappa &= \frac{E}{N\,(1+\nu-N\,\nu)} \end{aligned} \tag{2}$$

where the (μ) definition has been obtained from [3], whereas the (κ) definition has been obtained from [3] and [30].

The topology optimization problem

Basically, the objective of topology optimization is to minimize a function called 'Cost' subjected to some constraints. The 'Cost', also known as objective function, relates the problem domain (Ω) with the design parameters. Furthermore, the domain is composed of two subdomains Ω_o and Ω_1 thorugh the characteristic function $\mathcal{X}$. For a structural problem, the cost function ($\mathcal{J}$) depends on the displacements field (U) and ($\mathcal{X}$), which is the vector of integer nodal design variables. Therefore, $\mathcal{J}(\mathcal{X}, U)$. Formally, the problem is defined as follows.

$$\begin{aligned} \min_{\mathcal{X},U} \quad & \mathcal{J}(\mathcal{X},U) \\ s.t. \quad & K(\mathcal{X})\,U = F(\mathcal{X}) \\ & g_i(\mathcal{X},U) \leq 0 \end{aligned} \tag{3}$$

where the characteristic function ($\mathcal{X}$), depending on the domain, has the values 0 (Ω_o) or 1 (Ω_1). K is the stiffness matrix, F is the vector force and g_i are the constraints (e.g Volumetric). Since ($\mathcal{X}$) and (U) are related with an equation, the problem is also known as **PDE-constraint optimization problem**. Besides that, the previous topology optimization definition applied to structures is denoted in this thesis as the Comprehensive Topology Optimization Problem (CTOP).

Shape functionals

Shape functionals are operators that maps a function, such as the compliance, the perimeter or the volume, into a number. In the thesis, they are represented as $\mathcal{J}$. Note that are defined as shape functionals the objective function and constraints used during the optimization process.

It is first presented the compliance functional, which is

$$\mathcal{J} = \int \sigma\varepsilon d\Omega = \int \varepsilon^T \mathbb{C}\varepsilon d\Omega \tag{4}$$

where σ is the stresses, ε the strains and $\mathbb{C}$ the constitutive tensor.

Second, the perimeter is a key element during topology optimization in order to avoid the formation of very small beams and therefore ensuring manufacturability of the designs. Furthermore, it penalizes the "grey" areas of the domain, making it useful for the convergence to the solution. It is defined as

$$P(\Omega) = \frac{1}{2\epsilon\, P^*} \int (1 - L_\epsilon(\mathcal{X})\mathcal{X})\, d\Omega \tag{5}$$

where P^* is a reference perimeter.
Finally, the volume is another shape functional and an essential variable for topology optimization. Since topology optimization aims to obtain shapes that reduces cost and weight, it can be accomplished by means of the volume.

$$V(\Omega) = \int_\Omega \rho d\Omega \tag{6}$$

being ρ the density.

Topological derivative

The topological derivative shows how a shape functional changes when an infinitesimal amount of mass is added or subtracted from the structure. If an infinitesimal amount of mass is added to the structure, the process is known as stiffer inclusion insertion. On the other hand, if an infinitesimal amount of mass is subtracted from the structure, the process is known as weaker inclusion insertion [31].

No existence of solution

In topology optimization, the optimal solution does not exist. This statement can be understood with the following example.

Let a control surface (S) in two dimensions be a square, where external forces are applied at two not adjacent sides. Moreover, a constraint regarding the volume is added, which imposes S = 0.5. The topology optimization results can be as the following illustration.

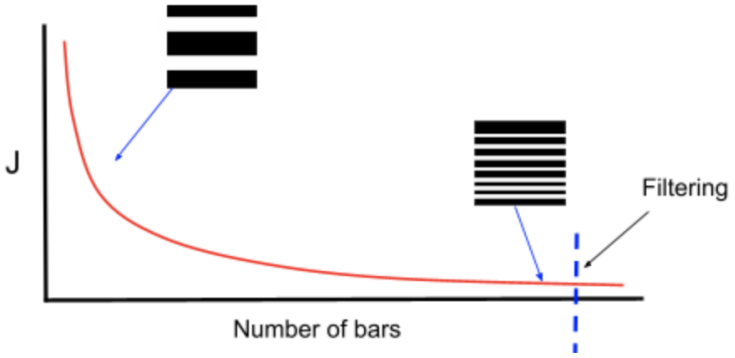

Figure 25: Optimal results from the same problem

For the same problem, different optimal solutions are obtained. It is remarkable that less $\mathcal{J}$ implies a lower compliance, which is better. Nevertheless, the best solution is obtained at the infinity (endless bars), where the micro-structure is infinitesimally small. There is no lower limit.

The solution to the problem is to add a filter. It can be accomplished by adding an extra constraint (e.g. perimeter constraint). Consequently, the number of solutions will be reduced to some point, see Figure 25.

7 Relaxed topology optimization formulation

The Comprehensive Topology Optimization Problem (CTOP) lies on the definition of the characteristic function $\mathcal{X}$. Nevertheless, this formulation presents difficulties that are beyond this work. Therefore, the characteristic function has been replaced from (CTOP) by a less restrictive density-like function $\rho \in \mathbb{R} := L^\infty(\Omega, [0,1])$, which in contrast with the characteristic function can take intermediate values. Thus, the problem can be formulated in a relaxed form of the comprehensive topology optimization problem as follows.

Definition. Let a fixed control volume Ω be composed of Ω_o and Ω_1, where $\Omega_o \cup \Omega_1 = \Omega$ and $\Omega_o \cap \Omega_1 = \emptyset$. On the other hand, let the displacements $(\overline{u})$ acting in the Dirichlet boundary Γ_D and the stress forces $(\bar{t})$ acting in the Neumann boundary Γ_N, be part of the boundary of Ω, where $\partial\Omega = \overline{\Gamma_D \cup \Gamma_N}$ and $\Gamma_D \cap \Gamma_N = \emptyset$. Finally, let the admissible functions $(\mathcal{U})$ and the admissible variations $(\mathcal{V})$ be

$$\mathcal{U} := \{\phi \in H^1(\Omega) : \phi|_{\Gamma_D} = \overline{u}_o\} \qquad \text{and} \qquad \mathcal{V} := \{\phi \in H^1(\Omega) : \phi|_{\Gamma_D} = 0\} \tag{7}$$

Definition. Allow the total potential energy associated to an isotropic and heterogeneous elasticity problem be

$$\int_\Omega \nabla^s u : \mathbb{C}(\rho) : \nabla^s u = \int_{\Gamma_N} \bar{t} \cdot u \tag{8}$$

where $\nabla^s u$ denotes the symmetric part of the gradient operator ∇ and 'u', which is the field's displacement. Furthermore, $(\bar{t})$ is the non-conservative force of the problem, which causes the deformations. Lastly, the constitutive tensor $\mathbb{C}(\rho)$ is a fourth order tensor described by the following expression.

$$\mathbb{C}(\rho) = 2\,\mu(\rho)\,\mathbb{I} + \left(\kappa(\rho) - \frac{2}{N}\,\mu(\rho)\right) I \otimes I \tag{9}$$

where 'N' denotes the problem dimension, $\mu(\rho) \in C^\infty([0,1],[\mu_o,\mu_1])$ and $\kappa(\rho) \in C^\infty([0,1],[\kappa_o,\kappa_1])$ are interpolation functions to be defined. Being μ_o, κ_o and μ_1, κ_1 the lamé parameters of Ω_o and Ω_1, respectively. Along with that, $\mathbb{I}$ and $I \otimes I$ represent the fourth- and second- order identity tensors.

The relaxed topology optimization problem is solved when: $u \in \mathcal{U}$ and $\rho \in \mathbb{R} := L^\infty(\Omega, [0,1])$ provide the minimum potential energy and the following constraints are fulfilled.

$$\begin{aligned} \int_\Omega \rho &\leq \; Volume \\ \int_\Omega \nabla^s u : \mathbb{C}(\rho) : \nabla^s \nu &= \int_{\Gamma_N} \bar{t} \cdot \nu \qquad\qquad \forall\, \nu \in \mathcal{V} \end{aligned} \tag{10}$$

Being (ν) an admissible test function.

On the other hand, the **gradient of the compliance**, can be obtained from equation (8) as

$$g(\rho) = -\nabla^s u : \mathbb{C}'(\rho) : \nabla^s u \tag{11}$$

This proof was developed in the work of [3].

8 Topological derivative

Definition. Let an unperturbed domain be $\Omega \subset \mathbb{R}^N$, where within it, a small inclusion $\mathcal{B}_\epsilon$ of radius $\epsilon > 0$ is placed at $\hat{x}$ and being 'N' the problem dimension. The new perturbed domain is $\Omega_\epsilon \subset \mathbb{R}^N$, hence the unperturbed domain is $\Omega \backslash \overline{\mathcal{B}_\epsilon}$.

Definition. Permit the shape functionals associated to the topologically perturbed and unperturbed domain, $\mathcal{J}_\epsilon(\Omega)$ and $\mathcal{J}(\Omega)$ respectively, be

$$\begin{aligned} \mathcal{J}_\epsilon(\Omega) &= \int_\Omega \nabla^s u_\epsilon : \mathbb{C}(\rho) : \nabla^s u_\epsilon + \int_{\Gamma_N} \bar{t} \cdot u_\epsilon \\ \mathcal{J}(\Omega) &= \int_\Omega \nabla^s u : \mathbb{C}(\rho) : \nabla^s u + \int_{\Gamma_N} \bar{t} \cdot u \end{aligned} \tag{12}$$

Remark. Since the problem deals with singular domain perturbations, the shape functionals $\mathcal{J}_\epsilon(\Omega)$ and $\mathcal{J}(\Omega)$ are associated to topologically different domains.

Remark. It is assumed that $\mathcal{J}_\epsilon(\Omega)$, the shape functional associated to $\mathcal{B}_\epsilon$, admits the following asymptotic expansion

$$\mathcal{J}_\epsilon(\Omega) = \mathcal{J}(\Omega) + f(\epsilon)\,\mathcal{T}(\hat{x}) + o(f(\epsilon)) \tag{13}$$

where the function $f(\epsilon)$ is always positive and $f(\epsilon \to 0) = 0$, in this work $f(\epsilon) = \pi\epsilon^2$ for two dimensions and $f(\epsilon) = \frac{4}{3}\pi\epsilon^3$ for three dimensions. Besides that, $\mathcal{T}(\hat{x})$ is the topological derivative, which in combination with $f(\epsilon)$ works as a first order correction of $\mathcal{J}(\Omega)$ to obtain $\mathcal{J}_\epsilon(\Omega)$. Finally, $o(f(\epsilon))$ imposes the compliance of the asymptotic expansion, and consequently the following limit for the topological derivative of $\mathcal{J}$ at $\hat{x}$ is possible.

$$\mathcal{T}(\hat{x}) = \lim_{\epsilon \to 0^+} \frac{\mathcal{J}_\epsilon(\Omega) - \mathcal{J}(\Omega)}{f(\epsilon)} \tag{14}$$

Taking into consideration the problem formulation and developing the standard topological derivative (14), $\mathcal{T}(\hat{x})$ takes the following form.

$$\mathcal{T}(\hat{x}) = -\nabla^s u(\hat{x}) : d\mathbb{C} : \nabla^s u(\hat{x}) \qquad \forall\, \hat{x} \in \Omega \tag{15}$$

It is remarkable that the topological derivative allows the variation of the compliance $\mathbb{C}$. When an infinitesimal mass is subtracted from the structure, the process is known as weaker inclusion insertion. Contrary, if an infinitesimal mass is added to the structure, the process is known as stiffer inclusion insertion.

9 Interpolation functions

In a relaxed topology optimization problem, the density changes during the iteration. During this process each element should be filled up with material (hard elements with the relative material volume equal to 1), no material (void elements with the relative material volume close to 0 but always positive) or intermediate values between 0 and 1. The final solution may contain these intermediate values, known as grey elements. The function used to parameterize the geometry is known as grey-scale density-like interpolation function [32].

9.1 Characteristics of grey-scale density-like interpolation function

The idea is to propose an interpolation function that reduces the grey elements. Besides that, the solution of the problem shall be an actual solution to be manufactured. It can only be accomplished by proposing an interpolation function that interpolates with an actual micro-structure. Taking it into consideration, the function should have the following shape.

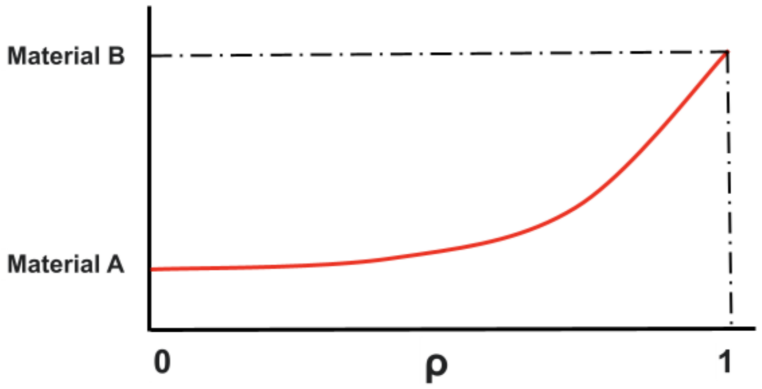

Figure 26: Grey-scale density-like interpolation function

It is remarkable that the 'Material A' from the previous graph has properties close to null but are always positive. It can not be zero so as to avoid invertibility of the problem. The interpolation scheme is known as void-interpolation. Nevertheless, if both materials are far from 0, the case is known as bi-material interpolation. On the other hand, the main features that an optimal interpolation function shall has, are provided below.

- **Smallest derivative on low relative density.** If an element is in this zone, it is not worth for the optimizer to propose greater relative densities, because the properties, mainly material A, do not change a lot. Therefore, the optimizer tends to have relative densities close to 0, which auto-penalizes itself.
- **Highest derivative on high relative density.** It implies that the optimizer has to pay a high cost in terms of structural properties when the relative density is reduced. Consequently, the optimizer tends to have relative densities close to 1, which auto-penalizes itself.
- **Describe an actual composite micro-structure.** In case an element has an intermediate value after the iteration process, the element has to be manufactured in the reality. It can be done by proposing an interpolation function inside physical bounds (e.g. Hashin-Shtrikman bounds).

9.2 SIMP method

The Solid Isotropic Microstructures with Penalization method (SIMP), is based on regularizing the problem and proposing a penalization interpolation function [3]. It combines the property of both materials as follows.

$$\mathbb{C}(\rho) = (1 - \rho^p)\mathbb{C}_o + \rho^p \mathbb{C}_1 \tag{16}$$

being p the penalization parameter, typically $p = 3$, $\mathbb{C}_o$ and $\mathbb{C}_1$ the constitutive tensors of Ω_o and Ω_1 respectively. It almost always met the first two constraint of a density-like interpolation function. However, the third restriction might not be met, generating micro-structures impossible to be found in the reality.

9.3 SIMP-All method

It is an improved SIMP method, based on the topological derivative and the Hashin-Shtrikman (HS) bounds. The idea is to propose a rational function tangent to the lower (HS) bound, where the relative density is low. And tangent to the upper (HS) bound, where the relative density is high. It allows to almost always meet the first two requirements of a density-like interpolation function. Since the rational function is always within the (HS) bounds [3], the final structure can always be understood as a micro-structure. The Hashin-Shtrikman bounds are further discussed in section 10. The SIMP-All rational functions are shown below.

$$R = \left\{ f \in C^{\infty}([0,1],[f_o,f_1]) \;\mid\; f(\rho) = \frac{A\rho^2 + B\rho + C}{D\rho + 1} \right\} \tag{17}$$

where A, B, C and D$\neq -1$ are parameters to be determined imposing the following restrictions.

$$f(0) = f_o, \quad f'(0) = \dot{f}_o, \quad f(1) = f_1, \quad f'(1) = \dot{f}_1 \tag{18}$$

Being $\dot{f}_o$ and $\dot{f}_1$ the topological derivative at Ω_o and Ω_1, respectively. The formalities are shown in subsection (12.2).

9.4 SIMP and SIMP-All functions examples

Figure 27 is an example of the SIMP and SIMP-All interpolation functions. It is obtained by means of the SIMPALL(E1,E0,nu1,nu0, N) function, which can be found in: `https://github.com/Jof-syntax/TFG_Jofre`.

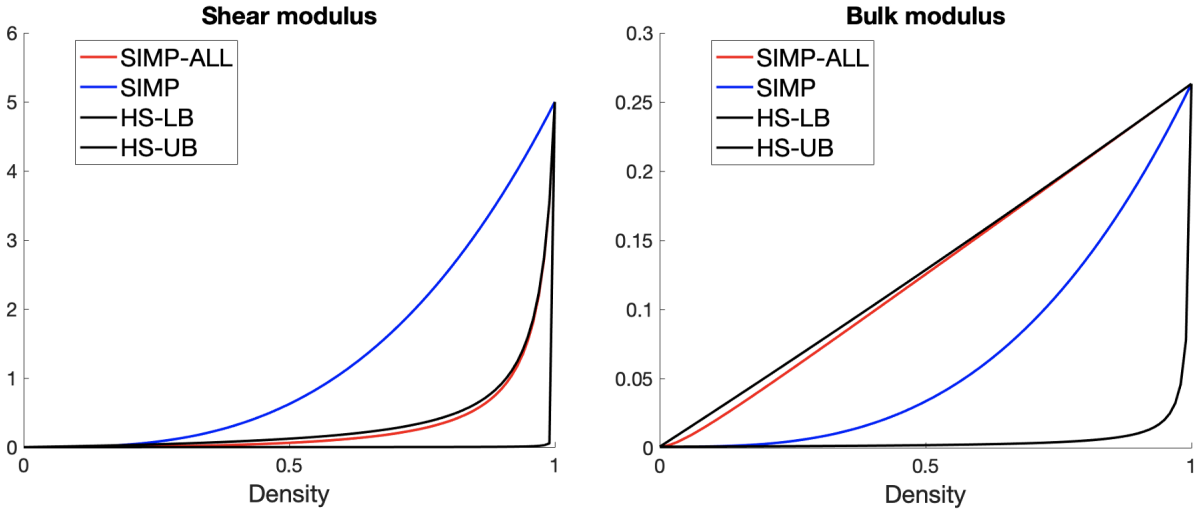

Figure 27: 2D case, where $E_o = 10^{-3}$, $\nu_o = 1/3$, $E_1 = 1$ and $\nu_1 = -0.9$

It is noteworthy to mention that in the shear function, the SIMP method is outside the Hashin-Shtrikman (HS) bounds, which means that the grey elements will not have an actual physical meaning. On the other hand, the SIMP-All function is within the (HS) bounds, which means that can be obtained as a lamination of the two materials. Regarding the bulk functions, SIMP and SIMP-All satisfy the main features of an interpolation scheme.

10 Hashin-Strikman Bounds

The Hashin-Strikman (HS) bounds are a combination of inequalities that describe the valid region of isotropic and homogeneous two-phase composite materials, these mathematical expressions were developed in [33]. Nevertheless, this section is based on the recompilation work [34], where more results concerning the bounds are discussed. The section is composed of two parts, the first one regards the deduction of the bounds and its features qualitatively, and the second part discusses the reality beyond the bounds. Besides that, the formal deduction of the Hashin-Shtrikman bounds shown in [33] can be seen in the Appendix section 4.

10.1 The bounds

Problem definition

The aim of the work is to allow the creation of isotropic and homogeneous two-phase composite material. Let the new material A^* be composed by the mix of two isotropic phases, where their Hooke's laws A and B are

$$\begin{aligned} A &= 2\,\mu_A\,\mathbb{I} + \left(\kappa_A - \frac{2}{N}\,\mu_A\right) I \otimes I \\ B &= 2\,\mu_B\,\mathbb{I} + \left(\kappa_B - \frac{2}{N}\,\mu_B\right) I \otimes I \end{aligned} \tag{19}$$

being μ_A, κ_A and μ_B, κ_B the lamé parameters of the phases A and B respectively. Besides that, $\mathbb{I}$ and $I \otimes I$ represent the fourth- and second- order identity tensors and 'N' the dimension of the problem. The characteristics of (19) can be summarized as isotropic, not null, and well ordered expressions, i.e.

$$0 < \mu_A \leq \mu_B, \qquad 0 < \kappa_A \leq \kappa_B \quad \text{and} \quad A \leq B \tag{20}$$

On the other hand, let G_θ be the set of all possible laminated materials stable under lamination, obtained by a combination of two isotropic phases. Furthermore, let the subset $L_\theta \in G_\theta$ be composed by all the possible lineal combinations of A and B in proportions θ and $(1-\theta)$.

Finally, let the sequence describing the actual micro-structure $\chi(x)$ be denoted as A^*. It is well known by [34], that if $\chi(x)$ depends on a single space coordinate, as in this case, then there exists an explicit formula for the homogenized Hooke's law A^* which depends only on A, B, θ (the weak limit of $\chi(x)$), and the direction of the oscillations of $\chi(x)$. Therefore, it is possible to obtain a whole family of composite by varying θ and the direction of the lamination.

Laminated composites formulation

First, let an smaller subset of L_θ be defined as a sequential laminates. Such laminates are obtained by an iterative process, where, the previous laminated is laminated again with a single pure phase. Consequently, two sequences are obtained for any lamination direction.

- Sequential laminates L_θ^+ with core A and matrix B. The sequence is obtained by laminating A and B, then, the resultant is laminated once again with the pure phase B. The laminate is laminated again with the pure phase B, etc, where the maximum number of laminates is defined by the rank-p.
- Sequential laminates L_θ^- with core B and matrix A. The sequence is obtained by laminating A and B, then, the resultant is laminated once again with the pure phase A. The laminate

is laminated again with the pure phase A, etc, where the maximum number of laminates is defined by the rank-p.

The number of possible lamination for A^* in $L_\theta^\pm$ is defined by the rank-p, whose limit is under the following constraint [34].

$$\text{rank-p} \leq (N+3)(N+2)(N+1)N/24 \tag{21}$$

If the rank-p of a sequential laminate (A^* in $L_\theta^\pm$) is greater than the previous restriction, the lamination is not possible to be obtained as a finite combination of A and B Hooke's laws.

Intermediate results

Important results to be highlighted in the work of [34] are:

- The minimum number of laminates to create a composite when the problem dimension is $N = 2$ is rank-p $= 3$.
- The minimum number of laminates to create a composite when the problem dimension is $N = 3$ is rank-p $= 6$.
- The sequential laminates L_θ^- and L_θ^+ are proven to be closed for any A and B Hooke's law.

It is important to note from the last point that any arbitrary combination of A and B Hooke's laws will be between the L_θ^- and L_θ^+ sequential laminates. For instance, the sequence begins by laminating A and B, then, the resultants is laminated once again with the pure phase A. Next, the resultant laminated with the pure first B.

Hashin-Strikman energy bounds

Since the sets $L_\theta^\pm$ are closed, the idea collected in [34] is to add and subtract a reference energy to the effective energy of A^*, till the effective energy is saturated to either one of these upper and lower bounds. The idea is proved with the Hashin-Shtrikman variational principal in [33], a simplified version is provided below. Even though, the formal deduction of the Hashin-Shtrikman bounds shown in [33] can be seen in the Appendix Section 4.

Theorem: *For any effective Hooke's law A^* in G_θ there exist two sequential laminates $A^+ \in L_\theta^+$ (i.e., with core A and matrix B) and $A^- \in L_\theta^-$ (i.e., with matrix A and core B), such that*

$$A^- \leq A^* \leq A^+ \tag{22}$$

in the sense of quadratic forms (of course, the sequential laminates A^+, A^- vary with A^*).

The previous theorem is proven by means of the Hashin-Shtrikman variational principle, where optimal bounds are obtain as a sum of energies. Firstly, the variational principle proves that $A^- \leq A^*$ by adding and subtracting a reference energy. In a similar manner, it is also proven that $A^* \leq A^+$. Therefore, it is shown that a isotropic and homogeneous two-phase composite material A^* can exist within the bounds and is stable under lamination.

By construction of $L_\theta^\pm$ and consequently $A^\pm$, there are inner bounds and composite materials obtained by lineal combination of the phases A and B where the matrix and core are not limited to just one phase. Within these bounds, the SIMPALL micro-structures $\chi(x)$ exists.

Finally, a direct application of the theorem are the original Hashin Shtrikman bounds on the

bulk and shear moduli of an isotropic two-phase composite material, whose Hooke's law can be generalized as

$$A^* = 2\,\mu^*\,\mathbb{I} + \left(\kappa^* - \frac{2}{N}\,\mu^*\right) I \otimes I \tag{23}$$

where the bulk and shear moduli, κ^* and μ^* respectively, satisfy the following HS inequalities

$$\begin{aligned}
\frac{\theta}{2\,\mu_1 - 2\,\mu^*_{UB}} &\leqslant \frac{1}{2\,\mu_1 - 2\,\mu_0} + \frac{(1-\theta)\,(\kappa_1 + 2\,\mu_1)\,(N-1)}{\mu_1\,(N^2+N-2)\left(\kappa_1 + 2\,\mu_1 - \frac{2\,\mu_1}{N}\right)} \\
\frac{1-\theta}{2\,\mu^*_{LB} - 2\,\mu_0} &\leqslant \frac{1}{2\,\mu_1 - 2\,\mu_0} + \frac{\theta\,(\kappa_0 + 2\,\mu_0)\,(N-1)}{\mu_0\,(N^2+N-2)\left(\kappa_0 + 2\,\mu_0 - \frac{2\,\mu_0}{N}\right)} \\
\frac{\theta}{\kappa_1 - \kappa^*_{UB}} &\leqslant \frac{1}{\kappa_1 - \kappa_0} + \frac{1-\theta}{\kappa_1 + 2\,\mu_1 - \frac{2\,\mu_1}{N}} \\
\frac{1-\theta}{\kappa^*_{LB} - \kappa_0} &\leqslant \frac{1}{\kappa_1 - \kappa_0} + \frac{\theta}{\kappa_0 + 2\,\mu_0 - \frac{2\,\mu_0}{N}}
\end{aligned} \tag{24}$$

Note: It is remarkable that these inequalities are defined by means of the porosity θ. Nevertheless, the work of this thesis has been developed in terms of the density ρ (the strong limit of $\chi(x)$), which is related to the porosity by $\theta = 1 - \rho$.

10.2 Points outside the boundaries

The bounds are generated after the compliance of some hypothesis and constraints. If a point is outside these boundaries, the definitions followed to prescribe the bounds will not be met. Therefore, the main implication can be summarized as:

On the one hand, the Hashin-Shtrikman variational principle will be violated, it means that the composites can not be defined as a sum of finite energy elements. Furthermore, since the HS bounds are part of $L^{\pm}_{\theta}$, being $L^{\pm}_{\theta}$ the limits of all possible combinations of composites, it is not possible to obtain a material A^* as a combination of A and B Hooke's laws. Consequently, the composites outside these bounds are not physically possible.

11 Generalized Simp-All parameters

In this section, the required parameters to apply the SIMP-All method for 2 dimensions and 3 dimensions will be obtained. Furthermore, it will be shown that the parameters not only are part of the Hashin-Shtrikman bounds, but they allow to define a generalized expression for them. On the other hand, a simplified expression for the (HS) bounds and its derivative will be found.

11.1 Simp-All η parameters for 2 dimensions

The topological derivative in $\hat{x}$ can be written in terms of the shear modulus ($d\mu$) and the bulk modulus ($d\kappa$) as follows.

$$\mathcal{T}(\hat{x}) = -\nabla^s u(\hat{x}) : d\mathbb{C} : \nabla^s u(\hat{x}) \qquad \text{with} \qquad d\mathbb{C} = 2d\mu\,\mathbb{I} + (d\kappa - d\mu)\; I \otimes I \tag{25}$$

However, this expression can not be applied without the parameters $d\kappa$ and $d\mu$. Since these parameters are unknown, the document [3] will be used. Firstly, $d\mathbb{C}$ is related with the polarization tensor $\mathbb{P}$ as $\mathbb{C} = \mathbb{C}_m : \mathbb{P}$. Being the $\mathbb{C}_m$ the constitutive tensor of the matrix domain.

$$\mathbb{P} = P_1\,\mathbb{I} + P_2\,I \otimes I \tag{26}$$

The parameters P_1 and P_2 have the following definitions.

$$\begin{aligned} P_1 &= \frac{(1+\beta)\,(\tau_1-\gamma)}{\beta\,\gamma+\tau_1} \\ P_2 &= \frac{(\alpha-\beta)\,(\gamma\,(\gamma-2\,\tau_3)+\tau_1\,\tau_2)}{2\,(\beta\,\gamma+\tau_1)\,(\alpha\,\gamma+\tau_2)} \end{aligned} \tag{27}$$

where the coefficients α, β, γ, τ_1, τ_2 and τ_3 are

$$\begin{aligned} \alpha &= \frac{1+\nu}{1-\nu}\,; & \beta &= \frac{3-\nu}{1+\nu}\,; & \gamma &= \frac{E^*}{E}\,; \\ \tau_1 &= \frac{1+\nu^*}{1+\nu}\,; & \tau_2 &= \frac{1-\nu^*}{1-\nu}\,; & \tau_3 &= -\frac{\nu^*\,(3\,\nu-4)+1}{\nu\,(3\,\nu-4)+1} \end{aligned} \tag{28}$$

For a 2D case, the Poisson coefficient (ν) and the elastic modulus (E) take the following definitions in terms of shear (μ) and bulk (κ).

$$\nu = \frac{\kappa-\mu}{\kappa+\mu} \tag{29}$$

$$E = \frac{4\,\kappa\,\mu}{\kappa+\mu} \tag{30}$$

Replacing these Lamé parameters in the defined coefficients (28), these become

$$\begin{aligned} \alpha &= \frac{\kappa}{\mu}\,; & \beta &= \frac{\kappa+2\,\mu}{\kappa}\,; & \gamma &= \frac{\kappa^*\,\mu^*\,(\kappa+\mu)}{\kappa\,\mu\,(\kappa^*+\mu^*)} \\ \tau_1 &= \frac{\kappa^*\,(\kappa+\mu)}{\kappa\,(\kappa^*+\mu^*)}\,; & \tau_2 &= \frac{\mu^*\,(\kappa+\mu)}{\mu\,(\kappa^*+\mu^*)}\,; & \tau_3 &= -\frac{(\kappa+\mu)\,(\kappa\,\mu^*-3\,\kappa^*\,\mu+4\,\mu\,\mu^*)}{2\,\mu\,(\kappa^*+\mu^*)\,(\kappa-2\,\mu)} \end{aligned} \tag{31}$$

Combining equation (27) with the new coefficients (31), it is obtained

$$\begin{aligned} P_1 &= \frac{2\,(\kappa+\mu)\,(\mu-\mu^*)}{\kappa\,\mu+\kappa\,\mu^*+2\,\mu\,\mu^*} \\ P_2 &= \frac{(\kappa+\mu)\,\left(\kappa^2\,\mu-2\,\kappa\,\mu^2+\kappa^2\,\mu^*-3\,\kappa\,\kappa^*\,\mu+\kappa\,\kappa^*\,\mu^*+4\,\kappa\,\mu\,\mu^*-2\,\kappa^*\,\mu\,\mu^*\right)}{2\,\kappa\,(\kappa^*+\mu)\,(\kappa\,\mu+\kappa\,\mu^*+2\,\mu\,\mu^*)} \end{aligned} \tag{32}$$

Defining

$$\begin{aligned} P_\mu &= P_1 \\ P_\kappa &= P_1+2\,P_2 \end{aligned} \tag{33}$$

Then

$$\begin{aligned} P_\mu &= \frac{2\,(\kappa+\mu)\,(\mu-\mu^*)}{\kappa\,\mu+\kappa\,\mu^*+2\,\mu\,\mu^*} \\ P_\kappa &= \frac{(\kappa+\mu)\,(\kappa-\kappa^*)}{\kappa\,(\kappa^*+\mu)} \end{aligned} \tag{34}$$

Furthermore, the shear and the bulk polarization coefficients can be defined following [1] as

$$\begin{aligned} q\mu &= \frac{d\mu}{\mu\cdot(\mu^*-\mu)} = \frac{P_\mu}{\mu-\mu^*} \\ q\kappa &= \frac{d\kappa}{\kappa\cdot(\kappa^*-\kappa)} = \frac{P_\kappa}{\kappa-\kappa^*} \end{aligned} \tag{35}$$

Note that these coefficients can be described in a general form as

$$q_i = \frac{i + \eta_i}{i \cdot (i^* + \eta_i)} \tag{36}$$

where the term 'i' is defined as the shear (μ) or bulk (κ). Insulating this equation by η_i

$$\eta_i = \frac{q_i \cdot i \cdot i^* - i}{1 - q_i \cdot i} \tag{37}$$

and combining equations (34) and (35) with η_i, equation (37), the following equations for the $\eta\mu$ and $\eta\kappa$ are obtained

$$\begin{aligned} \eta_\mu &= \frac{\kappa\,\mu}{\kappa + 2\,\mu} \\ \eta_\kappa &= \mu \end{aligned} \tag{38}$$

11.2 Refactoring of the Hashin-Shtrikman bounds in 2D

In this subsection, it will be proven that for 2 dimensions η, equations (38), are by themselves part of the Hashin-Shtrikman bounds. The first step is to recover the Hashin-Shtrikman bounds from [34], imposing N = 2. The deduction of these bounds can also be seen in Section 4 from Appendix.

$$\begin{aligned} \frac{\theta}{2\,\mu_1 - 2\,\mu^*_{UB}} &\leqslant \frac{1}{2\,\mu_1 - 2\,\mu_0} + \frac{(\kappa_1 + 2\,\mu_1)\,(1-\theta)}{4\,\mu_1\,(\kappa_1 + \mu_1)} \\ \frac{1-\theta}{2\,\mu^*_{LB} - 2\,\mu_0} &\leqslant \frac{1}{2\,\mu_1 - 2\,\mu_0} + \frac{\theta\,(\kappa_0 + 2\,\mu_0)}{4\,\mu_0\,(\kappa_0 + \mu_0)} \\ \frac{\theta}{\kappa_1 - \kappa^*_{UB}} &\leqslant \frac{1}{\kappa_1 - \kappa_0} + \frac{1-\theta}{\kappa_1 + \mu_1} \\ \frac{1-\theta}{\kappa^*_{LB} - \kappa_0} &\leqslant \frac{1}{\kappa_1 - \kappa_0} + \frac{\theta}{\kappa_0 + \mu_0} \end{aligned} \tag{39}$$

being θ the porosity.

Since the topology optimization problem has been expressed as a relaxed function of the density (ρ), the previous equations have to be replaced by $\rho = 1 - \theta$. On the other hand, the limit cases can be obtained by the insulation of the insertion parameters.

$$\begin{aligned} \mu^*_{UB} &= \mu_1 + \frac{1-\rho}{\frac{1}{\mu_0 - \mu_1} - \frac{\rho\,(\kappa_1 + 2\,\mu_1)}{2\,\mu_1\,(\kappa_1 + \mu_1)}} \\ \mu^*_{LB} &= \mu_0 - \frac{\rho}{\frac{1}{\mu_0 - \mu_1} - \frac{(\kappa_0 + 2\,\mu_0)\,(1-\rho)}{2\,\mu_0\,(\kappa_0 + \mu_0)}} \\ \kappa^*_{UB} &= \kappa_1 + \frac{1-\rho}{\frac{1}{\kappa_0 - \kappa_1} + \frac{\rho}{\kappa_1 + \mu_1}} \\ \kappa^*_{LB} &= \kappa_0 - \frac{\rho}{\frac{1}{\kappa_0 - \kappa_1} - \frac{1-\rho}{\kappa_0 + \mu_0}} \end{aligned} \tag{40}$$

Recovering the 2D η equations (38) and replacing them into the previous equations, it is possible to compress the bounders into the upper and lower cases as

$$\begin{aligned} f_{UB} &= F_1 + \frac{1-\rho}{\frac{\rho}{F_1 + \eta} + \frac{1}{F_0 - F_1}} \\ f_{LB} &= F_0 - \frac{\rho}{-\frac{1-\rho}{F_0 + \eta} + \frac{1}{F_0 - F_1}} \end{aligned} \tag{41}$$

where the shear and bulk H-S bounds are recovered by choosing the values described in the following table.

	μ_{LB}	μ_{UB}	κ_{LB}	κ_{UB}
f_0	μ_0	μ_0	κ_0	κ_0
f_1	μ_1	μ_1	κ_1	κ_1
η	$\eta\mu(\kappa_m, \mu_m, 2)$	$\eta\mu(\kappa_m, \mu_m, 2)$	$\eta\kappa(\kappa_m, \mu_m, 2)$	$\eta\kappa(\kappa_m, \mu_m, 2)$

Table 4: HS relation table for 2 dimensions cases

Equation (41) can be then refactored as

$$\begin{aligned} f_{UB} &= F_1 + \frac{(1-\rho)\,(F_0 - F_1)\,(F_1 + \eta_1)}{\eta_1 + F_0\,\rho + F_1\,(1-\rho)} \\ f_{LB} &= F_0 - \frac{\rho\,(F_0 - F_1)\,(F_0 + \eta_0)}{\eta_0 + F_0\,\rho + F_0\,(1-\rho)} \end{aligned} \tag{42}$$

It is possible to add and subtract the necessary terms, $F_0\,(1-\rho) + F_1\,\rho - [F_0\,(1-\rho) + F_1\,\rho]$, in order to obtain the following Hashin-Shtrikman generalized equation.

$$\begin{aligned} f_{UB} &= F_0\,(1-\rho) + F_1\,\rho + \left[-F_0\,(1-\rho) - F_1\,\rho + F_1 + \frac{(1-\rho)\,(F_0 - F_1)\,(F_1 + \eta_1)}{\eta_1 + F_0\,\rho + F_1\,(1-\rho)}\right] \\ f_{LB} &= F_0\,(1-\rho) + F_1\,\rho + \left[-F_1\,\rho + F_0\,\rho - \frac{\rho\,(F_0 - F_1)\,(F_0 + \eta_0)}{\eta_0 + F_0\,\rho + F_0\,(1-\rho)}\right] \end{aligned} \tag{43}$$

Expanding the terms inside the math brackets, it is obtained

$$\begin{aligned} f_{UB} &= F_0\,(1-\rho) + F_1\,\rho - \frac{\rho\,(1-\rho)\,(F_0 - F_1)^2}{\eta_1 + F_0\,\rho + F_1\,(1-\rho)} \\ f_{LB} &= F_0\,(1-\rho) + F_1\,\rho - \frac{\rho\,(1-\rho)\,(F_0 - F_1)^2}{\eta_0 + F_0\,\rho + F_1\,(1-\rho)} \end{aligned} \tag{44}$$

Consequently, a more general expression for the 2D Hashin Shtrikman Bounds is

$$f(\rho) = F_0\,(1-\rho) + F_1\,\rho - \frac{\rho\,(1-\rho)\,(F_0 - F_1)^2}{\eta + F_0\,\rho + F_1\,(1-\rho)} \tag{45}$$

11.3 Simp-All η parameters for 3 dimensions

The topological derivative in $\hat{x}$ can be written in terms of the shear modulus ($d\mu$) and the bulk modulus ($d\kappa$) as follows.

$$\mathcal{T}(\hat{x}) = -\nabla^s u(\hat{x}) : d\mathbb{C} : \nabla^s u(\hat{x}) \qquad \text{with} \qquad d\mathbb{C} = 2d\mu\,\mathbb{I} + \left(d\kappa - \frac{2}{3}d\mu\right) I \otimes I \tag{46}$$

However, this expression can not be applied without the parameters $d\kappa$ and $d\mu$. Since these parameters are unknown, the document [35] will be used. In this report, for a 3 dimensional case, the equation $d\mathbb{C}$ is expressed as

$$d\mathbb{C} = m_1(2\mathbb{I} + m_2 I \otimes I) \tag{47}$$

Where the parameters m_1 and m_2 are

$$m_1 = \frac{15\,\mu\,(\mu - \mu^*)\,(\nu - 1)}{(5\,\nu - 4)\,(2\,\mu - 2\,\mu^*) - 15\,\mu\,(\nu - 1)} \tag{48}$$

$$m_2 = \frac{(\lambda\ (\mu - \mu^*) - 5\,\mu\,\nu\ (\lambda - \lambda^*))\ (2\,\mu - 2\,\mu^*) + (15\,\lambda\,\mu\ (\nu - 1) - 2\,\lambda\ (5\,\nu - 4)\ (\mu - \mu^*))\ (\lambda - \lambda^*)}{(5\,\mu - 5\,\mu^*)\ (3\,\lambda\,\mu\ (\nu - 1) - \lambda\ (2\,\nu - 1)\ (\mu - \mu^*) + 3\,\mu\,\nu\ (\lambda - \lambda^*))}$$

The first step in order to apply the SIMP-ALL method is transform the parameters $m_1(\lambda, \mu, \lambda^*, \mu^*)$ and $m_2(\lambda, \mu, \lambda^*, \mu^*)$ from [35] in terms of shear and bulk.
In the same report, the relation between the Poisson coefficient, λ and μ for a 3D case is defined as

$$\nu = \frac{\lambda}{2\,\lambda + 2\,\mu} \tag{49}$$

The coefficient lambda (λ) in a 3D case, is related with κ and μ as follows

$$\lambda = \kappa - \frac{2}{3}\mu \tag{50}$$

Consequently, replacing the previous relations in equations (48), the parameters $m_1(\kappa, \mu, \kappa^*, \mu^*)$ and $m_2(\kappa, \mu, \kappa^*, \mu^*)$ are obtained

$$\begin{aligned} m_1 &= -\frac{5\,\mu\ (\mu - \mu^*)\ (3\,\kappa + 4\,\mu)}{9\,\kappa\,\mu + 6\,\kappa\,\mu^* + 12\,\mu\,\mu^* + 8\,\mu^2} \\ m_2 &= \frac{24\,k\,\mu^2 + 27\,k^2\,\mu + 18\,k^2\,\mu^* - 54\,\kappa^*\,\mu^2 + 40\,\mu^2\,\mu^* - 40\,\mu^3 - 27\,k\,\kappa^*\,\mu}{15\,\mu\ (\mu - \mu^*)\ (3\,\kappa^* + 4\,\mu)} + ... \\ &... + \frac{-18\,k\,\kappa^*\,\mu^* + 36\,k\,\mu\,\mu^* - 6\,\kappa^*\,\mu\,\mu^*}{15\,\mu\ (\mu - \mu^*)\ (3\,\kappa^* + 4\,\mu)} \end{aligned} \tag{51}$$

Taking into consideration equations (46) and 47, the following relations appear

$$\begin{aligned} d\mu &= m_1 \\ d\kappa &= m_1\,m_2 + \frac{2}{3}\,m_1 \end{aligned} \tag{52}$$

Replacing the parameters $m_1(\kappa, \mu, \kappa^*, \mu^*)$ and $m_2(\kappa, \mu, \kappa^*, \mu^*)$ into the previous equations, one obtains

$$\begin{aligned} d\mu(\kappa, \mu) &= -\frac{5\,\mu\ (\mu - \mu^*)\ (3\,\kappa + 4\,\mu)}{9\,\kappa\,\mu + 6\,\kappa\,\mu^* + 12\,\mu\,\mu^* + 8\,\mu^2} \\ d\kappa(\kappa, \mu) &= -\frac{(\kappa - \kappa^*)\ (3\,\kappa + 4\,\mu)}{3\,\kappa^* + 4\,\mu} \end{aligned} \tag{53}$$

On the other hand, the shear and the bulk polarization coefficients can be defined as

$$\begin{aligned} q\mu &= \frac{d\mu}{\kappa \cdot (\kappa^* - \kappa)} \\ q\kappa &= \frac{d\kappa}{\kappa \cdot (\kappa^* - \kappa)} \end{aligned} \tag{54}$$

These coefficients can be described in a general form

$$q_i = \frac{i + \eta_i}{i \cdot (i^* + \eta_i)} \tag{55}$$

where the term 'i' is defined as the shear (μ) or bulk (κ). Insulating this equation by η_i

$$\eta_i = \frac{q_i \cdot i \cdot i^* - i}{1 - q_i \cdot i} \tag{56}$$

Combining equations (53) and (54) with η_i, equation (56), the following equations for the $\eta\mu$ and $\eta\kappa$ are obtained

$$\begin{aligned} \eta\mu(\kappa,\mu) &= \frac{\mu\,(9\,\kappa+8\,\mu)}{6\,\kappa+12\,\mu} \\ \eta\kappa(\kappa,\mu) &= \frac{4\,\mu}{3} \end{aligned} \tag{57}$$

11.4 Refactoring of the Hashin-Shtrikman bounds in 3D

In this subsection, it will be proven that for 3D η, equations (57), are by themselves part of the Hashin-Shtrikman bounds. The first step is to recover the Hashin-Shtrikman bounds from [34], imposing N = 3. The deduction of these bounds can also be seen in the appendix section 4.

$$\begin{aligned} \frac{\theta}{2\,\mu_1-2\,\mu^*_{UB}} &\leqslant \frac{1}{2\,\mu_1-2\,\mu_0}+\frac{(1-\theta)\,(\kappa_1+2\,\mu_1)}{5\,\mu_1\left(\kappa_1+\frac{4\,\mu_1}{3}\right)} \\ \frac{1-\theta}{2\,\mu^*_{LB}-2\,\mu_0} &\leqslant \frac{1}{2\,\mu_1-2\,\mu_0}+\frac{\theta\,(\kappa_0+2\,\mu_0)}{5\,\mu_0\left(\kappa_0+\frac{4\,\mu_0}{3}\right)} \\ \frac{\theta}{\kappa_1-\kappa^*_{UB}} &\leqslant \frac{1}{\kappa_1-\kappa_0}+\frac{1-\theta}{\kappa_1+\frac{4\,\mu_1}{3}} \\ \frac{1-\theta}{\kappa^*_{LB}-\kappa_0} &\leqslant \frac{1}{\kappa_1-\kappa_0}+\frac{\theta}{\kappa_0+\frac{4\,\mu_0}{3}} \end{aligned} \tag{58}$$

being θ the porosity.

Since the topology optimization problem has been expressed as a relaxed function of the density (ρ), the previous equations have to be replaced by $\rho = 1-\theta$. On the other hand, the limit cases can be obtained by the insulation of the insertion parameters.

$$\begin{aligned} \mu^*_{UB} &= \mu_1+\frac{1-\rho}{\frac{1}{\mu_0-\mu_1}-\frac{6\,\rho\,(\kappa_1+2\,\mu_1)}{5\,\mu_1\,(3\,\kappa_1+4\,\mu_1)}} \\ \mu^*_{LB} &= \mu_0-\frac{\rho}{\frac{1}{\mu_0-\mu_1}-\frac{6\,(\kappa_0+2\,\mu_0)\,(1-\rho)}{5\,\mu_0\,(3\,\kappa_0+4\,\mu_0)}} \\ \kappa^*_{UB} &= \kappa_1+\frac{1-\rho}{\frac{\rho}{\kappa_1+\frac{4\,\mu_1}{3}}+\frac{1}{\kappa_0-\kappa_1}} \\ \kappa^*_{LB} &= \kappa_0-\frac{\rho}{-\frac{1-\rho}{\kappa_0+\frac{4\,\mu_0}{3}}+\frac{1}{\kappa_0-\kappa_1}} \end{aligned} \tag{59}$$

Recovering the 3D η equations (57) and replacing them into the previous equations, it is possible to compress the bounders into the upper and lower cases.

$$\begin{aligned} f_{UB} &= F_1+\frac{1-\rho}{\frac{\rho}{F_1+\eta}+\frac{1}{F_0-F_1}} \\ f_{LB} &= F_0-\frac{\rho}{-\frac{1-\rho}{F_0+\eta}+\frac{1}{F_0-F_1}} \end{aligned} \tag{60}$$

where the shear and bulk H-S bounds are recovered by choosing the values described in the following table.

	μ_{LB}	μ_{UB}	κ_{LB}	κ_{UB}
f_0	μ_0	μ_0	κ_0	κ_0
f_1	μ_1	μ_1	κ_1	κ_1
η	$\eta\mu(\kappa_m, \mu_m, 3)$	$\eta\mu(\kappa_m, \mu_m, 3)$	$\eta\kappa(\kappa_m, \mu_m, 3)$	$\eta\kappa(\kappa_m, \mu_m, 3)$

Table 5: HS relation table for 3 dimensions cases

Equation (60) can be refactored as

$$\begin{aligned} f_{UB} = \; & F_1 + \frac{(1-\rho)\,(F_0 - F_1)\,(F_1 + \eta_1)}{\eta_1 + F_0\,\rho + F_1\,(1-\rho)} \\ f_{LB} = \; & F_0 - \frac{\rho\,(F_0 - F_1)\,(F_0 + \eta_0)}{\eta_0 + F_0\,\rho + F_0\,(1-\rho)} \end{aligned} \tag{61}$$

It is possible to add and subtract the necessary terms, $F_0\,(1-\rho) + F_1\,\rho - [F_0\,(1-\rho) + F_1\,\rho]$, in order to obtain the following Hashin-Shtrikman generalized equation.

$$\begin{aligned} f_{UB} = \; & F_0\,(1-\rho) + F_1\,\rho + \left[-F_0\,(1-\rho) - F_1\,\rho + F_1 + \frac{(1-\rho)\,(F_0 - F_1)\,(F_1 + \eta_1)}{\eta_1 + F_0\,\rho + F_1\,(1-\rho)}\right] \\ f_{LB} = \; & F_0\,(1-\rho) + F_1\,\rho + \left[-F_1\,\rho + F_0\,\rho - \frac{\rho\,(F_0 - F_1)\,(F_0 + \eta_0)}{\eta_0 + F_0\,\rho + F_0\,(1-\rho)}\right] \end{aligned} \tag{62}$$

Expanding the terms inside the math brackets.

$$\begin{aligned} f_{UB} = \; & F_0\,(1-\rho) + F_1\,\rho - \frac{\rho\,(1-\rho)\,(F_0 - F_1)^2}{\eta_1 + F_0\,\rho + F_1\,(1-\rho)} \\ f_{LB} = \; & F_0\,(1-\rho) + F_1\,\rho - \frac{\rho\,(1-\rho)\,(F_0 - F_1)^2}{\eta_0 + F_0\,\rho + F_1\,(1-\rho)} \end{aligned} \tag{63}$$

Consequently, a more general expression for the 3D Hashin Shtrikman Bounds is

$$f(\rho) = F_0\,(1-\rho) + F_1\,\rho - \frac{\rho\,(1-\rho)\,(F_0 - F_1)^2}{\eta + F_0\,\rho + F_1\,(1-\rho)} \tag{64}$$

11.5 Generalization of the (HS) bounds and Simp-All η parameters for 'N' dimensions

In order to obtain a generalized expression for the Simp-All method, the first step is to recover the Hashin-Shtrikman bounds from [34], which are valid for 'N' dimensions. The deduction of these bounds can also be seen in the Appendix Section 4.

$$\begin{aligned} \frac{\theta}{2\,\mu_1 - 2\,\mu^*_{UB}} &\leqslant \frac{1}{2\,\mu_1 - 2\,\mu_0} + \frac{(1-\theta)\,(\kappa_1 + 2\,\mu_1)\,(N-1)}{\mu_1\,(N^2+N-2)\left(\kappa_1 + 2\,\mu_1 - \frac{2\,\mu_1}{N}\right)} \\ \frac{1-\theta}{2\,\mu^*_{LB} - 2\,\mu_0} &\leqslant \frac{1}{2\,\mu_1 - 2\,\mu_0} + \frac{\theta\,(\kappa_0 + 2\,\mu_0)\,(N-1)}{\mu_0\,(N^2+N-2)\left(\kappa_0 + 2\,\mu_0 - \frac{2\,\mu_0}{N}\right)} \\ \frac{\theta}{\kappa_1 - \kappa^*_{UB}} &\leqslant \frac{1}{\kappa_1 - \kappa_0} + \frac{1-\theta}{\kappa_1 + 2\,\mu_1 - \frac{2\,\mu_1}{N}} \\ \frac{1-\theta}{\kappa^*_{LB} - \kappa_0} &\leqslant \frac{1}{\kappa_1 - \kappa_0} + \frac{\theta}{\kappa_0 + 2\,\mu_0 - \frac{2\,\mu_0}{N}} \end{aligned} \tag{65}$$

being 'N' the problem dimension and θ the porosity.

Since the topology optimization problem has been expressed as a relaxed function of the density (ρ), the previous equations have to be replaced by $\rho = 1 - \theta$. On the other hand, the limit cases can be obtained by the insulation of the insertion parameters.

$$\begin{aligned}
\mu^*_{UB} &= \mu_1 + \frac{1-\rho}{\frac{1}{\mu_0-\mu_1} - \frac{2\,N\,\rho\,(\kappa_1+2\,\mu_1)\,(N-1)}{\mu_1\,(N^2+N-2)\,(\kappa_1\,N-2\,\mu_1+2\,\mu_1\,N)}} \\
\mu^*_{LB} &= \mu_0 - \frac{\rho}{\frac{1}{\mu_0-\mu_1} - \frac{2\,N\,(\kappa_0+2\,\mu_0)\,(N-1)\,(1-\rho)}{\mu_0\,(N^2+N-2)\,(\kappa_0\,N-2\,\mu_0+2\,\mu_0\,N)}} \\
\kappa^*_{UB} &= \kappa_1 + \frac{1-\rho}{\frac{1}{\kappa_0-\kappa_1} + \frac{\rho}{\kappa_1+2\,\mu_1-\frac{2\,\mu_1}{N}}} \\
\kappa^*_{LB} &= \kappa_0 - \frac{\rho}{\frac{1}{\kappa_0-\kappa_1} - \frac{1-\rho}{\kappa_0+2\,\mu_0-\frac{2\,\mu_0}{N}}}
\end{aligned} \tag{66}$$

Taking into consideration the results obtained in the previous subsections, the following definition can be made.

Definition. The generalized η parameters are defined so that the previous HS bounds equations can be regrouped regardless of the lamé parameter type.

$$\begin{aligned}
\eta\kappa(\kappa,\mu,N) &= \frac{2\,\mu\,(N-1)}{N} \\
\eta\mu(\kappa,\mu,N) &= -\frac{\mu\,\left(4\,\mu-\kappa\,N^2-2\,\mu\,N^2+2\,\mu\,N\right)}{2\,N\,(\kappa+2\,\mu)} = \frac{\mu}{2\mu+\kappa}\left(\frac{(N-2)(N+1)}{N}\mu+\frac{N}{2}\kappa\right)
\end{aligned} \tag{67}$$

Since the previous η are a generalized definition of the 2 and 3 dimension cases, see subsections (11.2) and (11.4), it can be proven that the following generalized Hashin Shtrikman bound equation is satisfied.

$$f_{HS}(\rho) = F_0\,(1-\rho) + F_1\,\rho - \frac{\rho\,(1-\rho)\,(F_0-F_1)^2}{\eta + F_0\,\rho + F_1\,(1-\rho)} \tag{68}$$

where the shear and bulk H-S bounds are recovered by choosing the values described in the following table.

	μ_{LB}	μ_{UB}	κ_{LB}	κ_{UB}
f_0	μ_0	μ_0	κ_0	κ_0
f_1	μ_1	μ_1	κ_1	κ_1
η	$\eta\mu(\kappa_m,\mu_m,N)$	$\eta\mu(\kappa_m,\mu_m,N)$	$\eta\kappa(\kappa_m,\mu_m,N)$	$\eta\kappa(\kappa_m,\mu_m,N)$

Table 6: HS relation table for N dimensions cases

It is noteworthy to mention that equations (67) are by themselves part of the generalized equations of the Simp-All method. Not only do they allow to simplify the η parameters for the 2D and 3D cases, but also because the Simp-All method only differs from one case to the other by η as it is shown in the following section.

In addition, the derivative of the generalized H-S bounds can be expressed as

$$f'_{HS}(\rho) = -\frac{(\eta+f_0)\,(\eta+f_1)\,(f_0-f_1)}{(\eta+f_0\,\rho+f_1\,(1-\rho))^2} \tag{69}$$

12 SIMP-All for 'N' dimensions

12.1 Topological derivative in terms of the shear and bulk modulus

Regarding the subsections (11.1) and (11.3), two cases can be identified.

(i) **Stiffer inclusion insertion:** Let in a point $\hat{x} \in \Omega_o$ be the matrix parameters $\mu = \mu_o$, $\kappa = \kappa_o$ and the inclusion $\mu^* = \mu_1$ and $\kappa^* = \kappa_1$.

$$\mathcal{T}_o(\hat{x}) = -\nabla^s u(\hat{x}) : d\mathbb{C}_o : \nabla^s u(\hat{x}) \tag{70}$$

with

$$d\mathbb{C}_o = 2\mu_o(\mu_1 - \mu_o)q_{\mu_o}\,\mathbb{I} + \left[\kappa_o(\kappa_1 - \kappa_o)q_{\kappa_o} - \frac{2}{N}\,\mu_o(\mu_1 - \mu_o)q_{\mu_o}\right] I \otimes I \tag{71}$$

where $q_{\mu_o} = q(\mu_o, \mu_1, \eta_{\mu_o})$, $q_{\kappa_o} = q(\kappa_o, \kappa_1, \eta_{\kappa_o})$, $\eta_{\mu_o} = \frac{\mu_o}{2\mu_o+\kappa_o}\left(\frac{(N-2)(N+1)}{N}\mu_o + \frac{N}{2}\kappa_o\right)$ and $\eta_{\kappa_o} = \frac{2\,\mu_o\,(N-1)}{N}$

(ii) **Weaker inclusion insertion:** Let in a point $\hat{x} \in \Omega_1$ be the matrix parameters $\mu = \mu_1$, $\kappa = \kappa_1$ and the inclusion $\mu^* = \mu_o$ and $\kappa^* = \kappa_o$.

$$\mathcal{T}_1(\hat{x}) = -\nabla^s u(\hat{x}) : d\mathbb{C}_1 : \nabla^s u(\hat{x}) \tag{72}$$

with

$$d\mathbb{C}_1 = 2\mu_1(\mu_o - \mu_1)q_{\mu_1}\,\mathbb{I} + \left[\kappa_1(\kappa_o - \kappa_1)q_{\kappa_1} - \frac{2}{N}\,\mu_1(\mu_o - \mu_1)q_{\mu_1}\right] I \otimes I \tag{73}$$

where $q_{\mu_1} = q(\mu_1, \mu_o, \eta_{\mu_1})$, $q_{\kappa_1} = q(\kappa_1, \kappa_o, \eta_{\kappa_1})$, $\eta_{\mu_1} = \frac{\mu_1}{2\mu_1+\kappa_1}\left(\frac{(N-2)(N+1)}{N}\mu_1 + \frac{N}{2}\kappa_1\right)$ and $\eta_{\kappa_1} = \frac{2\,\mu_1\,(N-1)}{N}$

12.2 Topological derivative as H-S bounds derivative

Taking into consideration Sections 7 and 8 , an interpolation scheme such that the gradient in Ω_0 and Ω_1 coincides precisely with the topological derivative can be imposed.

$$\begin{aligned} g(0) &= \mathcal{T}_o(\hat{x}) \\ -g(1) &= \mathcal{T}_1(\hat{x}) \end{aligned} \tag{74}$$

Regarding equations (11), (70) and (72), the following relations are obtained.

$$\begin{aligned} \mathbb{C}'(0) &= \quad d\mathbb{C}_o \\ \mathbb{C}'(1) &= -\,d\mathbb{C}_1 \end{aligned} \tag{75}$$

Basically, any infinitesimal insertion in the structure, caused by stiffer or weaker inclusion insertion, will be described by the compliance. Therefore, the addition or subtraction of an infinitesimal quantity of mass, is precisely a laminated composite generated by the two pure materials.

On the other hand, considering Sections 8 and 9, let the compliance be defined such that close to Ω_0 coincides with the H-S lower bound and close to Ω_1 with the H-S upper bound. It is equivalent to impose the gradient of the compliance is

$$\begin{aligned} \mathbb{C}'(0) &= f'_{HS}(0)|_{LB} \\ \mathbb{C}'(1) &= f'_{HS}(1)|_{UB} \end{aligned} \tag{76}$$

Since the compliance has the same derivative for the topological derivative and for the HS bounds, the following relation is obtained.

$$\begin{aligned} \mathbb{C}'(0) = &\quad d\mathbb{C}_o = f'_{HS}(0)|_{LB} \\ \mathbb{C}'(1) = &- d\mathbb{C}_1 = f'_{HS}(1)|_{UB} \end{aligned} \tag{77}$$

Basically, a change on the compliance at the neighbourhood of Ω_0 or Ω_1 implies the topological derivative, which proposes the insertion of an actual micro-structure (H-S micro-structure). Therefore, an infinitesimal inclusion inserted by the topological derivative is an actual homogenized micro-structure with circular or spherical inclusions whose fraction volume value is equivalent to the infinitesimal inclusion volume.

12.3 Comprehensive SIMP-All

Let the compliance be defined as equation (17) under the constraints (18).

$$R = \left\{ f \in C^\infty([0,1],[f_o,f_1]) \;\mid\; f(\rho) = \frac{A\rho^2 + B\rho + C}{D\rho + 1} \right\}$$

$$f(0) = f_o, \quad f'(0) = \dot{f}_o, \quad f(1) = f_1, \quad f'(1) = \dot{f}_1$$

The rational function and restrictions are provided for reference purposes.

Furthermore, the derivative of the rational function is.

$$f'(\rho) = \frac{AD\rho^2 + 2A\rho + (B - CD)}{(D\rho + 1)^2} \tag{78}$$

Therefore, the system composed of four equations can be solved to find the value of the parameters A, B, C, and D.

$$\begin{aligned} A = &\ \frac{-\dot{f}_1\dot{f}_o + (f_1 - f_o)^2}{\dot{f}_1 - (f_1 - f_0)} \\ B = &\ \frac{2f_o(f_1 - f_0) + \dot{f}_1\dot{f}_o - f_1\dot{f}_o - \dot{f}_1 f_o}{\dot{f}_1 - (f_1 - f_0)} \\ C = &\ f_o \\ D = &\ \frac{2(f_1 - f_0) - (\dot{f}_1 + \dot{f}_o)}{\dot{f}_1 - (f_1 - f_0)} \end{aligned} \tag{79}$$

On the other hand, considering (18), the relations shown in (77) and the derivative of the HS bounds (69), the constraints $\dot{f}_o$ and $\dot{f}_1$ become

$$\begin{aligned} \dot{f}_o = &\ f'_{HS}(0)|_{LB} \quad \Longrightarrow \quad \dot{f}_o = \frac{f_o + \eta}{f_1 + \eta}(f_1 - f_0) \\ \dot{f}_1 = &\ f'_{HS}(1)|_{UB} \quad \Longrightarrow \quad \dot{f}_1 = \frac{f_1 + \eta}{f_o + \eta}(f_1 - f_0) \end{aligned} \tag{80}$$

Moreover, it is shown in [3] that the compliance is always between the Hashin-Strikman bounds.

Finally, the comprehensive SIMP-All is a general expression composed by equations (17), (67), (79) and (80), where the parameters are defined by Table 6. Consequently, one function for the shear and another for the bulk are obtained.

13 Swan code Refactoring and modification

The results obtained during the development of this thesis have been implemented on SWAN [1]. It has allowed the comparison between the SIMP and SIMPALL interpolation curves and the increase of the topology optimization capabilities when computing actual micro-structures in 3D. On the other hand, the modified scripts have been refactored by means of the techniques described on Sections 2, 3, 4 and 5.

On the other hand, the unmodified code can be summarized in the following UML, whose scripts can be seen in [1]. It is noteworthy to mention that this code does not have the capacity to compute 3D structures by means of the SIMPALL method. Nevertheless, the improved code proposed in this thesis fulfills this deficiency.

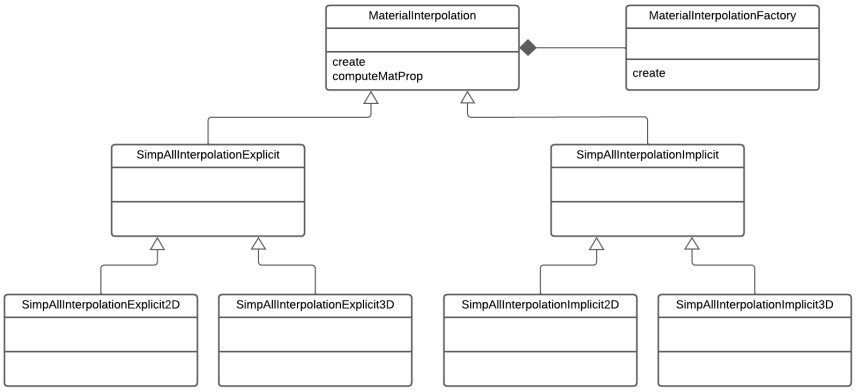

Figure 28: UML diagram of the SWAN code scripts before modification

The parts of the SWAN code that have been modified can be seen in Appendix Section 5. Furthermore, the new code has the following UML diagram.

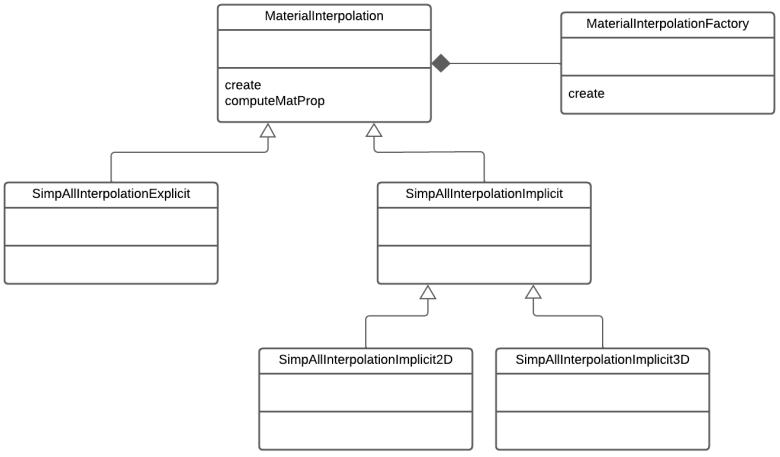

Figure 29: UML diagram of the SWAN code scripts after modification

14 Comparison between SIMP and SIMP-ALL

The section is divided into two parts, the first one aims to compare different cases for the SIMP and SIMP-All methods in two and three dimension scenarios. On the other hand, the second part proves the results obtained in the first part with numerical examples. These examples are obtained from the modified Swan code, see Section 13, using the same interpolation schemes shown in the first part.

14.1 Interpolation scheme comparison

In the regularized topology optimization problem, the material of the domain (base material) is usually interpolated with a void material, modeled with an extremely weak material. Nevertheless, the base material can be mixed with another material (non-void material). In this section, it is intended to compare the interpolation schemes in both situations for two and three dimensional cases.

Each of the figures below show the SIMP and SIMP-All interpolation schemes and how these vary with the parameters E_o, ν_o, E_1 and ν_1. In order to better contrast the SIMP and SIMP-All methods, the Hashin Shtrikman bounds are also plotted.

Besides that, these figures are obtained by means of the SIMPALL(E1,E0,nu1,nu0, N) function, which can be found in: `https://github.com/Jof-syntax/TFG_Jofre`.

Material void interpolation

For a two dimension case problem, the following interpolation functions are shown below.

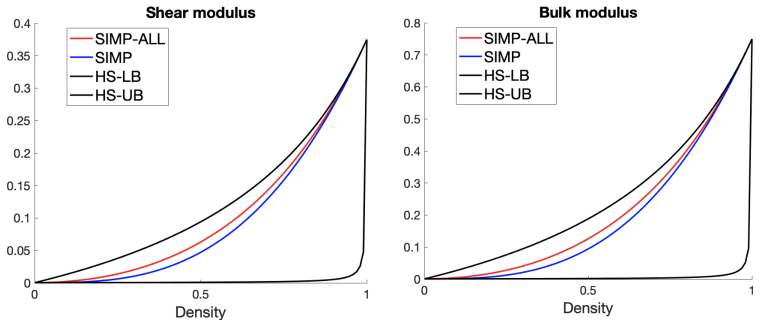

Figure 30: 2D case, where $E_o = 10^{-3}$, $\nu_o = 1/3$, $E_1 = 1$ and $\nu_1 = 1/3$

In this case, both interpolation schemes have similar shape and behaviour. On the one hand, the SIMP and SIMP-All functions are within the Hashin Shtrikman bounds, whose results will always be physically possible. On the other hand, the shape of the auto-penalizing scheme is similar for both and good enough to obtain converged results.

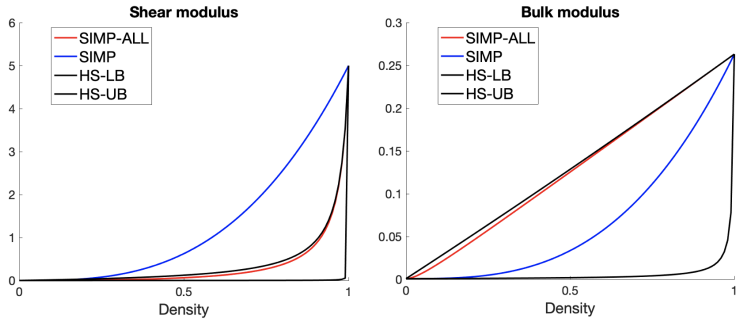

Figure 31: 2D case, where $E_o = 10^{-3}$, $\nu_o = 1/3$, $E_1 = 1$ and $\nu_1 = -0.9$

The second example, same as Figure 27, shows inconveniences for both methods. First, the SIMP shear function is outside the HS bounds. Therefore, its grey colors are not physically possible. Moreover, this function, in the shear case, is not as penalising as the SIMP-All interpolation scheme. However, it proposes actual solutions for the Bulk scheme. Meanwhile, the SIMP-All function has great auto-penalizing scheme for the shear but not for the bulk (as its derivative is almost constant). Finally, the order of magnitude in this example is favorable for the Shear scheme.

On the other hand, for a three dimension scenario, the following interpolation functions are discussed.

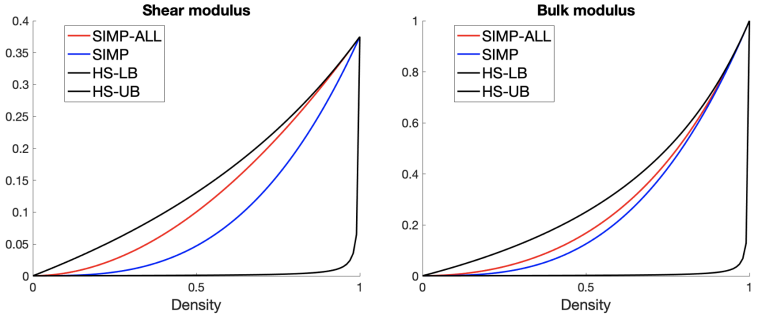

Figure 32: 3D case, where $E_o = 10^{-3}$, $\nu_o = 1/3$, $E_1 = 1$ and $\nu_1 = 1/3$

Similarly to Figure 30, Figure 32 has the SIMP and SIMP-All schemes within the Hashin Shtrikman bounds. The bulk and shear modulus have a similar order of magnitude and the curvature for both methods is almost the same. Therefore, not only will the results obtained from these interpolation curves be similar, but also the results will be physically possible.

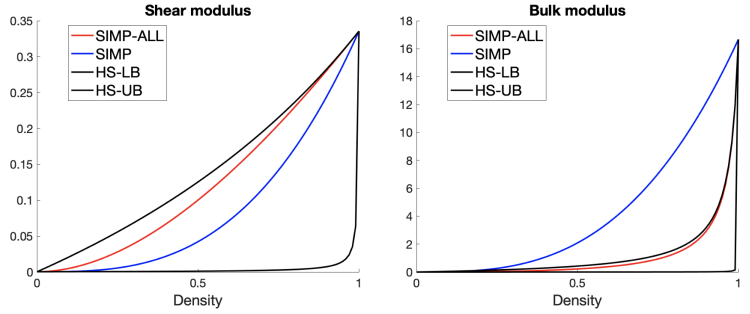

Figure 33: 3D case, where $E_o = 10^{-3}$, $\nu_o = 1/3$, $E_1 = 1$ and $\nu_1 = 0.49$

It can bee seen in Figure 33 that the SIMP function is outside the Hashin-Shtrikman bounds (Bulk scheme). Consequently, in case of grey parts, the results will not be physically possible. It is remarkable that the maximum bulk modulus is considerably larger than the maximum shear modulus. On the other hand, the shear scheme for both methods will have a similar behaviour.

Bi-material interpolation

In bi-material problems, the parameter 'p' for the SIMP method is unclear. However, the following figures have been computed with a value of $p = 3$. First, a two dimensional case is presented. Then, the same case is shown for three dimensions. Finally, a case with negative Poisson ratio is presented.

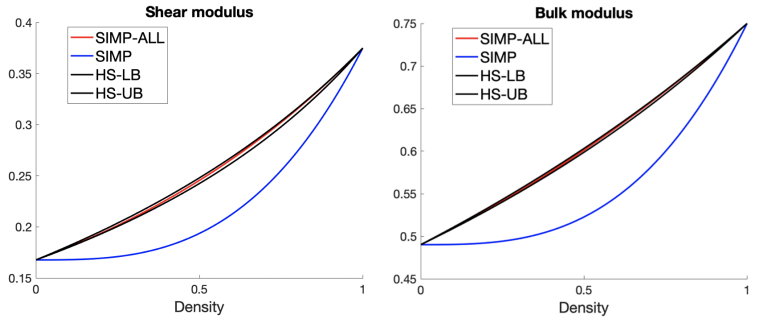

Figure 34: 2D case, where $E_o = 0.5$, $\nu_o = 0.49$, $E_1 = 1$ and $\nu_1 = 1/3$

It is important to note that the second material has a Poisson ratio tending to its positive maximum (0.5) and the Young Module is half effective compared to the first material. In line with the previous figures, the SIMP method has its values outside the physical bounds. Furthermore, the derivative of the SIMP-All method tend to be constant, which may cause grey colors in the numerical results. Nevertheless, the combination of materials that creates these grey colors will be physically possible.

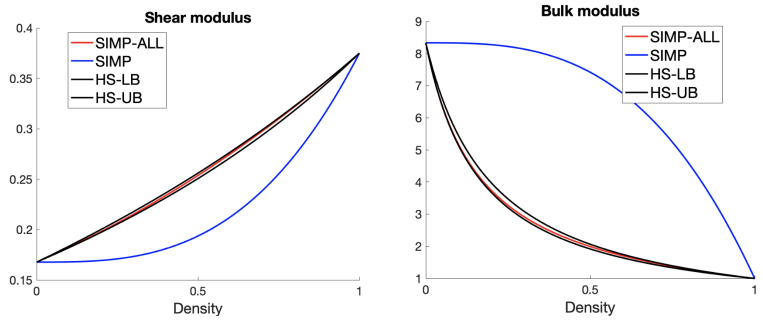

Figure 35: 3D case, where $E_o = 0.5$, $\nu_o = 0.49$, $E_1 = 1$ and $\nu_1 = 1/3$

Figure 35 is computed with the same parameters of Figure 34, but for a three dimension case. While the shear function remains the same, the bulk curve has different magnitude and derivative. Furthermore, not only do the SIMP and SIMP-All methods (bulk plot) have opposite shapes, but the SIMP function also is outside the HS bounds. It results in the impossibility for this method to obtain structures with physical meaning in case of grey colors.

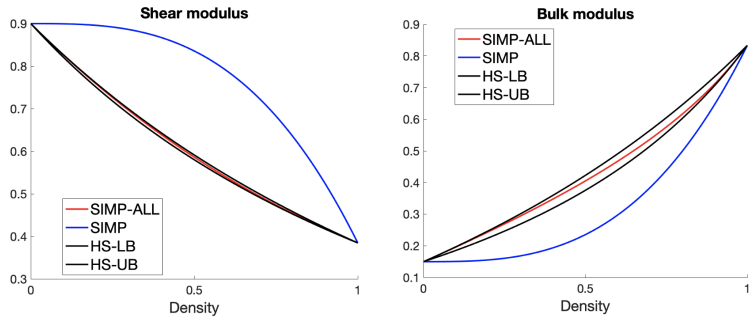

Figure 36: 3D case, where $E_o = 0.9$, $\nu_o = -0.5$, $E_1 = 1$ and $\nu_1 = 0.3$

The last plot shows a case of negative Poisson ratio, which results in having the SIMP Bulk and SIMP Shear functions outside the HS bounds. Meanwhile, the SIMP-All function is between the HS bounds as expected but its derivative is almost constant.

It has been shown that the SIMP-All method is always between the Hashin-Shtrikman bounds. Nevertheless, since the SIMP-All function depends on the HS bounds, the curvature of the function may be constant in some cases. It means that the results may present intermediate values. However, the unconverged part (grey color) is always physically possible to be manufactured. On the other hand, the SIMP method has been shown to be outside the limits of HS for some cases. Considering only these interpolation schemes, the SIMP-All method seems to be a better choice for topology optimisation problems.

14.2 Numerical examples comparison

Benchmark cases

The different meshes used in the development of this part are presented below. First, general characteristics of the meshes are presented in Table 7. Then, a definition of these examples is carried out, where the dimensions, boundary conditions and external forces are shown.

Case	Number of nodes	Number of elements
Cantilever (2D)	10329	20257
Cantilever (3D)	14909	78955
Chair 3D	14926	79048

Table 7: Mesh information

It is noteworthy to mention that these meshes have been obtain by means of GID [36] . These meshes can be found in: `https://github.com/Jof-syntax/Meshes-TFG`.

Cantilever beam

A Cantilever beam is a rigid structural element anchored at only one end, whereas the other side has an external force applied. When the structural load is applied, the beam carries the load to the attached side. This type of problem causes moment and shear stresses. Besides that, the scheme used to compute the Cantilever results for this section can be seen in the following images.

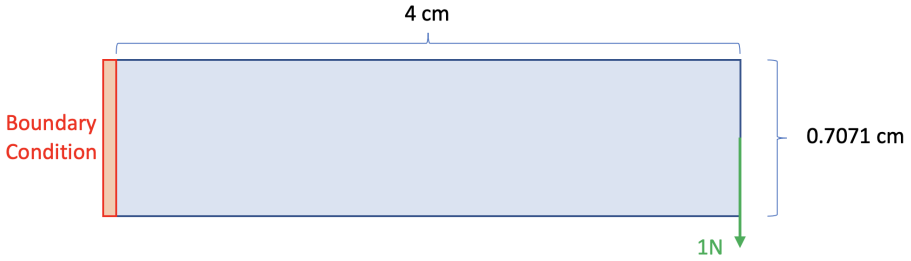

Figure 37: Cantilever beam scheme for 2D, where the boundary conditions (fixed nodes), the external force and the dimensions appear

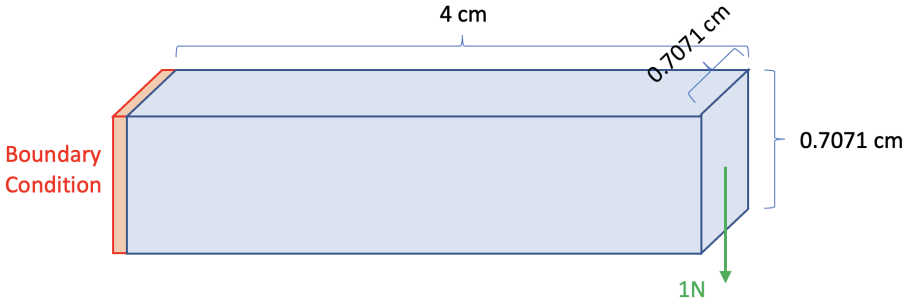

Figure 38: Cantilever beam scheme for 3D, where the boundary conditions (fixed nodes), the external force and the dimensions are shown

It is remarkable that this type of problem has analytical solution [37], therefore, it can be used as a test for FEM solvers.

Chair

Secondly, the chair problem is presented. Since the forces are perpendicular to the structure, this problem is an excellent example to study the bulk modulus. Therefore, it will be used to compute the $E_o = 10^{-3}$, $\nu_o = 1/3$, $E_1 = 1$ and $\nu_1 = 0.49$ cases in three dimensions. Furthermore, the scheme used to compute these examples has been carried out with the following scheme.

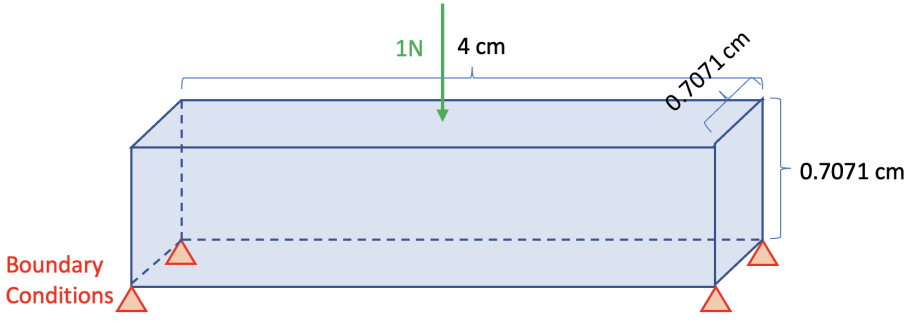

Figure 39: Chair scheme for 3D, where the boundary conditions (fixed nodes), the external force and the dimensions appear

The solution of this example tends to be a chair, where depending on the external forces and volumetric constraints it will have two, three or four legs.

Results

It has been considered the compliance as the cost function and the volume as a constraint for the relaxed topology optimization problem. Besides that, the MMA algorithm has been used with a stopping criteria tolerance of 10^{-3} and a maximum number of iterations of $nmax = 1000$. Furthermore, the objective volume for two dimension cases has been $V = 0.15$, whereas the objective volume for 3D cases has been $V = 0.02$. Nevertheless, for the chair problem in three dimensions, the objective volume has been modified to $V = 0.2$. On the other hand, the initial material domain has been chosen to be the stiffest, which coincide with $\rho = 1$.

Material void interpolation

It is noteworthy to mention that the white part of the following results is a void material.

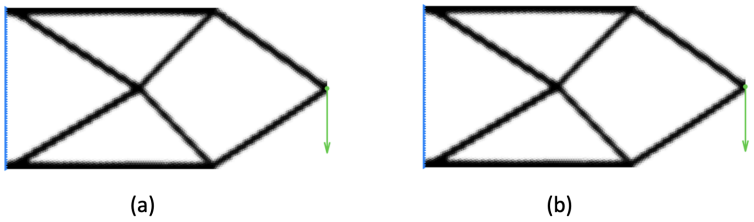

Figure 40: Two dimensional results from Swan, where $E_o = 10^{-3}$, $\nu_o = 1/3$, $E_1 = 1$ and $\nu_1 = 1/3$. Result obtained with the SIMP method (a) and with the SIMP-All method (b). [kktnorm = $8.09 \cdot 10^{-3}$ and $n = 349$ (a)] and [kktnorm = $4.27 \cdot 10^{-3}$ and $n = 412$ (b)].

In this case, the solution has converged into a similar shape. It can be attributed to the similar curvature of the interpolation functions, see Figure 30. Furthermore, since both functions are within the HS bounds, the solution is physically possible.

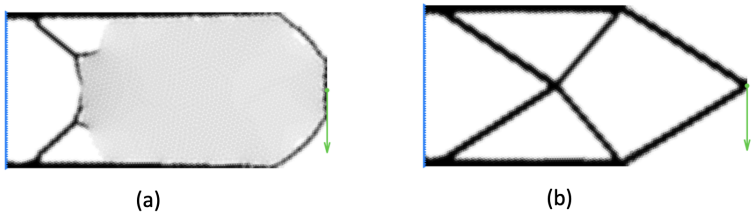

Figure 41: Two dimensional results from Swan, where $E_o = 10^{-3}$, $\nu_o = 1/3$, $E_1 = 1$ and $\nu_1 = -0.9$. Result obtained with the SIMP method (a) and with the SIMP-All method (b). [kktnorm = $1.21 \cdot 10^{-2}$ and $n = 1000$ (a)] and [kktnorm = $6.87 \cdot 10^{-4}$ and $n = 342$ (b)].

However, for the case of Figure 31, the results are not similar. In this case, the SIMP-All result has converged into the base materials, whereas the SIMP has not. Even though, each interpolation function has drawbacks, as it has been explained in the specific interpolation plot of subsection (14.1), SIMP function has been more inconvenient. In this case, the SIMP function from the shear modulus has caused the grey parts of Figure 41, since the curvature is not as penalizing as the SIMP-All's curvature and the shear modulus is greater than the bulk modulus. On the other hand, while the SIMP-All result is physically possible, the result proposed by the SIMP method is not physically possible because the function is outside the Hashin-Shtrikman bounds.

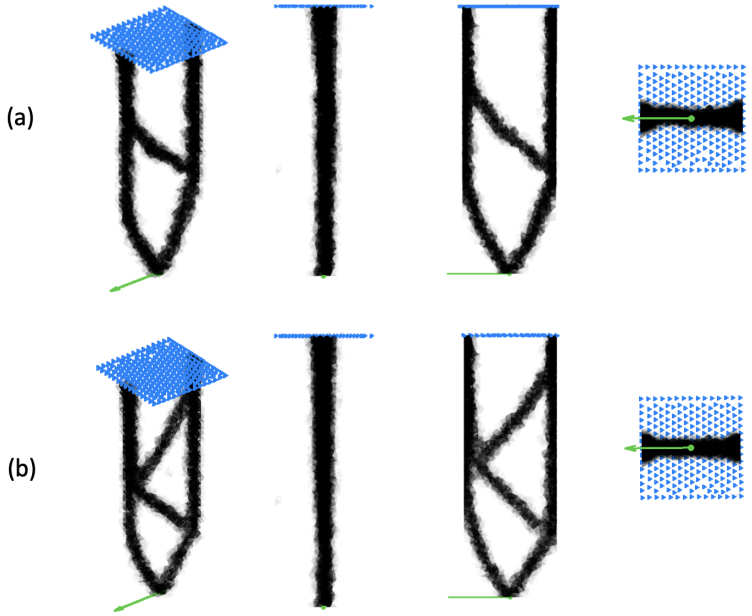

Figure 42: Three dimensional results from Swan, where $E_o = 10^{-3}$, $\nu_o = 1/3$, $E_1 = 1$ and $\nu_1 = 1/3$. Result obtained with the SIMP method (a) and with the SIMP-All method (b). [kktnorm $= 5.83 \cdot 10^{-3}$ and $n = 320$ (a)] and [kktnorm $= 6.87 \cdot 10^{-4}$ and $n = 351$ (b)].

Although the program has proposed two different shapes in Figure 42, the achieved characteristics are the same. It is because the problem's constraints are satisfied. Besides that, the interior of the black shapes are composed of grey zones. However, since both interpolation schemes are within the HS bounds (see Figure 32), these parts are actual solutions.

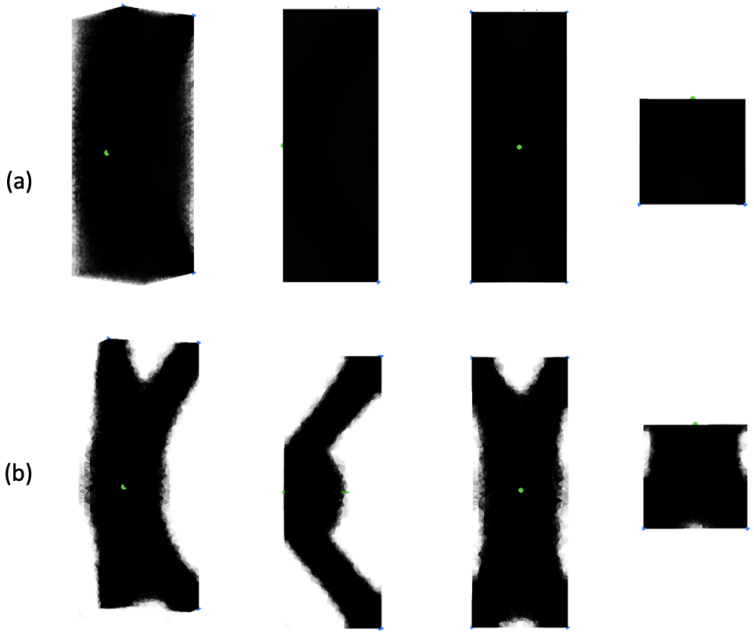

Figure 43: Three dimensional results from Swan, where $E_o = 10^{-3}$, $\nu_o = 1/3$, $E_1 = 1$ and $\nu_1 = 0.49$. Result obtained with the SIMP method (a) and with the SIMP-All method (b). [kktnorm $= 3.13 \cdot 10^{-2}$ and $n = 1000$ (a)] and [kktnorm $= 1.57 \cdot 10^{-3}$ and $n = 782$ (b)].

Similarly to Figure 41, the SIMP derivative from Figure 33 has caused the unconvergence of the solution in Figure 43. Nevertheless, the SIMP-All solution has been satisfactory, because the solution has converged into a shape and because all the parts are physically possible.

Bi-material interpolation

It is remarkable that the white part of the following results is a material and not void.

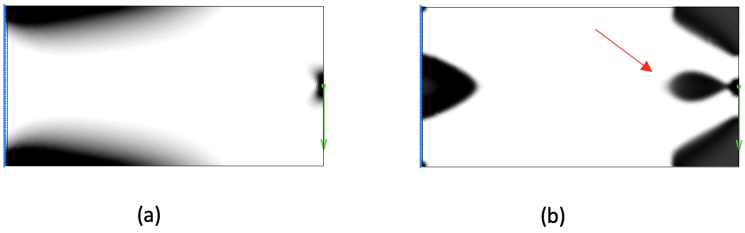

Figure 44: Two dimensional results from Swan, where $E_o = 0.5$, $\nu_o = 0.49$, $E_1 = 1$ and $\nu_1 = 1/3$. Result obtained with the SIMP method (b) and with the SIMP-All method (a). [kktnorm = $7.11 \cdot 10^{-3}$ and $n = 496$ (a)] and [kktnorm = $3.78 \cdot 10^{-3}$ and $n = 532$ (b)]. Besides that, the red arrow shows a grey region.

Different interpolation shapes has caused different solutions, see Figure 35. However, both results seem to be physically possible. On the one hand, the SIMP-All function is within the HS bounds, therefore, grey is physically possible. On the other hand, the grey zone from the SIMP method is not physically possible.

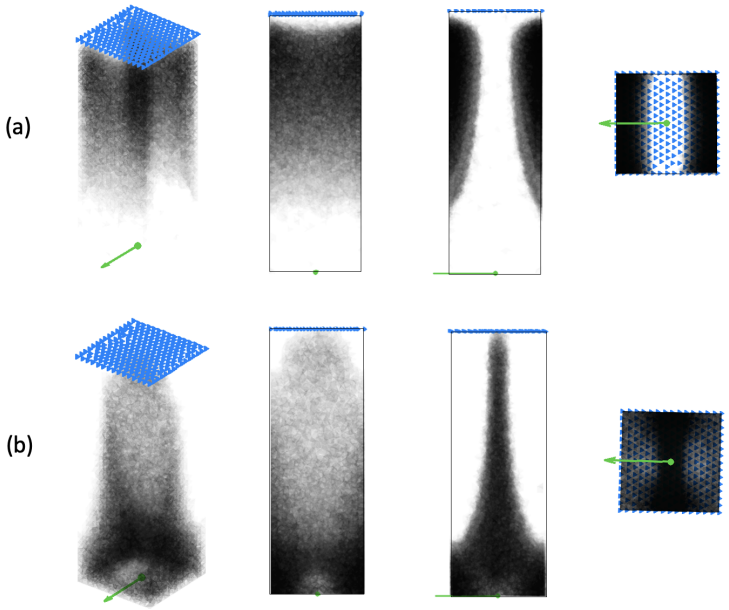

Figure 45: Three dimensional results from Swan, where $E_o = 0.5$, $\nu_o = 0.49$, $E_1 = 1$ and $\nu_1 = 1/3$. Result obtained with the SIMP method (b) and with the SIMP-All method (a). [kktnorm = $5.67 \cdot 10^{-3}$ and $n = 810$ (a)] and [kktnorm = $1.44 \cdot 10^{-3}$ and $n = 745$ (b)].

Looking at Figure 45, both interpolation schemes have found different shapes results. Nevertheless, only the SIMP-All solution is valid. The reason relies on the Hashin-Shtrikman bounds, where the SIMP-All interpolation scheme is between the bounds, whereas the SIMP is not. It implies that the grey regions is physically possible for the SIMP-All method, whereas the SIMP is not.

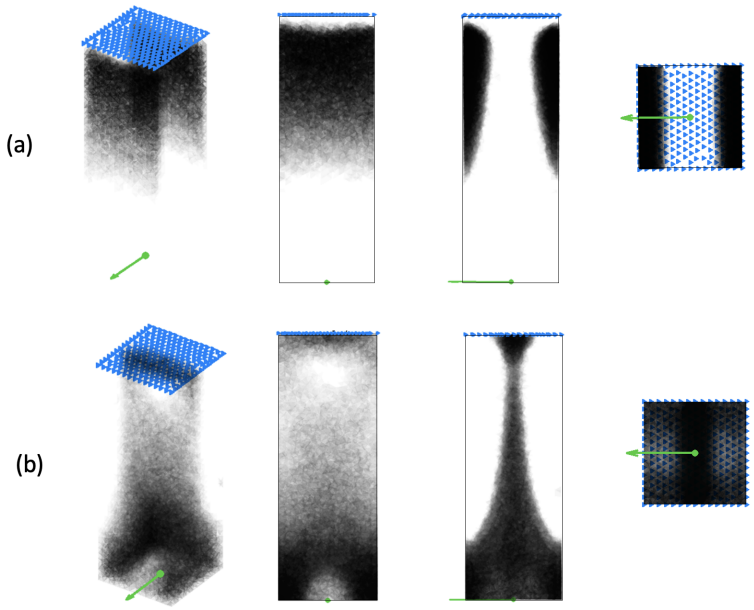

Figure 46: Three dimensional results from Swan, where $E_o = 0.9$, $\nu_o = -0.5$, $E_1 = 1$ and $\nu_1 = 0.3$. Result obtained with the SIMP method (b) and with the SIMP-All method (a). [kktnorm $= 5.33 \cdot 10^{-3}$ and $n = 912$ (a)] and [kktnorm $= 3.09 \cdot 10^{-3}$ and n=841 (b)].

The last plot shows similar results to those obtain in Figure 45. Nevertheless, in this case the bulk and shear have similar magnitude. Since the result has grey regions and the SIMP function is outside the HS bounds, the result is not an actual solution. Regarding the SIMP-All interpolation scheme is a physical solution because the function is between physical bounds.

Finally, SIMP-All has been shown to be a better option. It performs similarly to the SIMP method in terms of curvature, but has an advantage in the grey regions, as demonstrated.

Chapter IV. Economic and environmental study

15 Economic and environmental study

Economics

The development of the thesis has a total cost summarized in Table 8. Besides that, the complete budget breakdown can be seen in the BUDGET document.

Identification	**Cost (€)**
Human cost	9432
Software cost	370
Hardware cost	310
Electrical cost	25.07
Total Cost	**10137.07**

Table 8: Total budget

It is noteworthy to mention that the cost has been computed with an hourly pay rate of 18 €/h, an electrical cost of 0.33 €/kWh and a hardware amortization of 20 % annual.

Environmental impact

The environmental impact of this study is basically related to the study itself and it is almost null. The main impact is related with the CO_2 emissions caused by the electricity consumption of the computer used during the development of this work. From the budget, it is known that the study lasted 524 hours. During this period, the computer used has a mean consumption of 145W. Therefore, this study has consumed 75.98 kWh. Taking into consideration that 1 kWh is approximately 0.29 kg CO_2 [38], the project has produced 22.03 kg of CO_2.

Besides that, the results of the thesis will impact positively in the usage of row material. Since all the topology optimization results, generated as a consequence of the results of this thesis, are actual solutions to be manufactured and the structural volume usage is greatly reduced, the materials used to build the structure are the minimum ones. For instance, less materials will be used in the creation of a high performance automotive pieces. Nevertheless, the exact impact can not be computed because the results are not implemented in the industries yet.

Chapter V. Conclusions and future actions

16 Conclusions and future actions

Conclusions

The aim of the study is considered to be achieved since the SIMP-All method has been expanded to 'N' dimension scenarios and the results have been successfully implemented in the SWAN code. It is necessary to emphasize that these results could not be achieved without a correct comprehension of modern programming tools and topology optimization.

On the one hand, the refactoring of the "Codi Cante" has provided a practical approach of the modern programming tools sections of the thesis. During the rewriting of the code, concepts of object oriented programming, cloud-backed work, test driven development and clean code have been applied and further discussed in the report. Furthermore, the knowledge acquired during this part has helped on the correct implementation of the SIMP-All results in the Swan code. Thus allowing an easy correction, understanding and efficiency of the code. It is remarkable that these concepts are not limited to this project, they can be applied to any programming project and can be utilized with almost all the popular programming languages (e.g. JavaScript, C++, etc).

On the other hand, it has been possible to obtain a comprehensive equation for the SIMP-ALL method. Not only does the generalized equation allow to find optimal results, but the generated structures are also possible to be manufactured in the reality. During the deduction, knowledge of topology optimization, Hashin-Shtrikman bounds and problem formulation has been discussed. Furthermore, a comparison between the SIMP-All and SIMP methods has been carried out, showing that the SIMP-All method is better choice. Besides that, the equation has been implemented in the Swan program using the new programming tools. It will help other scientists to verify the results of this thesis by themselves.

Finally, while the results allow to go one step further in the science of topological optimisation, the programming knowledge discussed allows to be a better programmer for today's era.

Future actions

Since all the objectives of this thesis have been successfully accomplished, the future tasks could be devoted to improve the existing ones. Consequently, the actions to consider would be:

First, research for advanced programming tools that could simplify the computational process of the Swan code. If the research is successful, the improvements can be implemented in Swan.

Besides that, during the development of the thesis an article regarding the results has been started. Nevertheless, it is not completely finished. Therefore, another task could be focused on completing the current article. It includes review, improve and publish it. The structure, grammar and vocabulary has to be carefully analyzed and modified where necessary. The improvement task can be done by analyzing other case scenarios, improving the mesh quality or the number of iterations. Lastly, the article can be sent for correction and published in a scientific magazine.

Finally, refactor the Swan code in a larger scale. It could simplify the overall functionality of the program and improve its efficiency in the results' computation.

These tasks are intended to be done during the summer of 2022.

Chapter VI. Bibliography

References

[1] Ferrer, A. (s. f.). SwanLab. [Online]. Accessed on 22/02/2022. Available at: https://github.com/SwanLab/Swan.

[2] G. Allaire. Shape Optimization by the Homogenization Method. Springer, (1st edition) edition, 2002.

[3] Ferrer, A. SIMP-ALL: A generalized SIMP method based on the topological derivative concept. John Wiley & Sons, Ltd, 2019.

[4] git-scm. What is Git?, 2021. [Online]. Accessed on 12/02/2022. Available at: https://Git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F.

[5] Abhinav96. What is a GIT Repository?, 2019. [Online]. Accessed on 12/02/2022. Available at: https://www.geeksforgeeks.org/what-is-a-Git-repository/.

[6] Laurence Bradford. What Is GitHub?, 2020. [Online]. Accessed on 14/02/2022. Available at: https://www.thebalancecareers.com/what-is-Github-and-why-should-i-use-it-2071946.

[7] Michel Lewis. What exactly is Github anyway, 2019. [Online]. Accessed on 14/02/2022. Available at: https://techcrunch.com/2012/07/12/what-exactly-is-Github-anyway/.

[8] Github, Inc. Github, 2022. [Online]. Accessed on 15/03/2022. Available at: https://github.com.

[9] Github, Inc. GitHub Desktop, 2020. [Online]. Accessed on 12/02/2022. Available at: https://desktop.github.com.

[10] bluuweb. GITHUB, 2020. [Online]. Accessed on 16/02/2022. Available at: https://bluuweb.github.io/tutorial-github/02-github/#pull.

[11] git-scm. Basic Branching and Merging, 2018. [Online]. Accessed on 18/02/2022. Available at: https://Git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging.

[12] bluuweb. Basic Branching and Merging, 2020. [Online]. Accessed on 18/02/2022. Available at: https://bluuweb.Github.io/tutorial-Github/01-fundamentos/#ramas-o-branch.

[13] Erin Doherty. What is object-oriented programming? OOP explained in depth, 2020. [Online]. Accessed on 24/04/2022. Available at: https://www.educative.io/blog/object-oriented-programming#what-is.

[14] Sarah Lewis and Alexander S. Gillis. Object-oriented programming (OOP), 2021. [Online]. Accessed on 24/04/2022. Available at: https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP.

[15] Bill Wagner, GitHubPang and pkulikov. Polymorphism, 2021. [Online]. Accessed on 24/04/2022. Available at: https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/object-oriented/polymorphism.

[16] The MathWorks, Inc. Comparison of Handle and Value Classes, 2022. [Online]. Accessed on 24/04/2022. Available at: https://es.mathworks.com/help/matlab/matlab_oop/comparing-handle-and-value-classes.html.

[17] The MathWorks, Inc. Which Kind of Class to Use, 2022. [Online]. Accessed on 24/04/2022. Available at: https://es.mathworks.com/help/matlab/matlab_oop/which-kind-of-class-to-use.html.

[18] The MathWorks, Inc. Property Attributes, 2022. [Online]. Accessed on 24/04/2022. Available at: https://es.mathworks.com/help/matlab/matlab_oop/property-attributes.html.

[19] The MathWorks, Inc. Method Attributes, 2022. [Online]. Accessed on 24/04/2022. Available at: https://es.mathworks.com/help/matlab/matlab_oop/method-attributes.html.

[20] Emiliano Sutil. Beneficios del TDD en entornos DevOps, 2020. [Online]. Accessed on 08/04/2022. Available at: https://www.xeridia.com/blog/beneficios-del-tdd-en-entornos-devops#quees.

[21] Deister Software. [Online]. Accessed on 09/04/2022. Available at: https://www.deister.net/en/.

[22] Bruce Eckel. Thinking in Java. Prentince Hall, (3rd edition) edition, 2002.

[23] MathWorks. matlab.unittest.plugins.CodeCoveragePlugin class, 2022. [Online]. Accessed on 10/04/2022. Available at: https://es.mathworks.com/help/matlab/ref/matlab.unittest.plugins.codecoverageplugin-class.html.

[24] Stephen Watts, Chrissy Kidd. What is Code Refactoring? How Refactoring Resolves Technical Debt, 2018. [Online]. Accessed on 11/04/2022. Available at: https://www.bmc.com/blogs/code-refactoring-explained/.

[25] Lucidchart. UML Class Diagram Tutorial, 2022. [Online]. Accessed on 11/04/2022. Available at: https://www.lucidchart.com/pages/uml-class-diagram.

[26] Fadi Stephan. Top 4 Symptoms of Bad Code, 2016. [Online]. Accessed on 17/04/2022. Available at: https://www.excella.com/insights/top-4-symptoms-of-bad-code.

[27] Sergey Zolotaryov. Key Indicators of Problem Design, 2017. [Online]. Accessed on 17/04/2022. Available at: https://codingsight.com/indicators-of-problem-design/.

[28] Triona Moynihan. SOLID Design Principles, 2019. [Online]. Accessed on 17/04/2022. Available at: https://tinyurl.com/n4msvh4h.

[29] Robert Martin. Clean Code: A Handbook of Agile Software Craftsmanship. Pearson, (1st edition) edition, 2008.

[30] SubSurfWiki. 1st Lamé parameter, 2012. [Online]. Accessed on 04/06/2022. Available at: https://subsurfwiki.org/wiki/1st_Lame_parameter.

[31] Novotny AA, Sokołowski J. Topological Derivatives in Shape Optimization. Interaction of Mechanics and Mathematics Series. Springer-Verlag, 2013.

[32] Sett-Sensi. Material Interpolation, 2017. [Online]. Accessed on 15/05/2022. Available at: https://abaqus-docs.mit.edu/2017/English/TsoUserMap/tso-c-user-TopOpt-Sett-Sensi-MatInt.htm.

[33] Shtrikman S Hashin Z. A variational approach to the theory of the elastic behaviour of multiphase materials. J Mech Phys Solids., (1st edition) edition, 1963, Vol. 11, pp. 127to 140.

[34] Grégoire Allaire. Shape Optimization by the Homogenization Method. Springer, (1st edition) edition, 2002.

[35] S.M. GIUSTI. TOPOLOGICAL DERIVATIVE FOR 3D ELASTICITY PROBLEMS. National Technological University, 2017.

[36] CIMNE. GiD Simulation, 2022. [Online]. Accessed on 06/06/2022. Available at: https://www.gidsimulation.com.

[37] Luis Ortiz Berrocal. Resistencia de materiales. McGraw-Hill Interamericana de España S.L., (3rd edition) edition, 2007.

[38] Ministerios de Industria, Energía y Turismo y Ministerio de Fomento. Factores de emisiÓn de co2 y coeficientes de paso a energÍa primaria de diferentes fuentes de energÍa final consumidas en el sector de edificios en espaÑa. Tech. rep. , 2016.