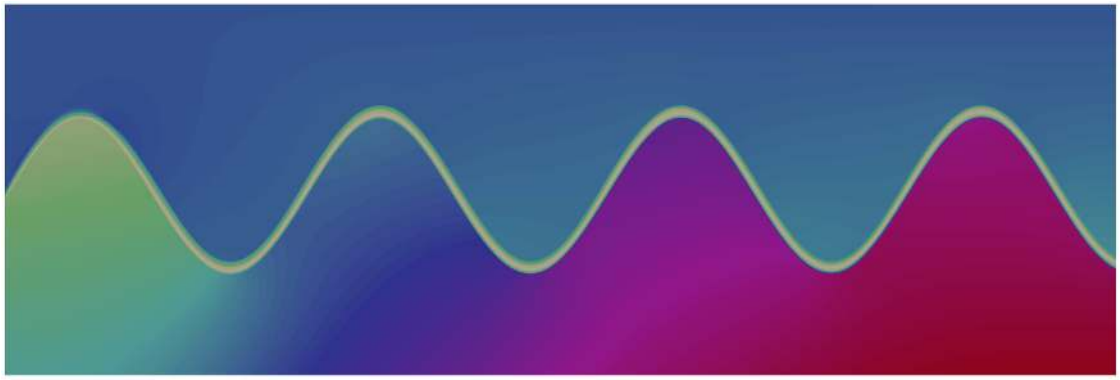
TÉCNICO LISBOA



Vasculature Optimization of Actively-Cooled Materials

Gonçalo Ferreira Nunes Gouveia Valente

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisor(s): Prof. Alejandro M. Aragón (*TU Delft*)
Prof. André Calado Marta (*IST*)

Examination Committee

Chairperson: Prof. Filipe Szolnoky Ramos Pinto Cunha
Supervisor: Prof. André Calado Marta
Member of the Committee: Prof. João Orlando Marques Gameiro Folgado

June 2016

Dedicated to my beloved family and friends

Acknowledgments

First and foremost, I would like to thank my supervisors Professor Alejandro M. Aragón and Professor André Calado Marta, the first for the guidance and support that allowed me to reach as far as I did in this project, the second for the constant feedback and interest in the project and this document as well as in my concerns, and both for their patience during the months it took me to complete this thesis.

Next, I would like to thank the *TU Delft* university and more specifically the *Mechanical, Maritime and Materials Engineering* (*3mE*) Department for providing me the research topic and the infrastructure needed for this project. The feedback provided in my progress presentations was also very helpful.

I would also like to thank my lab mates at *TU Delft* for being supportive and always up for a brainstorm session that has always been crucial in solving the issues that appeared during the course of this work.

Finally, I would like to thank all my friends and family for their love and support ever since I needed it. Their help has been important for my entire academic path but the remote support over the last few months of completing this thesis was crucial.

Without the help and support from this big group of people, this project and my academic path would not have been possible. Thank you everyone!

Resumo

Com a vontade de explorar todas as possibilidades do imaginário da humanidade surgiu o mundo da engenharia. Em aplicações de engenharia onde os processos de transferência de calor estão presentes surgem os materiais vasculares arrefecidos activamente, que fornecem motivação para esta tese.

O objetivo deste trabalho é implementar as ferramentas necessárias para modelar e optimizar problemas de transferência de calor conjugados (o calor dispersa-se pelo domínio não só através de condução mas também de convecção).

Com o desejo de uma implementação independente da configuração dos canais associado ao processo de optimização, é necessário encontrar uma solução de discretização inovadora, técnica que foi encontrada no Interface-enriched Finite Element Method.

As equações que definem a transferência de calor nas duas fases de materiais vasculares são diferentes. Nas regiões dominadas por convecção, as formulações de elementos finitos padrão são ineficazes e é necessário encontrar uma solução diferente, tendo-se escolhido o Streamline Upwind/Petrov-Galerkin.

Estando integrados num grupo de trabalho na *TU Delft*, todos os aspectos de implementação fazem parte de uma ferramenta computacional desenvolvida pelo referido grupo (*hybrida*).

Seguindo as premissas acima, foi possível comparar resultados com testes de verificação para os métodos discutidos e também optimizar geometrias vasculares, minimizando a temperatura maxima no seu domínio.

Usando a abordagem e resultados descritos, foi possível concluir que o trabalho desenvolvido resultou na aplicação correcta de cada método e também que a sua combinação é eficiente e eficaz na modelação e optimização de problemas de transferência de calor conjugados definidos por domínios vasculares arrefecidos activamente.

Palavras-chave: Interface-enriched Finite Element Method, Streamline Upwind/Petrov-Galerkin, Convecção de Calor, Optimização.

Abstract

Associated with an eagerness to explore every possibility from humankind imagination came the field of engineering. In engineering applications where heat transfer processes are present arise vascular actively-cooled materials, which provide motivation for this thesis.

The goal of this work is to implement the tools required to model and optimize conjugated heat transfer problems (heat disperses through the domain not only *via.* heat conduction but also convection).

In order to obtain an implementation that is independent of the configuration of the cooling channels (important for the optimization process), an innovative solution for the discretization method is in order, which can be found in the Interface-enriched Finite Element Method.

The equations that define heat transfer in the two phases of vascular materials are different. In the convection dominated regions, the standard finite element formulations are ineffective and it is necessary to find a different solution, being the Streamline Upwind/Petrov-Galerkin techique the chosen one.

Being developed in a work group at *TU Delft*, all implementation aspects are integrated in *hybrida*, a computational tool developed by said group.

Following this procedure made it possible to match results with verification tests for the discussed methods and also to optimize vascular geometries by minimizing their maximum domain temperature.

Using the approach and results described above, it has been possible conclude that the developed work resulted in the correct implementation of each method and also that their combination is effective and efficient modeling and optimizing conjugated heat transfer problems defined by vascular actively-cooled domains.

Keywords: Interface-enriched Finite Element Method, Streamline Upwind/Petrov-Galerkin, Heat Convection, Optimization.

Contents

Acknowledgments v
Resumo vii
Abstract ix
List of Tables xv
List of Figures xvii
Nomenclature xix
Glossary xxi

1 Introduction **1**
1.1 Perspective and Motivation 1
1.1.1 History 1
1.1.2 Bio-Mimetic Approach 1
1.1.3 Actively-Cooled Materials and their Applications 2
1.1.4 Interest in the Topic 4
1.2 Objectives and Work Overview 4
1.3 Thesis Outline 6

2 Theoretical Background **7**
2.1 Background in Heat Transfer 7
2.1.1 Heat Conduction in the Solid Phase 7
2.1.2 Heat Convection in the Fluid Phase 9
2.1.3 Boundary Conditions 11
2.2 Mathematical Fundamentals in Solid Mechanics 12

3 Discretization Method - IGFEM **15**
3.1 Finite Element Method Fundamentals 15
3.2 Requirements 16
3.3 Interface-enriched Generalized FEM 17
3.3.1 Formulation 18
3.3.2 $1D$ Sample Problem 19
3.4 Implementation 23
3.4.1 *Geometric Engine* 23

3.4.2 Elemental Assembly . . . 30
3.5 Verification . . . 33
3.5.1 Error Norms . . . 33
3.5.2 $2D$ Plate: Patch Test . . . 33
3.5.3 Eshelby Inclusion Problem . . . 36
3.6 Other Topics . . . 39
3.6.1 Enrichment Functions Scaling . . . 39

4 Weighted Residual Formulation - SUPG **41**
4.1 Requirements . . . 41
4.2 Streamline Upwind/Petrov-Galerkin method . . . 42
4.2.1 Problem Statement for the Conjugate Heat Transfer Problem . . . 43
4.2.2 Formulation . . . 44
4.3 Implementation . . . 45
4.4 Verification . . . 46
4.4.1 $1D$ Convection Approximation . . . 46
4.4.2 Convection Skew to the Mesh . . . 48

5 Test Cases Modeling Results **53**
5.1 CPU Cooler Test Case . . . 53
5.1.1 Problem Description . . . 53
5.1.2 Implementation Efficiency . . . 56
5.1.3 Solution . . . 57
5.2 Engine Cylinder Test Case . . . 58
5.2.1 Problem Description . . . 58
5.2.2 Solution . . . 60

6 Optimization Fundamentals **63**
6.1 Problem Formulation . . . 63
6.2 Optimization Tool . . . 64
6.3 Optimization Methods . . . 64
6.4 Selected Method . . . 66

7 Test Case Optimization Results **67**
7.1 Bounds and Constraints . . . 67
7.2 Enhanced Solution . . . 68

8 Conclusions **75**
8.1 Achievements . . . 75
8.2 Future Work . . . 77

Bibliography **79**

A Inclusion Problem Convergence **83**

A.1 Analytic Derivatives of the Displacement Field 83

A.1.1 Inside the Inclusion 83

A.1.2 Outside the Inclusion (matrix) 83

List of Tables

5.1 Computation times *vs.* number of finite elements. 56

7.1 Test case optimization results. 69

List of Figures

1.1 Honeycombs from Wahl et al. [4]. . . . 2
1.2 Vascular materials (from Lucas et al. [7], Nilsson [8] and Soghrati and Geubelle [9]). . . . 3
1.3 Active-cooling in a scramjet from Saravanamuttoo et al. [13]. . . . 3
1.4 *hybrida*'s original structure. . . . 5
1.5 Objectives flowchart. . . . 6

2.1 Illustration of heat conduction. . . . 8
2.2 Concept and classifications of heat convection. . . . 10
2.3 Relevant boundary layer types (from Bergman et al. [6]). . . . 10
2.4 $1D$ linear elastic solid under traction load. . . . 13

3.1 Mesh generation issues. . . . 16
3.2 $1D$ IGFEM sample problem geometry. . . . 19
3.3 Solution of $1D$ bar IGFEM example. . . . 23
3.4 Modifications to code's structure due to geometric engine. . . . 24
3.5 Geometric engine result in a square domain. . . . 28
3.6 Geometric engine result in a circular domain. . . . 29
3.7 Preliminary modeling of vascular material. . . . 30
3.8 Modifications to code's structure due to assembly. . . . 30
3.9 $2D$ bi-material plate problem statement for the convergence study. . . . 34
3.10 $2D$ bi-material plate problem. . . . 36
3.11 $2D$ circular inclusion problem statement. . . . 37
3.12 $2D$ circular inclusion. . . . 38
3.13 Enrichment function scaling. . . . 39
3.14 Condition number. . . . 40

4.1 Temperature distribution solution in $1D$ bar to point out spurious oscillations. . . . 42
4.2 Sample 2D domain for conjugate heat transfer problem. . . . 43
4.3 $1D$ fluid bar. . . . 46
4.4 Analytic and Galerkin temperature distribution solution in $1D$ bar for multiple Péclet numbers. . . . 47
4.5 Exact and central difference solution for convection in $1D$ bar. . . . 48

4.6 Analytic and SUPG temperature distribution solution in a bar. 49
4.7 Convection skew to the mesh problem statement. 49
4.8 Analytic temperature distribution for skew convection from Brooks and Hughes [44]. . . . 50
4.9 Galerkin temperature distribution for skew convection from *hybrida*. 50
4.10 Galerkin temperature distribution for skew convection from Brooks and Hughes [44]. . . . 50
4.11 SUPG temperature distribution for skew convection from *hybrida*. 51
4.12 SUPG temperature distribution for skew convection from Brooks and Hughes [44]. 51

5.1 CPU cooler test case baseline geometry and sample mesh. 54
5.2 Hagen–Poiseuille velocity profile from White [18]. 55
5.3 Sample mesh for $A = 2\,mm$, $n_{sw} = 2$ 55
5.4 Mesh convergence study. 56
5.5 CPU cooler test case baseline geometry solution. 57
5.6 Engine cylinder test case geometry. 59
5.7 Engine cylinder test case solution. 60
5.8 Temperature distribution along exit of cooling channel for engine cylinder test case. 61

7.1 Evolution of relevant optimization data with iteration number. 71
7.2 CPU cooler optimization problem's result for $A = 2\,mm$ and $n_{sw} = 2$. 71
7.3 CPU cooler optimization problem's result for $A = 4\,mm$ and $n_{sw} = 4$. 72
7.4 CPU cooler optimization problem's result final design in $2D$. 72
7.5 CPU cooler optimization problem's result final design $3D$ views. 73
7.6 Temperature distribution along bottom boundary of the cooler. 74

Nomenclature

Greek symbols

α	Thermal diffusivity of the material.
σ	Stress vector.
ϵ	Strain.
λ	Second Lamé constant.
μ	First Lamé constant.
ν	Poisson's ratio.
ψ	Enrichment function.
ρ	Volumetric mass density.
τ	SUPG stabilization factor.

Roman symbols

a	Cooling channel amplitude.
A	Cross-sectional area.
$\mathbf{b}$	Body force vector.
c_p	Specific heat at constant pressure.
E	Young's modulus of elasticity.
f	Heat source.
$\mathbf{k}$	Thermal conductivity of the material vector.
L	Domain length.
$\dot{m}$	Mass flow rate.
N	Standard Lagrangian shape function.
$n_s w$	Number of sinusoidal waves in the horizontal direction of the domain.

Δp Cooling channel pressure drop due to viscous effects.

Pe Péclet number.

$\mathbf{q}$ Heat transfer rate.

$\mathbf{q}''_{\mathbf{x}}$ Heat flux.

$\dot{q}$ Rate of energy generation per unit volume.

r_i Inclusion radius.

r_u Outer radius.

s Scaling factor.

t Traction force.

T Temperature.

u Main variable in finite element problem.

v Coolant velocity in the flow direction at a specified point.

w Finite element method weight function.

Subscripts

θ Azimutal coordinate.

i Evaluated at the i^{th} standard node.

j Evaluated at the j^{th} interface node.

k In the direction of the flow.

r Radial coordinate.

x, y, z Cartesian components.

Superscripts

h Approximated using a finite element method.

T Transpose.

Glossary

1D	One-dimensional.
2D	Two-dimensional.
3D	Three-dimensional.
BC	Boundary Condition.
BFGS	Broyden-Fletcher-Goldfarb-Shanno is an optimization method from the *SciPy* library.
BL	Boundary Layer.
CCW	Counterclockwise.
CFD	Computational Fluid Dynamics is a branch of fluid mechanics that uses numerical methods to solve problems that involve fluid flows.
CG	Conjugated Gradient is an optimization method from the *SciPy* library.
CPU	Central Processing Unit.
FEM	Finite Element Method is a numerical technique used for finding approximate solutions to boundary value problems described by partial differential equations.
GFEM	Generalized Finite Element Method is a modification of the standard Finite Element Method that combines desirable features from it and meshless methods.
IGFEM	Interface-enriched Generalized Finite Element Method is a modification of the Generalized Finite Element Method.
L-BFGS	Limited-memory Broyden-Fletcher-Goldfarb-Shanno is an optimization method from the *SciPy* library.
PDE	Partial Differential Equation.

SLSQP Sequential Least SQuares Programming is an optimization method from the *SciPy* library.

SUPG Streamline Upwind/Petrov Galerkin is a stabilization technique used for alleviating the spurious oscillations presented by the Galerkin method modeling convection dominated flows.

TNC Truncated Newton is an optimization method from the *SciPy* library.

XFEM eXtended Finite Element Method is a modification of the Finite Element Method.

Chapter 1

Introduction

This introductory chapter is divided into three sections and intends to report on the motivation to study vascular actively-cooled materials, to break that interest into explicit objectives and milestones and to provide a short summary of the chapters that follow.

1.1 Perspective and Motivation

The objective of this section is to situate the topic of vascular actively-cooled materials that is addressed in this dissertation in the global field of engineering, to introduce some basic concepts associated with said topic and to conclude about the interest in studying it.

1.1.1 History

As the years are going by, the engineering inventions are getting more advanced and complex, with an outburst of new technologies and respective applications. With the appearance of advanced applications, a need for new materials that have certain characteristics has been created. Due to this demand for new materials and improvement of existing technologies, materials research is an extremely dynamic field and the field of Materials Science and Engineering, which deals with the discovery and design of new materials, appeared and developed.

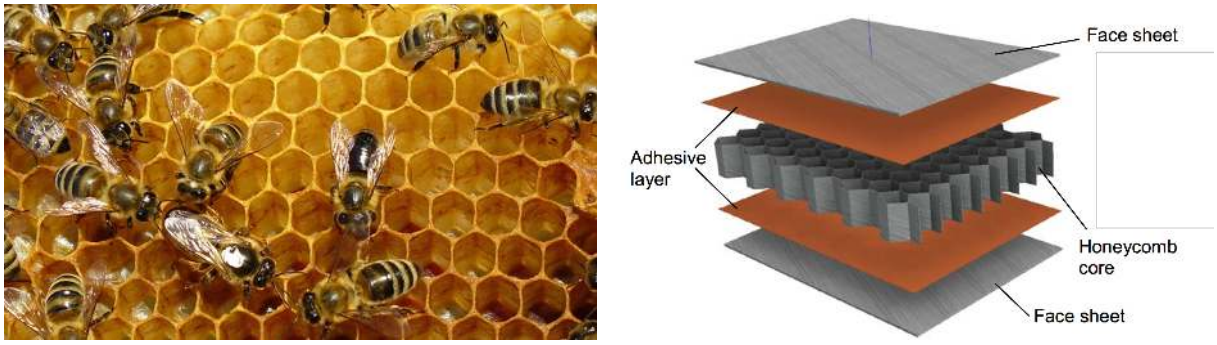
1.1.2 Bio-Mimetic Approach

One of the ways that is used within the field of Materials Science and Engineering to design or improve materials is to employ nature's solutions to deal with a certain problem. The materials that arise from this process can be called bio-mimetic and the chosen solutions being discussed here have already been time tested by nature through evolution and improved if successful (Bar-Cohen [1]).

Applications for these bio-mimetic materials range from biology and medicine to the more relevant topic of designing structural materials (with many other in the middle).

Before focusing on the topic at hand, it is necessary to take a step back and access the feasibility of bio-mimetic materials. According to Stone [2] and Fratzl [3], one may conclude that the successful implementations of bio-mimetic materials (or rather the employment of nature's solutions in engineering problems) may not be as straight-forward as imitating nature. The specific limitations that have to be dealt with in the topic of this thesis are described a few paragraphs below.

Moving from bio-mimetic materials in general to (bio-mimetic) structural materials intended for engineering applications, one can also find a wide range of applications. To provide an example of something that is probably already familiar to anyone with an engineering background, one can refer to man-made honeycomb structures, represented in Figure 1.1(b). Man-made honeycomb structures are bio-mimetic materials (inspired by natural structures such as beehives, Figure 1.1(a), and bone) with the goal of minimizing the amount of used material to reach minimal weight and minimal material cost. Despite having been used by men for more than 3000 years, these structures are used in complex applications in fields such as aerospace, automotive and railway (more information on honeycomb structures can be found in Wahl et al. [4]).



(a) Beehive honeycomb structure

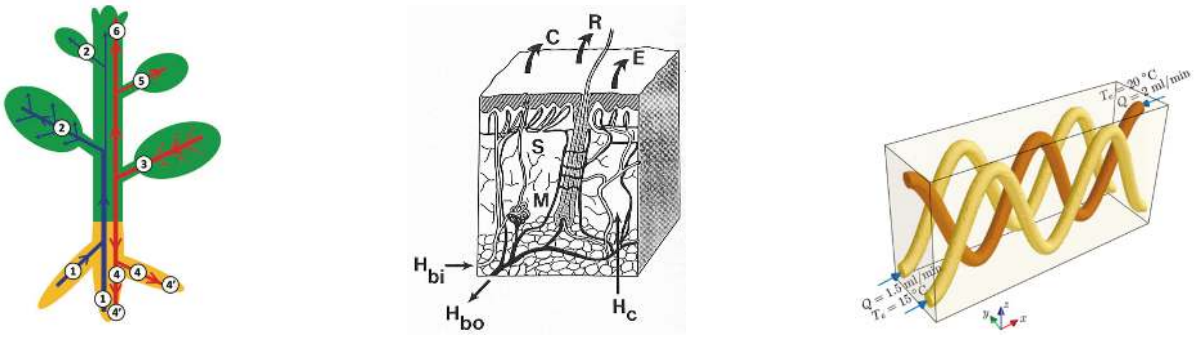
(b) Man-made honeycomb sandwich panel

Figure 1.1: Honeycombs from Wahl et al. [4].

1.1.3 Actively-Cooled Materials and their Applications

As described in the tittle of this dissertation, the main topic to be discussed here is vascular active-cooled materials. Although the name may be self-explanatory, it is tough to be important to provide the definitions of (man-made) vascular materials and active-cooling. Man-made vascular materials are designed using the vascular system found in plants and animals (both shown in Figure 1.2(a) and (b)) for inspiration and can be used to introduce characteristics such as active-cooling (mimicking a circulatory system and using a coolant as the circulating fluid) and even self-healing (described in Toohey et al. [5]). On the other hand, the term active-cooling simply clarifies that the method of cooling makes use of an external force to drive the coolant in the channels, unlike natural forces such as gravity (Bergman et al. [6]). Finally, Figure 1.2(c) represents a computational implementation of actively-cooled materials.

When comparing the described vascular approach to cooling against the traditional ones, such as solid fins (extended surfaces), one key advantage arises. This advantage is the redistribution of heat in the medium, which reduces the maximum temperature (Soghrati et al. [10]). This heat redistribution is a

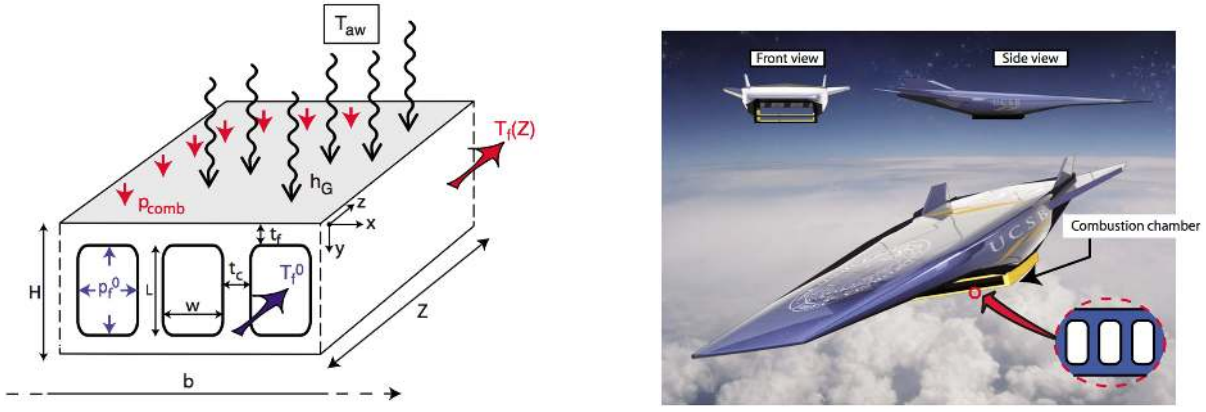


(a) Plants (b) Animals (c) Implementation

Figure 1.2: Vascular materials (from Lucas et al. [7], Nilsson [8] and Soghrati and Geubelle [9]).

result of the convection within the fluid phase and opens the door for high-temperature applications.

Although high-temperature applications are present in several engineering fields, it is thought to be interesting to name at least one example in aerospace engineering. This application is lightweight actively-cooled panels for use in combustors. Having been researched in several academic papers (like Valdevit et al. [11] or Vermaak et al. [12]), the use of an active-cooling strategy allows for the internal surfaces of the combustor in a scramjet engine to withstand the combination of combustion heat fluxes and aerodynamic heating without exceeding the limit temperatures of the composing materials. Figure 1.3 presents a schematic of a block of actively-cooled material and its application in a scramjet engine.



(a) Schematic of actively cooled panel (b) Render of hypersonic vehicle

Figure 1.3: Active-cooling in a scramjet from Saravanamuttoo et al. [13].

Resuming the above paragraph on the possible difficulties of applying a bio-mimetic solution to an engineering problem and focusing on the problem of active-cooling, one can intuitively think that the topology of the vascular system will affect the mechanical properties of the designed material and, more to the case, its thermal performance. Considering this, an interest in the usage of an optimization tool also arises as a way to check how much the thermal performance of a certain material employed in a determined test case can be improved.

1.1.4 Interest in the Topic

Taking the challenges described in the previous subsections into consideration, the interest to develop a dissertation on the subject appears as an answer to "how the cooling channels' configuration affects the thermal and mechanical performance in a determined test case?" is desired. The work on this subject has at its core the development of numerical and discretizaton tools that can, effectively and efficiently, help with the design and optimization of bio-mimetic vascular materials with the objective of active-cooling by solving the conjugate heat transfer problem (conduction and convection).

The interest in exploring this topic is thought to be clear with what was mentioned so far, as well as the interest in developing a computational tool that accomplishes the capabilities listed above.

The creation of the described numerical tool is not a simple task due to the large number of design variables involved (Aragón [14]) and to the particular numerical and discretization techniques that are needed to accurately model the geometries that need to be studied and represent all the involved phenomena. One key observation for this is that, in addition to having to deal with different material properties (solid and coolant), the convection in the coolant (fluid phase) must be considered in the numerical solution, which means that there is a change not only in the material properties but also in the governing equations (according to Soghrati et al. [10]).

Despite the possible difficulties, coupling the implementation of the required numerical and discretization models with an optimization process could lead to interesting results, which provides further motivation to take this project forward.

1.2 Objectives and Work Overview

The goal of this section is to provide a clear plan for the translation of the interest in the topic, which was thoroughly described in Section 1.1, into a list of explicit objectives and milestones.

Before describing the concise objectives of the work, it is worth mentioning that the topic of this thesis was introduced by *Technische Universiteit Delft* as a master thesis proposal and the tools to be explained are integrated in a project within the *Structural Optimization and Mechanics (SOM)* group in the *3mE* department called *hybrida* (see Aragón [15]), a new finite element method (FEM) package that is being developed by said group. This finite elements tool is being developed with the main goals of rapid development, performance and complete control and its structure (at least to the date of writing of this document) is presented in Figure 1.4. In the presented structure schematic, the thicker rectangle with the title *igfem* is a completely new module that is introduced during the course of this project and the boldface dashed line rectangle with the title *mesh* is a module where modifications are introduced also during the course of this project.

To get the results that are crucial to the development of a thorough research on the topic for this dissertation, there is an intent in developing an implementation of the numerical and discretization methods necessary to model a two-dimensional ($2D$) vascular block of material with active cooling within *hybrida* and later on to study ways of optimizing the vascular channels' distribution.

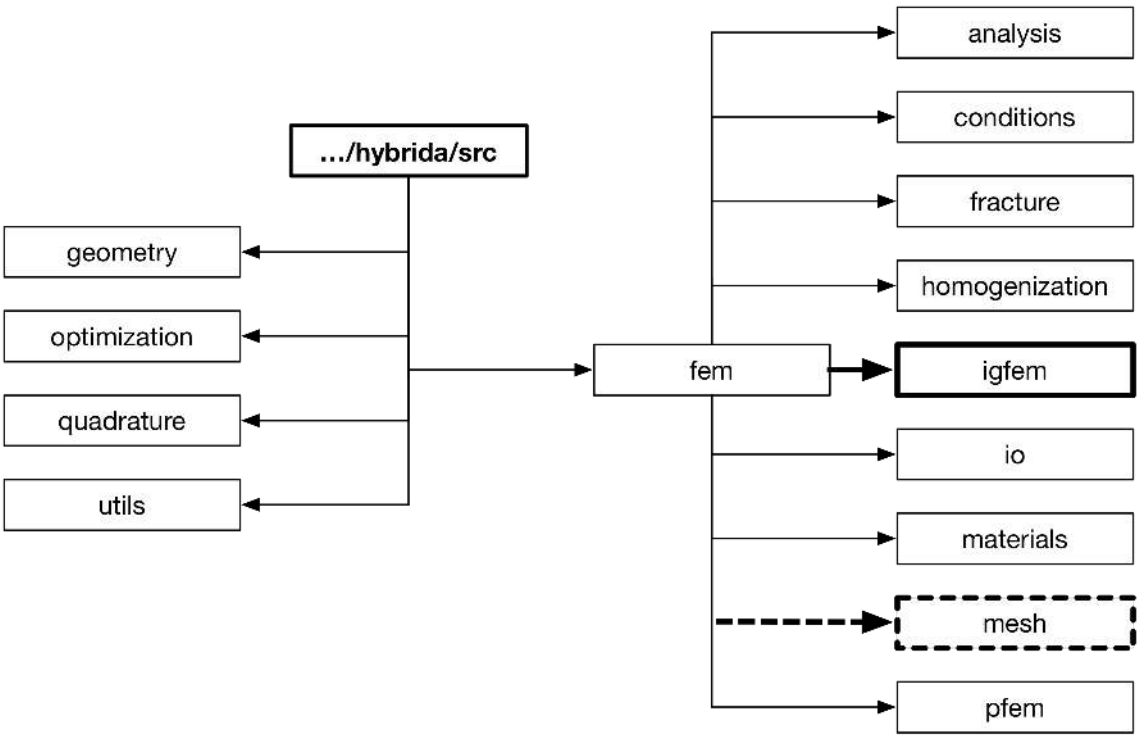


Figure 1.4: *hybrida*'s original structure.

Not only to make this document easier to analyze and follow but also as a way creating a thread to help the work to maintain the intended trajectory, it is convenient to start by dividing it into a few key milestones. These milestones are related to the implementation of the required numerical methods and programming in general (integrated in *hybrida*) whereas the analysis and conclusions on the obtained results are considered to be implicit in these milestones, which are:

1. Implementation of the Interface-enriched Finite Element Method (IGFEM, the discretization model);

2. Verification of IGFEM implementation;

3. Implementation of the Streamline Upwind/Petrov-Galerkin (SUPG, the weighted residual formulation);

4. Verification of SUPG implementation;

5. Using implemented tools to study the optimization of vascular channels' distribution, with the goal of obtaining the best geometry for a given set of boundary conditions.

As a way to systematize the objectives listed above and to visualize the main programming milestones, a flowchart containing the main milestones is presented in Figure 1.5.

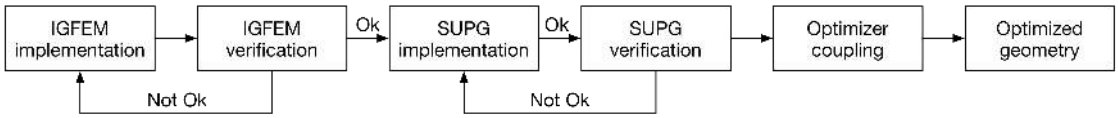


Figure 1.5: Objectives flowchart (programming milestones).

1.3 Thesis Outline

This final section of the introductory chapter presents the dissertation's structure and the contents of the chapters that follow. The remaining chapters try to answer all the questions that can arise from the topic in the best and most complete way.

This thesis is divided into seven chapters: the first (this one) is an introduction and motivation for the work (where the relevance and the interest in the subject is described).

In Chapter 2, a literature review of heat transfer and solid mechanics is presented in order allow the reader to understand the physical phenomena behind problems that are going to be analyzed.

Chapters 3 and 4 do an overview of the numerical methods that need to be employed, starting with the concept of those methods and their formulation and moving on to the topics of implementation, application examples and verification of the implementation in *hybrida*.

Chapter 5 showcases the power of the union of IGFEM and SUPG by modeling an actively-cooled vascular material block with BCs and geometry that mimic a CPU cooler using the developed implementation.

Chapter 6 is where everything related to the optimization process is presented, from the goals and a basic explanation of the geometry to the choices made regarding the optimization method and problem formulation.

Applying all the described methods and tools, Chapter 7 presents extensive results of the optimization of the CPU cooler geometry introduced in Chapter 5.

Finally, Chapter 8 is the conclusion of this project and a summary of all the findings in it, also including some suggestions for future work.

Chapter 2

Theoretical Background

The objective of this chapter is to describe the theoretical concepts relevant to the study of actively-cooled vascular materials. To do this, Section 2.1 introduces the fundamental concepts needed to understand the phenomenon of heat transfer in these materials. Being a basic concept review, a thorough mathematical deduction is not the main goal.

On the other hand, Section 2.2 provides a brief overview on the solid mechanics concepts that are relevant to do the verification of the discretization method to be discussed in Chapter 3. Because solid mechanics is not a subject directly linked to the subject being studied but more like a tool that helps implementation verification, this literature review focuses on mathematical concepts instead of physical ones.

Finally, it is important to understand that the concepts presented here describe the physical phenomena in a qualitative way, while Chapters 3 and 4 approach the solutions to implement them.

2.1 Background in Heat Transfer

2.1.1 Heat Conduction in the Solid Phase

Conduction is the dominant heat transfer process that occurs within the solid phase of a material block and it should be viewed as the energy transferal process that happens between particles at a greater temperature, which have more molecular energy, *i.e.* have a more intense random motion, and particles at a smaller temperature, which have less molecular energy, *i.e.* less intense random motion, due to interaction between them (Bergman et al. [6]). A simple illustration of the process is presented in Figure 2.1, where q'' is heat flux (concept explained below).

Before introducing and explaining any further concepts, it is important to state the major goal of the thermal analysis to be implemented in the computational tool that will model the vascular material under analysis which, like in any normal heat conduction analysis, is to determine the temperature field within a block of material resulting from the imposed boundary conditions (BCs).

The objective of determining the temperature distribution in a medium comes down to solving the appropriate form of the heat equation, which can derived from the *Fourier's Law* in Bergman et al. [6],

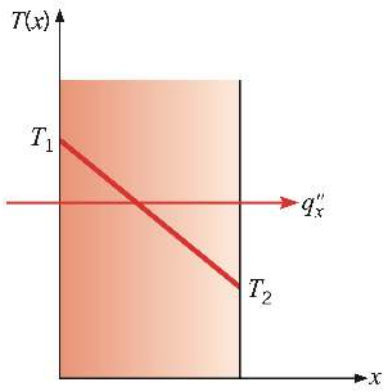


Figure 2.1: Illustration of heat conduction within a $1D$ solid from Bergman et al. [6].

applying the BCs that characterize the problem. Thus, in a one-dimensional ($1D$) problem, one must get the $T(x)$ solution, in a $2D$ problem $T(x, y)$ and, finally, $T(x, y, z)$ for a three-dimensional ($3D$) problem. This work focuses on computing temperature distributions on a $2D$ domain ergo the solution to obtain has the $T(x, y)$ form.

Although a complete mathematical demonstration of the method used to derive the heat equation is a little outside the scope of this work, it is thought to be important to provide an explanation of the fundamental concepts of heat conduction that are required to accomplish this.

The first concept to present when analyzing heat transfer processes is always the heat transfer rate, which for the case of conduction is called conduction rate. For a simpler understanding of the concepts, this description begins with the one dimensional case and is then generalized when appropriate. The rate of conduction in the x direction is given by Equation (2.1) (the equations for the y and z direction follow the same principle) and the minus sign in it indicates that heat is always transferred in the direction of decreasing temperature (following the convention presented in Bergman et al. [6] and represented in Figure 2.1) and is proportional to the temperature gradient in the corresponding direction,

$$q_x = -kA\frac{dT}{dx}\,, \tag{2.1}$$

where k is the thermal conductivity of the material in the x direction, A is the cross-sectional area and T is the temperature.

Removing the cross-sectional area from the equation introduces the concept of heat flux,

$$q''_x = -k\frac{dT}{dx}\,. \tag{2.2}$$

This time, the appropriate rate equation accounts for the amount of energy being transferred per time and unit area and is known as *Fourier*'s Law, once again proportional to the temperature gradient in the corresponding direction.

At this moment, it is appropriate to generalize the concepts described thus far. To do this, it is necessary to realize that in $2D$ and $3D$ heat is transferred in all of the available directions, *i.e.* is a vector quantity. Consequently, a general form of the *Fourier*'s Law can be written

$$\boldsymbol{q''} = -k\vec{\bigtriangledown}T = -k\left(\boldsymbol{i}\frac{\partial T}{\partial x} + \boldsymbol{j}\frac{\partial T}{\partial y} + \boldsymbol{k}\frac{\partial T}{\partial z}\right), \tag{2.3}$$

where the boldface symbols represent vector quantities, being i, j, k the unit vectors along the x, y, z directions, respectively.

Heat equation

To achieve the major goal of the thermal analysis stated in the beginning of Subsection 2.1.1 (compute the temperature distribution in a block of material), it is necessary to solve the correct form of the heat equation. The process of obtaining the heat equation is now going to be briefly described.

Using the above described concepts, one of the methods that can be used to derive the heat equation is to apply energy conservation to a differential control volume. For this control volume, it is necessary to identify the relevant energy transfer processes and introduce the appropriate rate equations. This mathematical demonstration is thoroughly explained in Bergman et al. [6] and the heat equation for conduction in Cartesian coordinates, simplified for a material with constant thermal conductivity in all directions, which is suited for the case being studied, yields

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} + \frac{\dot{q}}{k} = \frac{1}{\alpha}\frac{\partial T}{\partial t}, \tag{2.4}$$

where $\dot{q}$ is the rate of energy generation per unit volume and $\alpha = k/(\rho c_p)$ is the thermal diffusivity of the material, the ability to conduct thermal energy relative to its ability to store thermal energy. For the definition of α, ρ is the volumetric mass density and c_p the specific heat at constant pressure.

The solution of Equation (2.4) yields the temperature distribution $T(x, y, z)$ as a function of time and is, as mentioned above, the key to conduct an heat conduction analysis.

When applied to a $2D$ domain in a steady state regime, the heat equation becomes simpler

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\dot{q}}{k} = 0, \tag{2.5}$$

and, resorting to an adequate computational strategy, is adequate to study the temperature distribution in the solid phase of the type of vascular materials that is under analysis.

In order to solve any differential equation, it is necessary to have the correct BCs. The possible types of BCs used to solve the heat equation can be found in Subsection 2.1.3.

2.1.2 Heat Convection in the Fluid Phase

Considering that the actively-cooled vascular materials under analysis have a moving fluid phase and according to Soghgrati et al. [16] and Soghrati and Geubelle [9], the phenomenon of heat advection is not only non negligible but also dominant in describing the heat transfer process.

In addition to the heat transferred by conduction, which is described above in Subsection 2.1.1, heat is also transferred *via* the macroscopic motion of the fluid if a temperature gradient exists. The total

heat transferal by convection is thus constituted by the combination of two effects (Bejan [17]): heat conduction and this newly described process that takes place due to the macroscopic motion of the fluid (bulk fluid motion), which is often called advection.

The concept of convection is somewhat broader than conduction in the sense that there are various types of convection, which means that it is convenient to categorize it. Considering the vascular materials that are being studied, the nature of the flow can be said to be both internal, meaning it is confined by a surface and forced, with an external driving force. Figure 2.2 provides a clear overview of the relevant categories that are used to define convection.

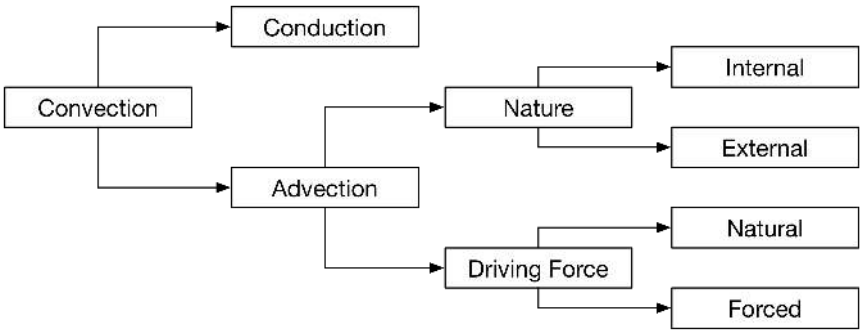
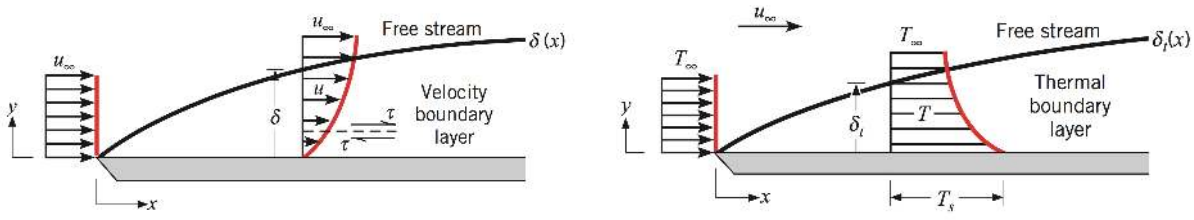


Figure 2.2: Concept and classifications of heat convection.

The mathematical concept of convection (the heat equation that defines it) has one more term than conduction. As the goal here is not a thorough mathematical review and the cases being studied are simplified (mostly because they are $2D$), only a few key concepts are presented.

To study convection (even in a brief approach like this one), it is necessary to have a basic understanding of the concept of boundary layer (BL). In the case of the internal flow being studied, the fundamental concepts are those of velocity BL and thermal BL (see White [18]). A velocity BL develops whenever there is a fluid motion over a body due to the non-slip condition in the surface and the dimension of this BL is dependent on the presence of velocity gradients and shear stresses within the particles of the flow. On the other hand, the thermal BL develops if there is a temperature gradient between the fluid stream and the surface and the dimension depends on the temperature gradient and properties of the flow. Both types of BLs are represented in Figure 2.3.



(a) Velocity BL (b) Thermal BL

Figure 2.3: Relevant boundary layer types (from Bergman et al. [6]).

The result of the existence of these BLs is translated (when implementing the flow in the cooling

channels) into parabolic flow velocity profiles inside them. These velocity profiles are explained in further detail when needed to model the flow inside the cooling channels. It is important to mention that the fluid flow is considered to be fully developed throughout the domain, which means that that effects of both velocity and thermal BLs have spread throughout the fluid in the channel all over the domain.

The added complexity in the energy balance (where, like in what was described for heat conduction, it is necessary to identify the relevant energy transfer processes and introduce the appropriate rate equations) due to the existence of a moving fluid in the medium means that it is also necessary for perform a mass flux analysis.

Considering all that is mentioned above about the temperature distribution resulting from convection and the fact that it results in an added term to the equations that have to be solved in order to obtain temperature distributions, these equations are only presented in following chapters, where their terms can explained in the context of implementation, which is more useful than a vague description here.

Finally, it is worth mentioning that the BCs that are presented in Subsection 2.1.3 also apply to the more complete heat equations that need to be solved also considering convection within the domain.

2.1.3 Boundary Conditions

To get a solution for the heat equation (resulting in the temperature distribution), the correct BCs must be used. These correct BCs affect the solution and correspond to physical conditions existing in the boundaries of block of material being analyzed. If the problem is not steady-state, *i.e.* is time-dependent, the initial conditions in the medium also need to be accounted for.

Analyzing Equation (2.5) one logically concludes that, because this equation is second order in space, it is necessary to enforce two BCs for each coordinate needed to describe the system. On the other hand, because the type of problems being analyzed is steady-state (stationary problems), there is no need to specify a temporal BC.

As the name implies, BCs are applied to the boundary of the domain under analysis, which means that, for the $2D$ domains being studied, they can be applied to the line edges of the domain. The BCs relevant to the problem can be divided into three major categories (see Bergman et al. [6]):

1. Dirichlet BC (or BC of the first kind) - the temperature of the edge in question is fixed.

2. Neumann BC (or BC of the second kind) - fixed heat flux, related to the temperature gradient at the edge in question by Fourier's law (Equation 2.3)

$$- k\frac{T}{x_i}\bigg|_{x_i=0} = q_S" \tag{2.6}$$

3. Robin BC (or BC of the third kind) - this kind of BC is related to the existence of a convection heat transfer process at the edge in question and it is not used during the course of this work.

2.2 Mathematical Fundamentals in Solid Mechanics

As mentioned, the concepts and equations provided by the field of solid mechanics are used only as a source of results to check the implementation of the discretization method chosen to model the vascular materials, which means that this section is short and concise to envision only the basics.

The first step to explore the relevant concepts is to consider a $3D$ body made of a linear elastic material, which is subject to point loads, meaning they are applied to a specific point on the domain, as well as body forces which act throughout the volume of a body, for example gravity. The equilibrium equation for the body being considered is (from Reddy [19])

$$\nabla \cdot \boldsymbol{\sigma} + \mathbf{b} = 0\,, \tag{2.7}$$

where $\boldsymbol{\sigma}$ is the stress along the domain and $\mathbf{b}$ is the body force. This equation states that the sum of the point loads (left term) with the body force (right term) is null, which means that the body is in equilibrium.

Although this work focuses on $2D$ problems and the verification of the discretization method's implementation takes place using test cases of this dimension, it is relevant to review the $1D$ linear elastic deformation equations of bars under tension because it is also used (in Chapter 3) to present a simple explanatory example of this discretization method. On the other hand, the analytic solution for one of the test cases for verification can also be approximated to a $1D$ field. Taking only the x component in Equation 2.7, it is possible to obtain the equilibrium equation in $1D$ with an integration in y and z

$$\frac{d\sigma}{dx} + b(x) = 0\,. \tag{2.8}$$

Considering the constitutive equation for linear isotropic materials, also known as Hooke's law ($\sigma = \varepsilon E$), one obtains the first expression in Equation (2.9). Considering that the extension is nothing more than the derivative of the axial displacement caused by the applied loads, it is possible to obtain the second expression. On the other hand, for a material with constant Young's modulus, one can write the last expression,

$$\frac{dEA\varepsilon}{dx} + b(x) = 0 \Rightarrow \frac{d}{dx}\left(EA\frac{du}{dx}\right) + b(x) = 0 \Rightarrow EA\frac{d^2u(x)}{dx^2} + b(x) = 0\,, \tag{2.9}$$

where E is the Young's modulus of the material, ε is the strain and $u(x)$ is the axial displacement.

According to Gould [20], the necessary equations to deal with a linear elastic solid under static equilibrium like the one in Figure 2.4 have already been combined and obtaining the displacement solutions now comes down to integration and enforcing BCs.

Returning to the last expression of Equation (2.9) and integrating it twice in space (x), the displacement field solution is obtained as a function of the body force (in this case a distributed load) and of the BCs (to be obtained for each problem).

$$Eu(x) + \int\int b(x)\,dx = C_1 x + C_2\,. \tag{2.10}$$

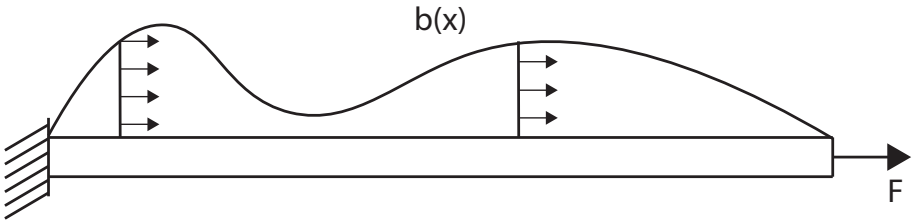


Figure 2.4: $1D$ linear elastic solid under traction load.

The second test problem that is used to verify the discretization model's implementation in Chapter 3 is more complex and cannot be solved using an $1D$ approximation. Because of everything mentioned so far, it is not within the main topic of this work to derive more complex solid mechanics equations, which means that the results of the implementation are compared with analytic solutions obtained from literature (Sukumar et al. [21]) which are presented when necessary.

This ends this chapter on theoretical fundamentals and background, which means that all of the theoretical concepts that describe qualitatively the physical phenomena that is computationally implemented with the techniques discussed in Chapters 3 and 4 have been discussed.

Chapter 3

Discretization Method - IGFEM

3.1 Finite Element Method Fundamentals

As is mentioned above in the introduction to this chapter, the FEM tool is used in this work to approximate the solutions of PDEs and get approximate boundary values with adequate levels of accuracy. The FEM envisions two main aspects, which are the discretization method (being discussed here) and the weighted residual formulation (which is approached in Chapter 4).

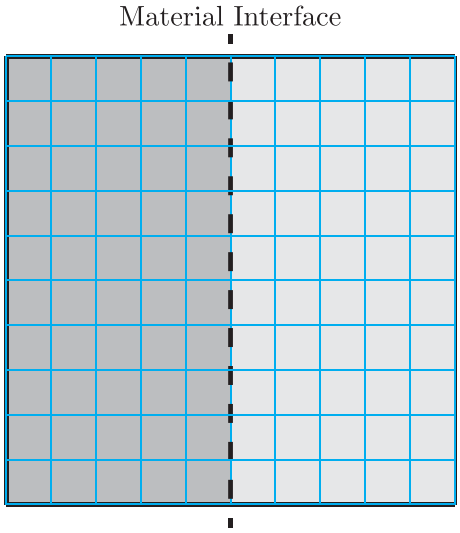
Using just a few lines of text to describe the two main aspects of FEM mentioned (a complete formulation of the FEM is thoroughly presented in Reddy [19]): it divides the problem's domain into smaller elements that together form the finite element mesh and uses approximated functions of the governing equations to create the elemental matrices called stiffness matrices, that contain the coefficients for one element's equations. Then, those elemental matrices are assembled into one global stiffness matrix, resulting in a system of equations that can be solved by enforcing the boundary conditions and that yields the distribution of a main variable. If more results are needed, they can be obtained in an optional phase of post processing.

The FEM has, according to Schweitzer [22], proven to be a very versatile numerical tool over the last decades (the first formulations of this method can be traced back to as far as 1941 in Hrennikoff [23]). However, to utilize this versatility, the discretization of the domain has to be made in such a way that a good quality conforming mesh (that accounts for all of the shapes and borders of the domain that is being studied) is generated, regardless of the underlying physics. Characterized by its refinement and the aspect ratio of the constituting elements, the finite element mesh has, according to Cuillière et al. [24], a crucial impact on the accuracy and stability of the approximation. To model physics and engineering problems, one often deals with with complex geometries, which means that creating a good quality mesh is not an easy task. Despite the effort placed on creating efficient algorithms to create meshes that conform to complex geometrical features since the beginning of FEM (Ho-Le [25]), this process still accounts for a large percentage of the total cost of the analysis.

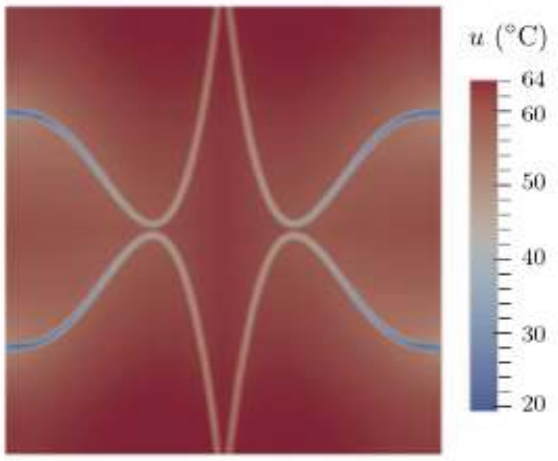
3.2 Requirements

As mentioned in the last paragraph of the previous section, to get accurate solutions using standard FEMs is only possible using a conforming mesh. In the case of domains where there are at least two materials, this requirement can only be addressed using a mesh whose elements boundaries are perfectly aligned with the interface.

This conforming mesh would be relatively cheap and straight-forward to implement for very simple cases like the one in Figure 3.1(a), where the domain is composed of a square plate made out of two solid materials with a straight vertical interface. However, generating a conforming mesh could prove itself to be a complex and computationally heavy or even an impossible task (Yue and Robbins [26]) when dealing with complex structures. Vascular materials are possibly an example of complex geometries to mesh, with a phase change that results in both a change in material properties and governing equations. An example of complex meshing geometries in vascular materials is represented in Figure 3.1(b), where a square plate is intersected by cooling channels. For a better understanding of the problems in correct meshing of a geometry, refer to Beall et al. [27], which provides a very complete overview of the topic.



(a) Material interface problem conforming mesh



(b) Example of vascular material from Soghrati [28]

Figure 3.1: Mesh generation issues.

Summing up the last few paragraphs, in vascular cooled materials different regions of the mesh have not only different material properties but also different governing equations, as the heat advection in the fluid phase is not negligible (Soghgrati et al. [16]). Taking this into account, it is mandatory that the discretization method is efficient dealing with discontinuities in the gradient field.

On the other hand, optimizing any geometry with a standard discretization method requires the generation of several conforming meshes, which is potentially computationally very expensive. This means that choosing a dicretization technique that does not require a full mesh update at each iteration step is also an advantage.

3.3 Interface-enriched Generalized FEM

The limitations of standard FEM in problems that present discontinuities in the gradient field, which is the case of vascular materials, motivated research for modifications to said method and the discretization technique that has been chosen for this work is one of the results of that research.

The first developments in modifying the FEM resulted in the Generalized FEM (GFEM, thoroughly described and explained in Belytschko et al. [29]) / eXtended FEM (XFEM) which, according to Schwebke and Holzer [30], was first introduced by Melenk [31] in 1995. Also according to Schwebke and Holzer [30], Babuska et al. [32] and Belytschko et al. [29], the *GFEM* is a direct extension of the standard finite element method and combines the desirable features of the standard finite element method with meshless methods. This leads to the possibility of obtaining an accurate solution of engineering problems in complex domains with discontinuities, singularities and localized deformations (using nonconforming meshes) which could be practically impossible to solve using the standard FEM.

According to Soghrati et al. [10], the potential of this method is obtained by using *a priori* knowledge of the solution field in the numerical approximation (using enrichment functions). According to Melnek and Babuska [33], these enrichment functions are obtained using the partition of unity method and consequently present the possibility of solving problems with field/gradient discontinuities while using nonconforming meshes to discretize the problem.

Going one step further in the direction of modeling vascular materials, which is a material or phase interfaces problem, yet another method that has been developed (this time derived GFEM / XFEM) must be considered. The Interface-enriched GFEM (IGFEM) was first introduced in 2011 by Soghrati et al. [10] and intends to improve on the standard GFEM with an easier implementation and handling of Dirichlet boundary conditions (refer to Chapter 2 for the formulation of the different types of boundary conditions) and a lower computational cost (among others).

According to Soghrati and Geubelle [34], these new characteristics (improvements) are achieved using new enrichment functions (which go to zero at the standard nodes of enriched elements (by construction), allowing for a straightforward imposition of Dirichlet boundary conditions on enriched elements that conform to the Dirichlet boundary) and a different approach for applying them at the interface. In the latter, instead of applying the generalized degrees of freedom to the nodes of the original mesh, as is done in the normal GFEM, they are applied to the new nodes on the intersection of the phase interface with element edges (hence the name Interface-enriched).

Considering all of the advantages of the IGFEM already described, it is chosen to deal with the geometric modeling challenges resulting from the phase interface(s). According to Tan et al. [35], IGFEM is very interesting in problems with complex geometries and particularly well suited for this task of designing and optimizing vascular composites, as the same nonconforming mesh can be used to model several vascular configurations.

Other applications of the IGFEM are, among others, contact mechanics and crack propagation problems. This work's implementation of IGFEM has already been adapted into a state-of-the-art implementation of a crack propagation formulation done in *hybrida* (see Aragón [15]). Another master thesis

project from the *TU Delft* research group that supported this work in the topic of contact mechanics is currently being studied and will also use the base provided by the IGFEM implementation developed for this project.

After conceptually describing IGFEM and before presenting algorithms, it is convenient to clarify on its additional implementation steps over standard FEMs. Breaking down the method in two big milestones and specifying for the case of vascular cooled materials results in the following steps:

1. Modifying the original mesh - consists in subdividing the elements that are cut by the phase interface (parent elements) into smaller triangular elements (children or integration elements). The phase interface should be included in the new elements' edges. According to Soghrati [28], the reason for the children or integration elements to be triangular is simply to simplify the computation of the enrichment functions;

2. Establishing the enrichment functions - this is the task where the differences between IGFEM and the normal GFEM lie. This step's description requires a mathematical is the formulation of the method and is thoroughly described in Subsection 3.3.1. IGFEM is a very recent method and, because it is still under research in several aspects, modifications to its formulation continue to appear within the scientific community. Because of this, the presented formulation already presents one improving modification (that is presented later in this chapter). The latest developments on the topic can be found in Soghrati [28], Soghrati et al. [36] and Soghrati and Barrera [37].

3.3.1 Formulation

All FEMs are computational tools that use approximate functions to a problem's governing equations in the smaller domains that are the finite elements to obtain an approximated solution. To the set of equations that the solution must satisfy over the domain to define the problem (normally a differential equation and the boundary conditions required to solve it) is given the name of strong form. On the other hand, the weak form of the problem states the conditions that the solution must satisfy in an integral sense and it is the equation to be approximated by FEMs.

Using GFEMs, the aforementioned approximated solutions are obtained using

$$u^h = \underbrace{\sum_{i=1}^{n} N_i^p u_i}_{std.\ FEM} + \underbrace{\sum_{j=1}^{n_{en}} s_j \psi_j \alpha_j}_{enriched} , \tag{3.1}$$

where u^h is the approximated solution for the main variable, N_i^p is a set of n Lagrangian shape functions evaluated in the original element (of the nonconforming mesh). According to Zienkiewicz and Taylor [38], the standard Lagrangian shape functions are a set of standard polynomial functions of the least degree that at each point x_j assume the corresponding value y_j and that respect the partition of unity principle, $\sum N_i = 1$. On the other hand, u_i represents the main variable of interest at the node labeled i in the mesh being used and s_j is an additional scaling factor that imposes that a well-conditioned stiffness matrix is constructed. The improvements on the original formulation of IGFEM introduced by

the implementation done for this thesis in *hybrida* are related to this scaling factor and are discussed in Subsection 3.6.1. Finally, ψ_j and α_j are a set of enrichment functions and the generalized degree of freedom associated with the j^{th} interface node created from the intersection point of the material interface with element edges, respectively.

As shown in Equation (3.1), the approximation of the variable field for GFEMs is the sum of the standard FEM approximation with the contribution of the enrichment functions to capture gradient discontinuities (these enrichments vary from one GFEM formulation to another and the method for obtaining enrichment functions for IGFEM is presented below).

Enrichment functions

The enrichment functions for a certain degree of freedom in the IGFEM formulation are obtained by evaluating the standard shape functions in the child element. This means that, after discretizing the parent element into smaller children elements it is only necessary to combine the standard shape functions for triangular elements evaluated at the parent and children elements functions in order to obtain the complete approximation.

Obtaining the enrichment functions for applying IGFEM is described throughout the set of references already provided in this chapter (Soghrati [28] provides the latest developments in this formulation and an overview of what has been researched before). However, to clearly understand the procedure, a sample problem has been solved by hand and is presented below in Subsection 3.3.2.

3.3.2 $1D$ Sample Problem

This subsection aims at providing an example that gives a clear understanding of the IGFEM formulation, mostly focusing differences from the standard method. A simple $1D$ bar with an absolute length of $L = 3\,m$ and cross-sectional area of $A = 1\,m^2$ made out of two materials is considered. The material interface is in the middle of the bar, at $x_\Gamma = 1.5\,m$, and the characteristics of both materials are

$$\begin{cases} E_1 = 1\,Pa & \text{if } x \leq 1.5 \\ E_2 = 10\,Pa & \text{if } x \geq 1.5\,. \end{cases} \tag{3.2}$$

To solve the problem of computing the displacement throughout the considered bar, the boundary conditions are a fixed left extremity (at $x = 0m$) and a point load $f = 1N$ at the right extremity (at $x = 3m$), as represented in Figure 3.2. To follow this example, basic knowledge in standard FEM concepts is assumed (the steps that are presented to solve the problem follow, once again, the formulation presented by Reddy [19] and correspond to the "universal" FEM procedure used to solve problems),

(a) Illustration (b) FEM model

Figure 3.2: $1D$ IGFEM sample problem geometry.

The first step in applying any FEM is obtaining the finite element mesh (discretization of the domain) and in the considered bar, represented in Figure 3.2(b), the standard FEM discretization is defined by the three finite elements (I, II and III) bounded by two circles. After applying what would be called the *Geometric engine* in *hybrida*'s IGFEM (name given in the implementation to the set of functions that divide the nonconforming elements into integration element, to explain in Section 3.4.1), the result is that element II gets split into two smaller integration elements (IIa and IIb).

The stiffness matrices for elements I and III are obtained in the standard way, using the two standard shape functions for bar elements (shown below) and the standard formulation

$$k = \int_0^l EA \frac{dN^T}{dx} \frac{dN}{dx} dx \,, \tag{3.3}$$

that yields

$$\frac{E_1\, A}{l} = 1 \Rightarrow k^I = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \qquad \frac{E_2\, A}{l} = 10 \Rightarrow k^{III} = \begin{bmatrix} 10 & -10 \\ -10 & 10 \end{bmatrix} \tag{3.4}$$

Regarding element II, obtaining an accurate displacement solution is not possible while using standard FEM, which can be solved by employing the IGFEM formulation in

$$u^h = \sum_i N_i^{II} u_i^{II} + \psi\alpha = \begin{Bmatrix} N_1^{II} & N_2^{II} & \psi \end{Bmatrix} \begin{Bmatrix} u_1^{II} \\ u_2^{II} \\ \alpha \end{Bmatrix} , \tag{3.5}$$

where N_i^{II} are the standard Lagrangian shape functions evaluated in element II and u_i^{II} are the degrees of freedom of the original nodes 2 and 3. This means that the term $\sum_i N_i u_i$ corresponds to a standard FEM formulation. On the other hand, ψ is the enrichment function (that is the standard shape function evaluated in elements IIa or IIb) and α is the degree of freedom in the node inserted at half length, in the material interface. The standard $1D$ Lagrangian shape functions in global coordinates (being a $1D$ simple example, there is no need for a coordinate transformation) are

$$N_1^{II} = \frac{x_2 - x}{x_2 - x_1} = \frac{x_2 - x}{l} \,, \tag{3.6}$$

$$N_2^{II} = \frac{x - x_1}{x_2 - x_1} = \frac{x - x_1}{l} \,, \tag{3.7}$$

where x_1 and x_2 are the left and right coordinates of element II, respectively, and l is its length.

The enrichment function is a linear combination of the two Lagrangian shape functions

$$\psi = N_2^{IIa} + N_1^{IIb} \,. \tag{3.8}$$

The integral definition for the stiffness matrix in a generic finite element with a length of l is again given by Equation (3.3), which means that obtaining the stiffness matrix of element II comes down to summing the stiffness matrices of the integration elements, which are obtained by solving the integrals

$$k^{IIa} = \int_0^l E_1 A \left(\frac{dN^T}{dx}\right)^{IIa} \left(\frac{dN}{dx}\right)^{IIa} dx \tag{3.9}$$

and

$$k^{IIb} = \int_0^l E_2 A \left(\frac{dN^T}{dx}\right)^{IIb} \left(\frac{dN}{dx}\right)^{IIb} dx\,. \tag{3.10}$$

From Equation (3.8), the enrichment function is a linear combination of the standard shape functions evaluated at the child element and can be obtained using

$$\psi = \begin{cases} \frac{x-x_1}{x_\Gamma - x_1} & \text{if } x_1 \le x \le x_\Gamma \\ \frac{x_2-x}{x_2-x_\Gamma} & \text{if } x_\Gamma \le x \le x_2 \end{cases} \Rightarrow \psi = \begin{cases} \frac{x-x_1}{l_1} & \text{if } x_1 \le x \le x_\Gamma \\ \frac{x_2-x}{l_2} & \text{if } x_\Gamma \le x \le x_2 \end{cases}\,, \tag{3.11}$$

where $l_1 = l_2 = 0.5\,m$ are the dimensions of the children elements to the left and right of the interface, respectively.

Recalling the IGFEM formulation in Equation (3.5), the shape function vector is defined by parts by the shape functions for element II and the correct shape function for the child element,

$$N = \begin{Bmatrix} \frac{x_2-x}{L} & \frac{x-x_1}{L} \end{Bmatrix} \begin{Bmatrix} \frac{x_2-x}{l_1} & \text{if } x_1 \le x \le \alpha \\ \frac{x-x_1}{l_2} & \text{if } \alpha \le x \le x_2 \end{Bmatrix} \begin{Bmatrix} u_2 \\ u_3 \\ \alpha \end{Bmatrix} \Rightarrow \frac{dN}{dx} = \begin{cases} \begin{Bmatrix} -\frac{1}{L} & \frac{1}{L} & \frac{1}{l_1} \end{Bmatrix} & \text{if } x_1 \le x \le \alpha \\ \begin{Bmatrix} -\frac{1}{L} & \frac{1}{L} & -\frac{1}{l_2} \end{Bmatrix} & \text{if } \alpha \le x \le x_2 \end{cases} \tag{3.12}$$

Having computed all of the necessary terms, it is now possible to determine the stiffness matrices in element II. The results are

$$k^{IIa} = \int_{x_1}^{\alpha} \underbrace{E_1 A}_{1} \begin{Bmatrix} -\frac{1}{L} \\ \frac{1}{L} \\ \frac{1}{l_1} \end{Bmatrix} \begin{Bmatrix} -\frac{1}{L} & \frac{1}{L} & \frac{1}{l_1} \end{Bmatrix} dx \Rightarrow$$

$$\Rightarrow k^{IIa} = \begin{bmatrix} \frac{1}{L^2} & -\frac{1}{L^2} & -\frac{1}{Ll_1} \\ -\frac{1}{L^2} & \frac{1}{L^2} & \frac{1}{Ll_1} \\ -\frac{1}{Ll_1} & \frac{1}{Ll_1} & \frac{1}{l_1^2} \end{bmatrix} l_1 = \begin{bmatrix} \frac{l_1}{L^2} & -\frac{l_1}{L^2} & -\frac{1}{L} \\ & \frac{l_1}{L^2} & \frac{1}{L} \\ sym. & & \frac{1}{l_1} \end{bmatrix} = \begin{bmatrix} 0.5 & -0.5 & -1 \\ & 0.5 & 1 \\ sym. & & 2 \end{bmatrix} \tag{3.13}$$

$$k^{IIb} = \int_{\alpha}^{x_2} \underbrace{E_2 A}_{10} \begin{Bmatrix} -\frac{1}{L} \\ \frac{1}{L} \\ -\frac{1}{l_2} \end{Bmatrix} \begin{Bmatrix} -\frac{1}{L} & \frac{1}{L} & \frac{1}{l_2} \end{Bmatrix} dx = 10 \begin{bmatrix} \frac{1}{L^2} & -\frac{1}{L^2} & \frac{1}{Ll_2} \\ & \frac{1}{L^2} & -\frac{1}{Ll_2} \\ sym. & & \frac{1}{l_2^2} \end{bmatrix} = \begin{bmatrix} 5 & -5 & 10 \\ & 5 & -10 \\ sym. & & 20 \end{bmatrix}. \tag{3.14}$$

The next step is to assemble the elemental stiffness matrices into a global stiffness matrix. The procedure is similar to a standard FEM (the fifth line and column correspond to α, the degree of freedom of the inserted interface node). To avoid a a repetitive description of the assembly of matrices k^I, k^{IIa}, k^{IIb} and k^{III} it is conveniently illustrated by

$$K = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 1+0.5+5 & -0.5-5 & 0 & -1+10 \\ 0 & -0.5-5 & 10+0.5+5 & -10 & 1-10 \\ 0 & 0 & -10 & 10 & 0 \\ 0 & -1+10 & 1-10 & 0 & 2+20 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 6.5 & -5.5 & 0 & 9 \\ 0 & -5.5 & 15.5 & -10 & -9 \\ 0 & 0 & -10 & 10 & 0 \\ 0 & 9 & -9 & 0 & 22 \end{bmatrix}. \quad (3.15)$$

In order to be able to solve the problem, it is necessary to apply the boundary conditions to the global stiffness matrix obtained above by eliminating the line and column corresponding to the imposed displacement. Because in the example being considered node 1 is fixed (null displacement), it is necessary to eliminate the first line and column, resulting in

$$K_{f_1} = \begin{bmatrix} 6.5 & -5.5 & 0 & 9 \\ -5.5 & 15.5 & -10 & -9 \\ 0 & -10 & 10 & 0 \\ 9 & -9 & 0 & 22 \end{bmatrix}. \quad (3.16)$$

The displacement within the bar can be computed by

$$K_{f_1} u_{f_1} = f_{f_1} \Rightarrow u_{f_1} = K_{f_1}^{-1} f_{f_1} \quad (3.17)$$

and the result is given by 3.18

$$u_{f_1} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1.55 & 1.55 & 0.225 \\ 1 & 1.55 & 1.65 & 0.225 \\ 0 & 0.225 & 0.225 & 0.1375 \end{bmatrix} \begin{Bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{Bmatrix} \Rightarrow u_{f_1} = \begin{Bmatrix} u_2 \\ u_3 \\ u_4 \\ \alpha \end{Bmatrix} = \begin{Bmatrix} 1 \\ 1.55 \\ 1.65 \\ 0.225 \end{Bmatrix} \quad (3.18)$$

To obtain the modified solution u^h, with the correct values at the enriched node, it is necessary to use the formulation presented in Equation (3.5). Because the materials are both elastic, the plot of the displacement for the load is linear in the domain with a change in slope at the interface (see Figure 3.3). In Equation (3.19), the use of the IGFEM formulation is demonstrated by computing the displacement at the interface location $x_\Gamma = 1.5$,

$$u^h(x_\Gamma) = N_1(x_\Gamma)\times 1 + N_2(x_\Gamma)\times 1.55 + 1\times 0.225 \Rightarrow u^h(x=1.5) = \frac{1}{2}\times 1 + \frac{1}{2}\times 1.55 + 1\times 0.225 = 1.5\,, \quad (3.19)$$

where $N_1(x_\Gamma)$, $N_2(x_\Gamma)$ and $\psi(u_4)$ are the values of the standard shape functions and the value of the enrichment function at the interface node, respectively.

In order to verify that this example is correctly solved and that the IGFEM is adequate for this kind

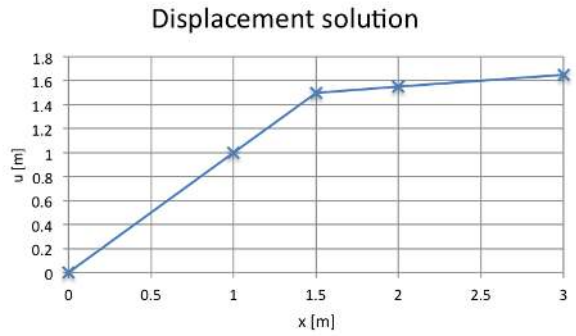


Figure 3.3: Solution of $1D$ bar IGFEM example.

of problem, it is convenient to check the obtained result against the analytic solution. Recalling what is presented in Section 2.2 and Equation (2.10) and applying the correct BCs,

$$\begin{cases} u(x=0) = 0 \Rightarrow C_1 \times 0 + C_2 = 0 \Rightarrow C_2 = 0 \\ E\frac{du(x)}{dx}\Big|_{x=L} = f \Rightarrow C_1 = f \end{cases} \Rightarrow u(x) = \frac{fx}{EA} \tag{3.20}$$

Because the domain is composed of two materials, the analytic displacement solution is, as already seen for the approximated case, defined by parts

$$u(x) = \begin{cases} \frac{fx}{E_1} = x & \text{if } 0 \leq x \leq L/2, \\ \frac{fL}{2E_1} + \frac{f(x-L/2)}{E_2} = 1.5 + \frac{(x-1.5)}{10} & \text{if } L/2 \leq x \leq L\,. \end{cases} \tag{3.21}$$

From Figure 3.3 and Equation (3.21), it is confirmed that the IGFEM solution and the analytic solution are exactly the same, which means that there are no mistakes in the process and that the method is adequate for this interface problem.

3.4 Implementation

3.4.1 *Geometric Engine*

The “first step” of applying IGFEM is, as described in Section 3.3, to modify a non-conforming finite element mesh by creating children elements that result of the division of elements from the original mesh that are intersected by the interface and whose boundaries conform to said interface. This section describes the mentioned step, which can be referred to as *geometric engine*, mentioning the difficulties that need to be overcome and some of the algorithms that are used to do it. Finally, some simple applications are presented.

To frame the implementation of this functionality, the diagram in Figure 3.4 goes a bit deeper within the *hybrida* code structure (recall Figure 1.4) and shows where the involved modules are located. In this diagram, the module in the rectangle *geometric engine* is responsible for all mesh related operations and the rectangle *igfem manager* is where information is managed for later use in the assembly of the

matrices.

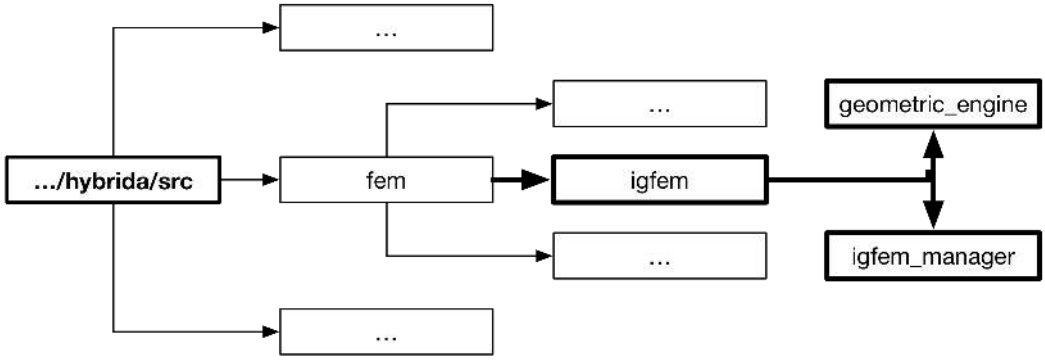


Figure 3.4: Modifications to code's structure due to geometric engine.

As previously stated but worth clearly mentioning again, the only limitation for new children elements is that their boundaries must conform with discontinuity edges or surfaces of the domain. On the other hand, unlike in normal FEM elements, their aspect ratio does not affect the accuracy of the solution (Soghrati et al. [10]).

The reason to devote such attention to this geometric engine in this report is that, when dealing with finite element meshes, the variety of scenarios that can occur from slicing the finite elements implies that imaginative solutions must be employed. Being integrated in a larger project (where the geometric engine is already being used in several other implementations, such as crack propagation, periodicity problems and contact mechanics), the goal here is to present algorithms that lead to a powerful and general enough implementation, coping with the possibility of problems with different underlying physics and various mesh geometries without requiring deep tweaks by the user.

From an implementation point of view, the task of obtaining the integration elements for a material interface problem is, in the case of this project, facilitated by the fact that it starts from a FEM tool that already has a lot of basic features implemented (*hybrida*).

Taking this simplification into account, the *geometric engine*'s implementation starts by receiving a set of adequate data structures that already contain all of the information needed to use the original mesh (types of elements, node mapping, total number of nodes, *etc.*) and the composing material (initially the domain is uniform). The last thing needed to run the implementation of IGFEM that is explained below is information on the interfaces, which must be in the form of level set functions.

From Abbena et al. [39], the level set of a function is a set of values for which the function equals a given constant. The shape of the level set of a function depends on the dimension that is being considered (for example, when the number of variables is two, a level set is generically a curve, refer to the applications presented below in this section for explicit examples). These are used in such a way that the level set function returns a value dependent on the coordinates of the node being tested and the sign of that value is used to assess the material characteristics and governing equations.

With all of the premises and requirements listed and described, the most comprehensive algorithm for the *geometric engine* is presented in Algorithm 1.

Algorithm 1 Geometric engine algorithm

```
1: Input: FE mesh, interfaces information (list of functions and new material)
2: Output: original data structures modified to envision integration elements and interface nodes
3:
4: for each interface do
5:     compute level set value of all mesh nodes using coordinates
6:     for each element in the original FE mesh do
7:         if same level set sign for all nodes in the element then
8:             if negative level set sign then
9:                 change material of the element
10:                continue to next element
11:            else
12:                continue to next element
13:            end if
14:        else
15:            create line section using the points of zero level set value
16:            slice element
17:            arrange element connectivity
18:            create children elements
19:        end if
20:    end for
21: end for
```

Combining the need for further discussion on some of the steps (functions) in Algorithm 1 that do not justify presenting the complete script with some remarks on the *pseudo code*:

- For line 7 of the *pseudo code*, the assertions are made using an user specified precision (can be changed). In fact, because this is the field of computation, the values obtained always have some sort of numerical error, which means that for all assertions of the kind from now on the conclusions are made based on this specified precision;
- In *pseudo code* lines 14 and 15 is where the actual modification of the mesh begins. To do so, elements that have nodes with different level set signs are searched for and a line segment object is created with the two points of zero level set (obtained *via* linear interpolation);
- Also starting in line 14, geometric algorithms start being employed. For the functions *segment*, *slice* and *create children*, a combination of algorithms inspired in video game development (Ericson [40]) and newly introduced ones (that arise from the IGFEM's formulation needs) are used. These are thoroughly presented below.

After the processing the mesh with this implementation of the geometric engine, the original data structures containing all of the relevant information (elements, nodes, materials, *etc.*) are modified, being ready for the remaining steps of the finite element method.

The first sub algorithm that is relevant to present here is the *slicing* of the original mesh element that is intersected by an interface into two sets of nodes ordered in a counterclockwise (CCW) manner (convention), depending on the side of the interface on which they lie. The Algorithm 2 has been especially created for this implementation and is possibly the most elaborate one within the geometric engine due to the wide range of possible scenarios that can result from the intersection of an interface with the various types of elements and their orientation.

Algorithm 2 Slice element algorithm

```
1: Input: element data (connectivity and coordinates) and line segment
2: Output: two data structures with nodes to left and right of the interface
3:
4: use first node of the element as current node
5: while current node is not the first one and not currently on first iteration do
6:     determine on which side of the interface the current node lies (using level set sign)
7:     if current node lies on one of the sides of the interface then
8:         add it to appropriate container
9:     else
10:        add it to both sides
11:    end if
12:    if side of current node and side of the next one are different and neither is null then
13:        find intersection between element's edge and interface
14:        add it to both sides
15:        compute scaling factor, s_j
16:    end if
17:    use next node of the element as current node
18: end while
```

Once again, it is convenient to make some remarks regarding Algorithm 2, since the reason why some steps need to be taken and how they are taken may not be clear enough:

- Line 6 of the algorithm is fundamental to its functioning. As is stated, it is where the program is able to determine the side of the interface on which the node being tested is and that is done using one of the game developing algorithms that is implemented in the segment object called *leftTurn*. Despite being fundamental to the *slice* algorithm, the assertion made by *leftTurn* is done simply by comparing the cross product in the z coordinate of two vectors;
- The *if-else* statement in lines 7-10 deals with the storing the node tested by *leftTurn* in the appropriate container. If this point lies neither to the left or to the right of the line segment (feature of the *leftTurn* algorithm, means that one of the nodes in the original mesh is already an interface node) it is on it and should be added to both the right and left containers because it is an interface node;
- Line 12 on the other hand checks if the sign of the level set changes from the node being tested to the next one. If it does, that means that the element's edge formed by those two nodes is intersected by an interface, which means that the intersection point must be computed (line 13) and once again added to both containers (line 14);
- The last step in this algorithm is taken care of in line 15 and is the computation of the enrichment functions' scaling factor, s_j. This scaling factor is one of the aspects that has been investigated during the implementation of the IGFEM done for this thesis and is briefly discussed in Subsection 3.6.1.

As mentioned above, the functionality of *slicing* finite elements requires imaginative solutions regarding its implementation. Consequently Algorithm 2 is not presented with all of its tests and checks to ensure that utilizing the resultant data structures can be used successfully later on. Additionally, there are "degenerate" cases (for example the interface being aligned with the element's edge) for which the procedure is not exactly as described.

To make the implementation of the assembly routines simpler, namely the computation of parent element shape functions and enrichment functions, it has been decided that, whatever the quantity of interface nodes in the integration elements is, they should be in the last positions of the nodes array. To do that, another algorithm has been created and although it is not completely straightforward, it is not really relevant to present it in detail here, as the algorithm would be too focused in the way the data structures are implemented and in the structure of *Python* code itself.

Following the procedure in Algorithm 1, the next step is to take the groups of nodes to the left and right of the interface and create triangular assembly elements. This is taken care of by the function *create children*, which is presented in Algorithm 3.

Algorithm 3 Integration elements creation algorithm

```
 1: Input: group of nodes ordered in CCW manner
 2: Output: connectivity of resulting integration element(s)
 3:
 4: check number of input nodes and select corresponding rules for creating integration elements
 5: for each rule do
 6:     create integration elements with node indexes (in the input data structure) in the rule
 7:     add new element to adequate data structure
 8:     if level set sign of first node in the new integration element is negative then
 9:         material of integration element is the new one supplied by the interface computational object
10:     else
11:         material of the integration element is the original material
12:     end if
13: end for
14: for each integration element created do
15:     store information on corresponding parent element
16: end for
```

Creating the integration elements is also an important functionality of the *geometric engine*, as they need to conform to the requirements already described. It is also in this function (as is documented in Algorithm 3) that the correct material for each of these integration elements is assigned. Like in the other algorithms that have already been presented, this one demands some clarification on its procedure:

- The first step is to select the rules for creating the integration elements according to the number of nodes in the input data structure. These rules consist of the node indexes that must be selected in the original input data structure in order to obtain integration elements with at least one interface node and the number of rules is the same as the number of integration elements to create by one run of this function;

- Lines 8-12 of the pseudo-code take care of selecting and storing information on the material for the current integration element. The way to do this is similar to what has already been explained for the standard elements with the difference that the integration elements don't have an original material because they have just been created;

- For the purposes of the assembly routines, namely computing the parent shape functions, it is necessary to store the information on what is the corresponding parent element to the recently created integration element. This is done in lines 14-16 of the pseudo-code;

- As is done for the previous algorithms, not all tests and checks that are used to ensure a successful discretization are presented here, but they exist in the code.

With this, the implementation of the main features in the *geometric engine* are presented and explained in adequate detail, which means it is convenient to present some possible applications.

Applications

In order to show the result of applying the *geometric engine*, three examples are presented below. In the first two, the domain is a $2D$ region composed of two materials divided by a material interface that is defined by a level set function, while the third is a first example of a vascular material modeled using *hybrida* and this implementation of IGFEM. The original meshes are created manually using *Gmsh* (an open source finite element mesh generator, see Geuzaine and Remacle [41]).

Aside from exemplifying the use of the *geometric engine*, the first two applications examples also intend to prove its element type independence, as it works in $2D$ all element types, and mesh type independence, as it works for both structured and unstructured meshes.

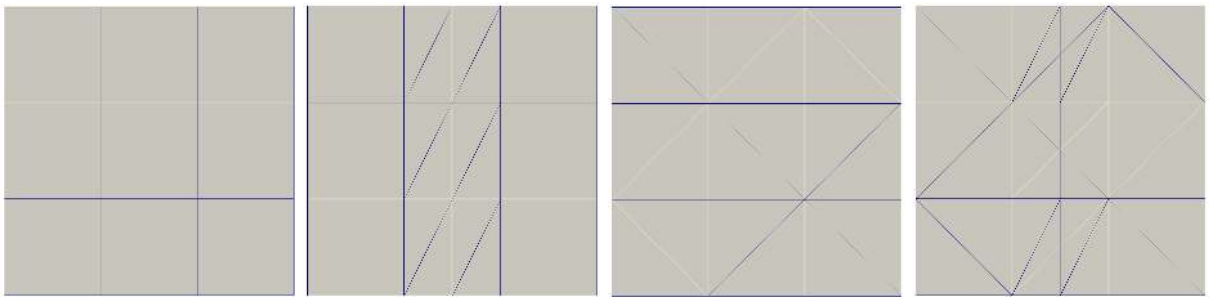
Square domain

As mentioned above, the first example consists of a $2D$ square plate domain (referential placed at the center of the square). Two meshes have been created for this example (Figure 3.5(a) and (c)), being one of them formed by quadrangular elements and the other by triangular ones.

The goal here is to observe the result of introducing a vertical interface using the *geometric engine*. To do so, it is necessary to create the level set function that describes this interface

$$f(\mathbf{x}) = -x + 0 \times y\,. \tag{3.22}$$

In the level set function ($f(\mathbf{x})$) presented Equation (3.22), the return value is a positive number if the node being tested is to the left of a vertical line that goes trough the center of the rectangle and negative if it is to the right.



(a) Original quad. mesh (b) Modified quad. mesh (c) Original tri. mesh (d) Modified tri. mesh

Figure 3.5: Geometric engine result in a square domain.

The meshes that result from the intersection of the original meshes with the interface defined by the level set function are presented in Figure 3.5(b) and (d). In the resulting meshes, all of standard and

children elements to the left of the interface have the original material and all of the elements to the right have the new material, whose properties are passed to the *geometric engine* in the computational object that also contains interface information.

It is important to mention that the implementation of the *geometric engine* being described works for both types of elements tested (in fact, it works for all $2D$ element types) in this case for structured meshes.

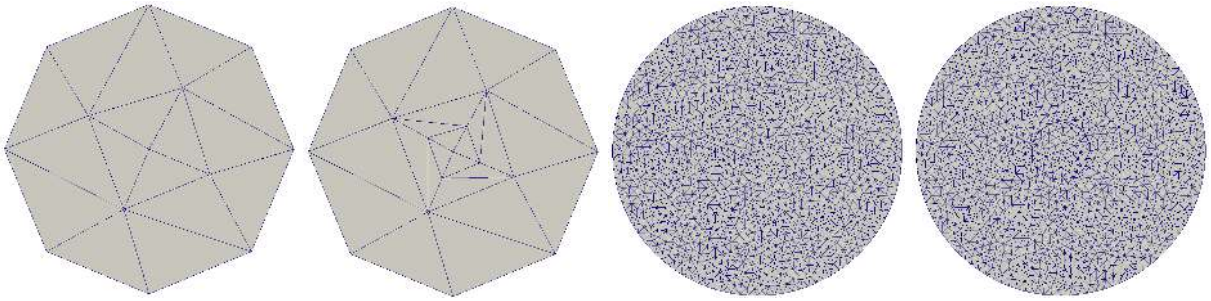
Circular domain

The next example has a similar concept to the square plate with a material interface. This time, the referential is located at the center of a circular domain where there is smaller a circular inclusion centered at the same referential and made of a different material. To conduct this example, two unstructured triangular meshes have been created, one coarser than the other to appreciate the different results, as presented in Figure 3.6(a) and (c).

In this case, the interface is not a straight line segment, it is a circle that can be defined by the level set function (which is based in the equation of a circumference)

$$f(\mathbf{x}) = \sqrt{x^2 + y^2} - r_i \,, \tag{3.23}$$

where r_i is the radius of the circular inclusion. In this case, if the node being tested has the level set function returning a negative value, it is inside the interface and *vice-versa*.



(a) Original quad. mesh (b) Modified quad. mesh (c) Original tri. mesh (d) Modified tri. mesh

Figure 3.6: Geometric engine result in a circular domain.

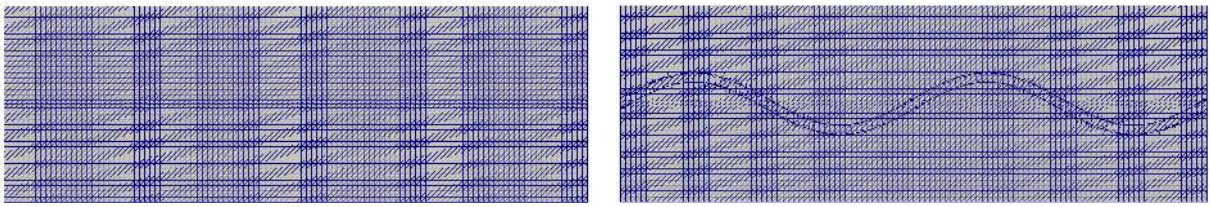
The results are presented in Figure 3.6(b) and (d). In the resulting meshes, all elements of the matrix (outside the inclusion) have the original mesh material and the elements inside the inclusion have the material supplied by the interface computational object.

Looking at both the square plate example and this one, it is possible to conclude that the *geometric engine* is also mesh type independent, since the first example has been presented with structured meshes and this one with unstructured ones.

Vascular material

Finally, and to show some preliminary steps in the way of modeling vascular materials, the *geometric engine* has been used on the mesh in Figure 3.7(a) to create a $2D$ rectangular plate with a vascular sinusoidal channel intersecting it, which is presented in Figure 3.7(a).

There are two interfaces in this case, which means that two level set functions are needed to model it. However, the process of creating them is similar to what has been described in the previous examples and it is further discussed in Chapter 5.



(a) Original quad. mesh (b) Modified quad. mesh

Figure 3.7: Preliminary modeling of vascular material.

3.4.2 Elemental Assembly

In this subsection, the goal is to provide some insight on the modifications that are needed to certain aspects of the code that are not ready for the complexity needed in this project. These modifications are mainly needed due to the assembly of the integration elements. This explanation does not have the goal of going into detail on the theory behind numerical integration or isoparametric mapping of elements, as they are on the basis of most FEM implementations of this complexity.

These algorithms are located in the mesh module, more specifically in the *element* module and in the *triangle* module, as is presented in Figure 3.8. The modifications to the *element* module are related with two aspects: the first is that integration elements need to be separated from standard elements due different assembly routines (that are implemented in the *triangle* module), while the second is related to the fact that the fluid elements have different governing equations (to further discuss in Chapter 4).

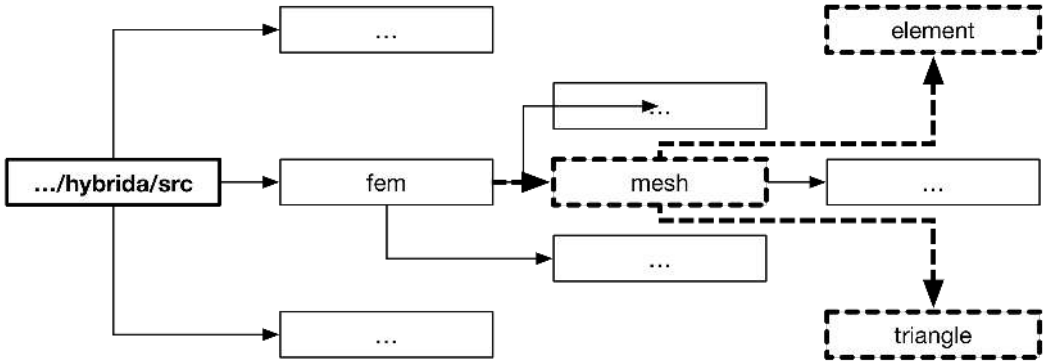


Figure 3.8: Modifications to code's structure due to assembly.

Before going any further, it is important to note that the solution obtained in the end of using the

algorithms presented in this chapter is not the final solution to the problem at the interface nodes (some post processing is required). Due to the formulation of the method, the solution at the interface nodes is the value of the enrichment and not the actual solution to the problem, which means that some post-processing is due in order to obtain a continuous solution.

The most generic algorithm for computing the stiffness matrix for one finite element can be found in the code's module *triangle* and is presented in Algorithm 4.

Algorithm 4 Elemental assembly data algorithm

```
1:  Input: information for element being assembled
2:  Output: elemental stiffness matrix, force vector and all information needed for global assembly
3:
4:  if element type isn't supported (because it is an integration element) then
5:      use "special" assembly routine for integration elements
6:  else
7:      get data for the element
8:      for each quadrature point in the element do
9:          compute assembly data
10:         add contribution of current quadrature point to the element's data structures
11:     end for
12:     if composing material is fluid then
13:         assemble convection
14:     end if
15: end if
```

Some further comments are due in order to further explain the steps presented in Algorithm 4:

- A quadrature point is one of the discrete points where the value of the function is numerically approximated. The contribution of all quadrature points (number depends on the type of element and quadrature rule) in the finite element allows the stiffness matrix to be built;

- Line 5 mentions the use of a "special" assembly routine for integration elements. This "special" assembly routine is not the only one already created in the code, as other advanced projects also require different assembly steps. Some other examples of these functions are the triangular elements integration elements for the crack propagation implementation already mentioned, hierarchical elements and mixed-enhanced elements, among others;

- Describing the steps mentioned in lines 7, 9 and 10 would mean going into detail into numerical integration and isoparametric mapping of elements which means that, for the purpose of this document, they are considered standalone functions (the procedure that is described below on the assembly of integration elements shares some insight on computing assembly data);

- The explanation for line 11 is that the fluid elements' governing equations are different from the solid ones, which means that some additional steps and corrections are needed to model them correctly, which is also done in a standalone function. Because this step is related to the weighted residual formulation of the FEM and implies the choice of yet another numerical scheme, it discussed in further detail in Chapter 4.

The only differences in the assembling algorithm for standard elements and integration elements are that in the latter there is no check for unsupported elements (Algorithm 4's line 5) and the standalone function that computes the assembly data. Algorithm 5 provides both an explanation on the procedure of obtaining the assembly data for these integration elements and also some insight on what is done for standard elements. Just as a clarification, this algorithm would fall under the functionality of computing assembly data, which means that it happens at a specified quadrature point.

Algorithm 5 Compute assembly data for integration elements algorithm

```
1: Input: information for element being assembled
2: Output: elemental stiffness matrix, force vector and all data for global assembly
3:
4: get additional information on the current integration element
5: compute standard shape functions for the integration element
6: select enrichment functions from standard shape functions of the integration element
7: scale enrichment functions (optional)
8: compute standard shape functions for the parent element
9: concatenate parent shape functions and enrichment functions into one array
10: compute vectors and matrices for numerical integration
11: return data
```

Finally, some remarks on the last algorithm (Algorithm 5) of this method's implementation are needed for clarification purposes:

- To compute the parent shape functions and enrichment functions, the additional information that is needed and mentioned in line 4 of the pseudo-code is the corresponding parent and the number of interface nodes;
- The step of selecting the enrichment functions among the standard shape functions for the integration element in line 6 is made a lot simpler due to the decision of having the interface nodes in the last positions of the element's node array;
- The scaling of the enrichment functions in line 7 is optional (as is mentioned before) and by this time, its value has already been computed in the *geometric engine*. Information obtained using the parent element id and the data structure containing information on scaling of the enrichment functions;
- Line 9 of the pseudo-code ends the differences in the process of obtaining the assembly data for integration elements. For these, the shape functions data structures is a concatenation of the parent standard shape functions with the enrichment functions and from this point on, the process is the same as in standard FEMs (except for the post-processing step needed to obtain the continuous solution).

In addition to the assembly routines created for the integration elements, an error function for them has also been created. This error function is not useful to solve problems *per se* but it is needed for the next section for verification purposes. This function basically computes the difference between approximated value and exact value (that needs to be provided if the goal of the problem is an error analysis) at each quadrature point and its implementation is similar to the assembly routines.

3.5 Verification

To have confidence in any computational results obtained using the implementation of a computational methods requires some sort of validation or verification and this implementation of IGFEM is no exception.

Consequently, having described the IGFEM and its implementation, the next step is verification. To do this is to assess the quality of the finite element approximations obtained using *hybrida*, which can be done using sample problems for which the exact solution is known. This involves comparing the approximated solution for the displacement and its derivative obtained for a set of meshes against the exact values.

3.5.1 Error Norms

Doing a somewhat short introduction on the topic error analysis in FEMs by summing the information present in Babuska and Szabo [42], one may point out that there are two main error measurements: the strain energy and the L_2 error norms.

Without getting too specific in the mathematical details of these error metrics, since they are standard for verifying the implementation finite element methods, the L_2 and strain energy error norms are measures of the error found in the solution field and its derivatives, respectively, when compared to the corresponding analytic solution.

Both of these are normalized with the exact solution (which is either the derivative of the solution, in the case of the strain energy error norm or with the actual solution for the L_2 error norm) and their mathematical derivation and description can be found in all introductory books on finite element modeling (some of which have already been mentioned in this document) or in Babuska and Szabo [42].

The tool for the computational error analysis is a convergence plot (error *vs.* finite element size) from which conclusions can be drawn. The data points for this plot are selected from what is computed at each quadrature point by the error functions mentioned in the previous chapter (the quadrature point with the largest error is considered) and their slope is the convergence rate. There are optimal values for these convergence rates depending on the methods being analyzed.

3.5.2 $2D$ Plate: Patch Test

In this subsection, a $2D$ bi-material square shaped plate problem is used to do a convergence study for the implementation of IGFEM. An identical example is also used in Ramos et al. [43] with the goal of proving both a correct implementation and also ideal convergence rates for IGFEM. With this in mind, the objective here is to match the results obtained in this work with the results obtained from this source.

This square shaped plate, for which a schematic representation can be found in Figure 3.9, has a side length of $L = 2\,m$ and is split by a material interface at $x = 0\,m$. As for the physical properties, material 1 has ($E_1 = 10\,Pa$, $\nu_1 = 0$) and material 2 ($E_2 = 1\,Pa$, $\nu_1 = 0$). The boundary conditions applied are also represented in the figure and correspond to a clamped left extremity and a constant traction per

unit length applied at the right extremity. Although it is not visible in the figure, a constant body force is also being applied to the domain of the problem $b_1 = 2\,N/m^2$ (corresponds to a source term in the linear elasticity equation) because otherwise the obtained solution would be equal to the analytic one even for a very coarse mesh of only one element (verified with the developed implementation).

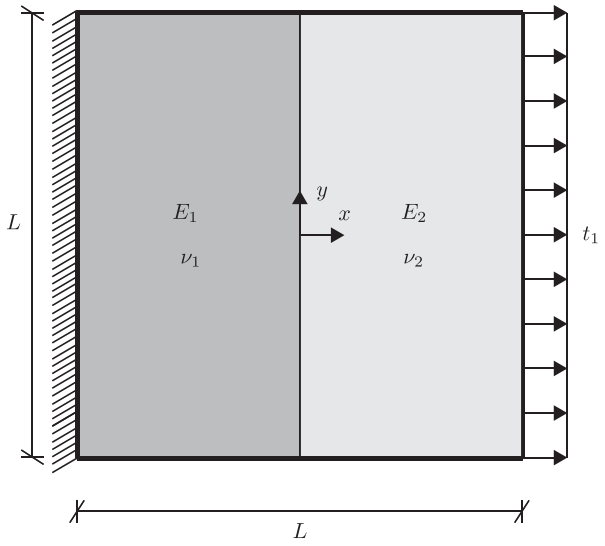


Figure 3.9: $2D$ bi-material plate problem statement for the convergence study.

Since the Poisson's ratio being used for both materials is null (under uniaxial traction and these boundary conditions the deformation only happens in the x direction), the process of obtaining the analytic solutions for the displacement and its derivative (necessary for the convergence analysis) starts from the formulation in Equation (2.10), with the displacement in the x direction being identical to the $1D$ solution and the displacement in the y direction being null.

As is shown for the $1D$ example in Subsection 3.3.2, the next step is to do a first integration and apply the second order boundary condition (which is translated into $EA\frac{du}{dx}|_{x=L} = t_1 = 2\,N/m$), which is taken care of in Equation (3.24). The difference here is that this time the b_1 term in the equation is nonzero (and corresponds to the constant horizontal body force source term being applied to the domain),

$$EA\frac{du}{dx} = -b_1 x + C_1 \Rightarrow t_1 = b_1 L + C_1 \Leftrightarrow C_1 = t_1 + b_1 L \overset{b_1 = t_1}{=} t_1(1+L)\,. \tag{3.24}$$

The next step is another integration and applying the first order boundary condition (which translates into $u|_{x=0} = 0$) considering that, from here on, the numerical values of b_1 and t_1 are the same,

$$EAu = \frac{-t_1}{2}x^2 + C_1 x + C_2 \Rightarrow C_2 = 0\,. \tag{3.25}$$

Having computed all of the integration constants, the mathematical expression to obtain the displacement in a referential centered on the left extremity of the domain (similar to what is used in the $1D$

example) is easily obtained

$$u_x(x) = \frac{1}{EA}\frac{-t_1}{2}x^2 + xt_1(1+L) \Leftrightarrow u_x(x) = \frac{1}{EA}t_1 x(1+L-\frac{x}{2})\,. \tag{3.26}$$

Shifting the referential to the one presented in Figure 3.9 and taking into account a domain composed of two different elastic materials, it is also straightforward to obtain the general expression for the displacement within the square shaped plate for the loading conditions being used

$$\begin{aligned} u_x &= \begin{cases} \frac{(x+1)[2t_1 - L(x-3)]}{2E_1} & \text{if } -L/2 \leq x \leq 0 \\ \frac{E_2(3L+2t_1)+E_1 x[2(t_1+L)-Lx]}{2E_1E_2} & \text{if } 0 < x \leq L/2 \end{cases} \\ u_y &= 0\,(\nu_1 = \nu_2 = 0)\,. \end{aligned} \tag{3.27}$$

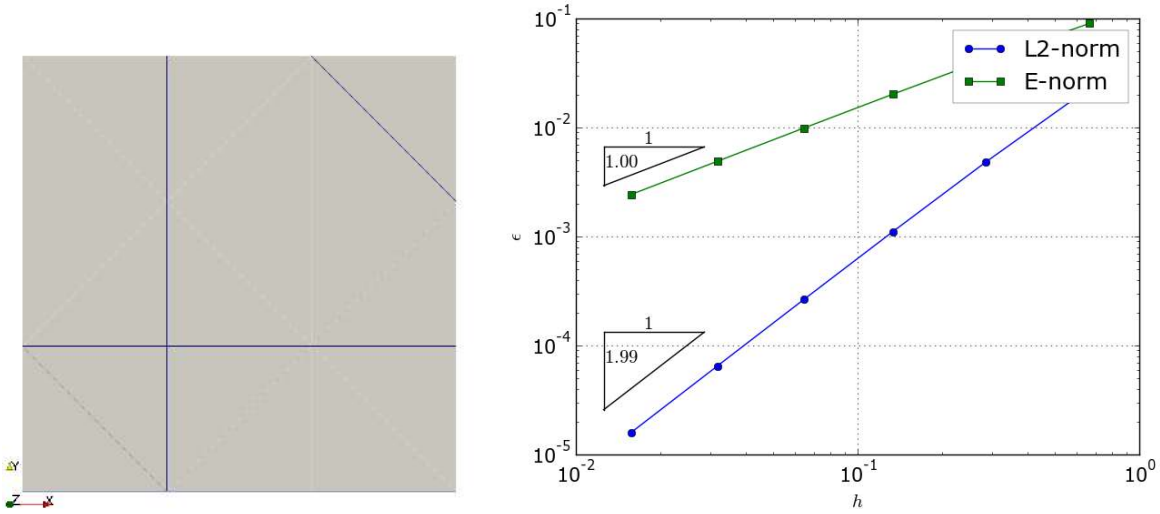
As mentioned at the beginning of this section, the energy error norm is related to the derivative of the solution field, which means that it also necessary to obtain the exact solution for the derivative of the displacement field. Due to the shift in the position of the referential, it is simpler to obtain the derivatives of the displacement by deriving the expression for the displacement field in Equation (3.27) rather than directly from its expression in Equation (3.24),

$$\begin{aligned} \frac{du_x}{dx} &= \begin{cases} \frac{-Lx-t1-L}{E_1} & \text{if } -L/2 \leq x \leq 0 \\ \frac{-Lx-t1-L}{E_2} & \text{if } 0 < x \leq L/2 \end{cases} \\ \frac{du_x}{dy} &= \frac{du_y}{dx} = \frac{du_y}{dy} = 0\,. \end{aligned} \tag{3.28}$$

Having now obtained all the formulas needed to get the analytic solution for the displacement in the plate, it is time to focus on the type of finite element mesh to use in order to obtain the approximated solution field. Although the meshes needed only have the requisite of being nonconforming, which means that the number of horizontal divisions has to be odd (interface in the middle), it is best to create meshes that are as symmetric as possible because, due to the use of the source term, the existence of nodes with different stiffness contributions leads to odd results, with deformation in the y direction even though the Poisson ratio is null and there is no loading in that direction. Taking these aspects into account, the meshes created for this convergence analysis range from three to one hundred and twenty seven side divisions of the domain (the coarser mesh, with three side divisions, is presented in Figure 3.10(a).

With everything that has been mentioned thus far, it is possible to plot a graph for the convergence rates of the error norms mentioned, which is presented in Figure 3.10(b). From this convergence plot come the following observations:

- The obtained convergence rates are not only equal to the ones found in Ramos et al. [43] but also the same as the optimal convergence rates for the standard FEM with a conforming mesh, which



(a) Sample mesh

(b) Convergence plot

Figure 3.10: $2D$ bi-material plate problem.

would not be possible for the nonconforming meshes that have been used with IGFEM;

- Taking the order of magnitude of the error found for the mesh sizes tested, it is possible to conclude (in a qualitative way) that the results are very promising even without looking into the rates of convergence.

Taking all the information in the previous observations, the final conclusion is that the IGFEM implementation in *hybrida* has passed its first test case.

3.5.3 Eshelby Inclusion Problem

In the convergence study problem presented in the previous subsection, ideal convergence rates are shown for a simple $2D$ bi-material square shaped plate. However, due to the simplicity of that example and the not so wide range of geometric cases it can generate for the different meshes, it is relevant to analyze something more complex.

Following once again the procedure used to prove the correct implementation of the method present in Ramos et al. [43], the second convergence test case that has been chosen is based on the classic Eshelby inclusion problem. Using a similar problem statement, the objective is once again to match the results obtained in the implementation for this thesis with the results obtained from this source (optimal convergence rates).

This inclusion problem consists of the circular $2D$ domain with a concentric circular inclusion made out of a different material presented in Figure 3.11. For this convergence analysis, the outer radius chosen is $r_u = 2m$ and the radius of the inclusion is $r_i = 0.4m$. On the other hand, the inclusion is made out of material 1, with ($E_1 = 1Pa, \nu_1 = 0.25$), and the matrix out of material 2, with ($E_2 = 10Pa, \nu_2 = 0.3$).

Regarding the deformation field, plane strain conditions are assumed. The transformation of coordinates from cylindrical (in the figure) to Cartesian and *vice versa* are

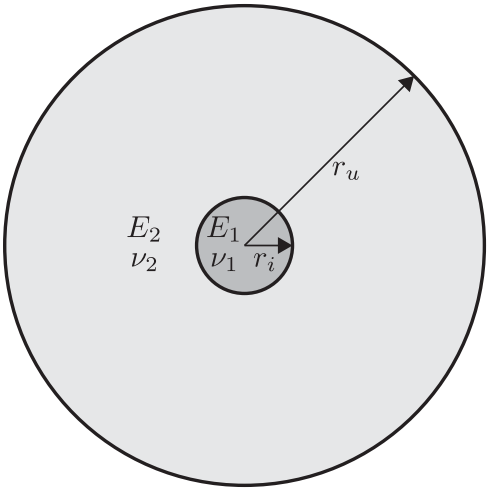


Figure 3.11: $2D$ circular inclusion problem statement.

$$\begin{cases} x = r\,cos(\theta) \\ y = r\,sin(\theta) \end{cases} \tag{3.29}$$

$$\begin{cases} r = \sqrt{x^2 + y^2} \\ \theta = arctan(y/x)\,. \end{cases} \tag{3.30}$$

On the other hand, there is only one BC applied to the domain which is a linear displacement applied along the outer boundary which is (in cylindrical coordinates),

$$\begin{cases} u_r = r_u \\ u_\theta = 0 \end{cases} \Rightarrow \begin{cases} u_x(\theta) = r_u\,cos(\theta) \\ u_y(\theta) = r_u\,sin(\theta)\,, \end{cases} \tag{3.31}$$

where θ is obtained using Equation (3.30). Obtaining the exact solution for the displacement field is not as straightforward as using the bar equation (such as in the previous example). Being a classic solid mechanics examples, the exact solution for the displacement is available and can be obtained in cylindrical coordinates from the same source as the results that are trying to be reproduced (Ramos et al. [43]). To obtain the solution in Cartesian coordinates (as is what is used in this implementation), it is only necessary to replace the instances of r and θ in the cylindrical solution with the expressions in Equation (3.30),

$$\begin{aligned} u_r(r) &= \begin{cases} \left[\left(1 - \frac{r_u^2}{r_i^2}\right)\alpha + \frac{r_u^2}{r_i^2}\right] r & \text{if } 0 \le r \le r_i \\ \left(r - \frac{r_u^2}{r}\right)\alpha + \frac{r_u^2}{r} & \text{if } r_i < r \le r_u \end{cases} \Rightarrow \begin{cases} u_x(x,y) = \frac{r_u^2 + \alpha(r_i - r_u)(r_i + r_u)}{r_i^2} x & \text{if } 0 \le r \le r_i \\ u_y(x,y) = \frac{r_u^2 + \alpha(r_i - r_u)(r_i + r_u)}{r_i^2} y & \text{if } 0 \le r \le r_i \\ u_x(x,y) = x\left(\alpha - \frac{(\alpha-1)r_u^2}{x^2+y^2}\right) & \text{if } r_i < r \le r_u \\ u_y(x,y) = y\left(\alpha - \frac{(\alpha-1)r_u^2}{x^2+y^2}\right) & \text{if } r_i < r \le r_u\,, \end{cases} \\ u_\theta(r) &= 0 \end{aligned} \tag{3.32}$$

where α is computed using

$$\alpha = \frac{(\lambda_1 + \mu_1 + \mu_2) r_u^2}{(\lambda_2 + \mu_2) r_i^2 + (\lambda_1 + \mu_1)(r_u^2 - r_i^2) + \mu_2 r u^2} \tag{3.33}$$

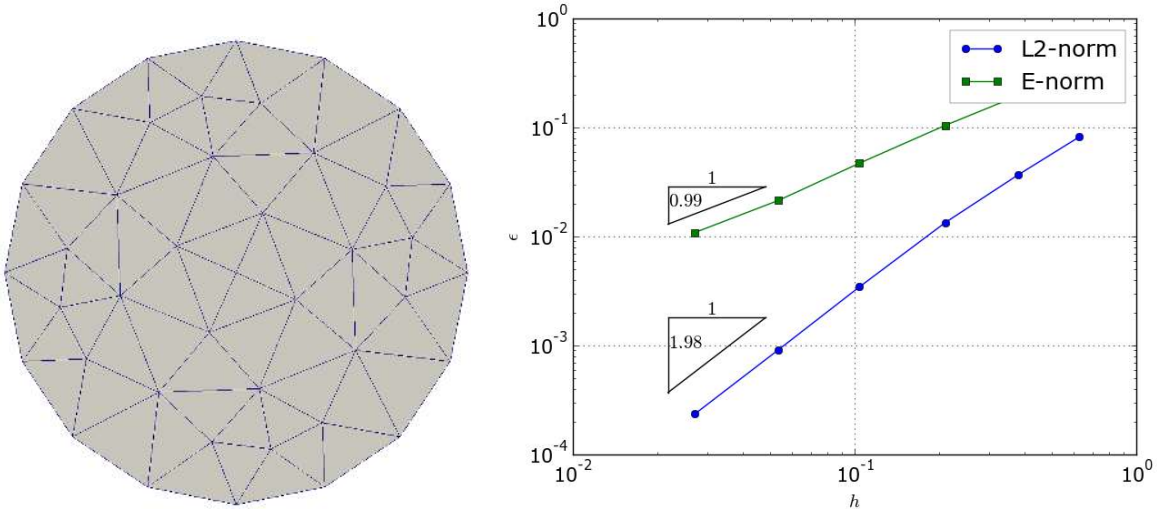
and is a function of the geometry and Lamé constants for both the matrix and the inclusion (which can be computed using Equations (3.34) and (3.35)). The displacement solution in Cartesian coordinates is composed of four equations, two equations for each of the considered regions due to the two directions being considered.

$$\mu_i = \frac{E_i}{2(1+\nu_i)} \tag{3.34}$$

$$\lambda_i = \frac{2\nu_i\mu_i}{1-2\nu_i} \tag{3.35}$$

Having the analytic solution for the displacement in the circular plate, the derivatives of said solution are still needed in order to compute the energy norm of the error for the implementation. Because the displacement solution in Cartesian coordinates is composed of four equations, obtaining the derivatives $\frac{du_x}{dx}$, $\frac{du_x}{dy}$, $\frac{du_y}{dx}$ and $\frac{du_y}{dy}$ generates eight equations which, despite being straightforward to obtain by hand, would be a possible source of mistakes. To eliminate this possible source of mistakes, these equations have been obtained using the symbolic calculus tool *Mathematica* and can be found in Appendix A.

Unlike in the previous example, it is not possible to create a structured conforming mesh for this domain (mesh that conforms to the outer circle). However, doing a convergence analysis for an unstructured mesh is also a relevant test. Taking this into account, the meshes that are used for this study are similar to the one in Figure 3.12(a), this being the coarser. Using this and the finer meshes made it possible to obtain the convergence plot of Figure 3.12(b).



(a) Sample mesh

(b) Convergence plot

Figure 3.12: $2D$ circular inclusion.

There is not much to add as in comments to the convergence plot, since the result is identical to what has been obtained previous one, with optimal convergence rates that are the same as in the inspiring source.

The obtained optimal convergence rates for both this example and the one in Subsection 3.5.2 prove a correct implementation of the method, which means that the burden of creating a mesh that conforms to a possibly intricate geometry or of creating a new mesh for each optimization step can be avoided.

3.6 Other Topics

3.6.1 Enrichment Functions Scaling

Now, despite the feature of not needing good quality (reasonable aspect ratio) finite elements as children elements, initial formulations of IGFEM had a problem in stiffness matrix conditioning (the condition number of a matrix determines how much the output changes in regard to a small change in the input argument).

Looking further into this in more recent sources on this kind of method (like Soghrati and Barrera [37]) reveals that the numerical difficulties appear in situations where an interface node is too close to an original mesh node and due to the fact that the enrichment functions are Lagrangian shape functions evaluated at the integration element. The answer to this problem is scaling the method's enrichment functions, which is shown in Figure 3.13. This operation does not affect the solution in any way, as it is accounted for in both creating the enrichment functions and in the post-processing step.

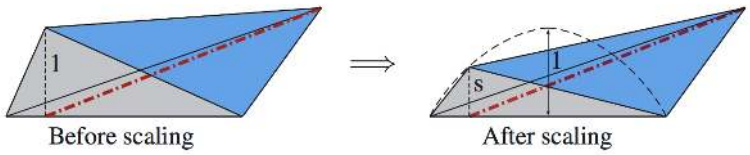


Figure 3.13: Enrichment function scaling from Soghrati et al. [10].

In order to try to fix this issue, a simple test case has been devised that is similar to what has been used in the square plate verification problem but this time using only one mesh and a tilted interface (fixed in the centroid of the plate). Varying the tilt angle of the interface made it possible to obtain the stiffness matrix of the entire model for different distances between the interface nodes and original mesh nodes. This problem can be used to test the various scaling functions to verify what happens to the condition number.

Despite having tested a few more scaling functions, the relevant results to present are only obtained for three of them. They are given by

$$s_j = 1\,, \tag{3.36}$$

$$s_j = \frac{min(d_1,\, d_2)}{d_1 + d_2}\,, \tag{3.37}$$

$$s_j = \frac{min(d_1,\, d_2)}{max(d_1,\, d_2)}\,, \tag{3.38}$$

where d_1 and d_2 are the distances from the interface to the previous and following original mesh nodes in the element connectivity data structure, respectively (recall that the nodes in these data structures are always ordered in a CCW manner). The three scaling functions in Equations (3.36) through (3.38) correspond to a unitary scaling factor (standard), to verify that the method presents in fact an ill-conditioning

problem, and the scaling functions from the Soghrati et al. [10] and Soghrati et al. [36], which correspond to the initial IGFEM formulation (original scaling) and an hierarchic adaptive formulation (modified scaling) of the same method (more advanced).

In order to visualize the results of this study, Figure 3.14 represents the condition number of the global stiffness matrix *vs.* the minimum distance between interface nodes and standard mesh nodes of the tilted interface square plate test case for the different scaling functions that are presented in Equations (3.36) through (3.38).

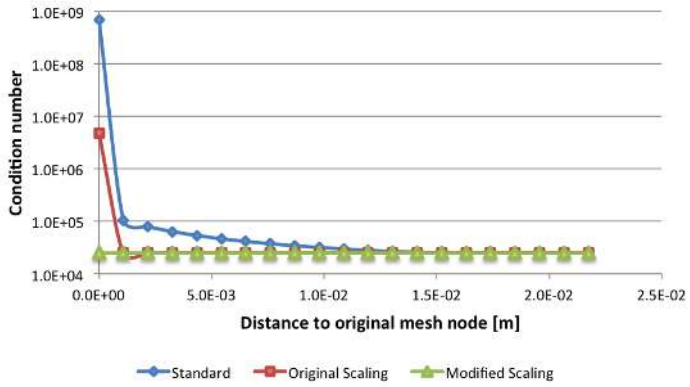


Figure 3.14: Condition number *vs.* minimum distance between interface nodes and original mesh nodes.

Regarding the results obtained for the condition number *vs.* minimum distance between interface nodes and original mesh nodes in Figure 3.14, some observations can be pointed out:

- The unitary scaling factor (does not do any scaling, it just serves as reference), represented by the blue line and diamond shaped markers, clearly points out that the reported ill-conditioning problem exists when the minimum distance between interface nodes and original mesh nodes decreases, as the condition number explodes for the last data point of the curve;
- The scaling factor introduced in the original formulation, represented by the red line and square shaped markers, alleviates this situation but still does not present ideal results, as the condition number remains approximately constant until the last data point of the curve, where it still diverges, presenting a value smaller by two orders of magnitude when compared to the unitary scaling;
- Finally, the scaling factor introduced in adaptive IGFEM, represented by the green line and triangular markers, appears to solve the problem, with a conditioning number approximately constant for any minimum distance between interface nodes and original mesh.

Given that the scaling factor introduced in adaptive IGFEM presents the best results as far as the conditioning of the stiffness matrix goes, it is used for this implementation of IGFEM. Together with the innovative algorithms employed for the *Geometric engine*, this implementation of IGFEM in *hybrida* is a development on what was originally presented.

Chapter 4

Weighted Residual Formulation - SUPG

As briefly described in the previous chapter and according to Brooks and Hughes [44], the FEM makes use of a discretization method and a weighted residual formulation and yields an approximated solution to the initial boundary value problem. The choice of discretization technique, which is featured in the IGFEM, has already been described and justified, which means it is now necessary to discuss the choice of the weighted residual formulation.

4.1 Requirements

The most common weighted residual formulation chosen and used for the FEM (even for IGFEM's unconventional discretization technique) is the Galerkin method (refer to Reddy [19] for a thorough review on the topic), where weighting functions and interpolation functions are from the same class. Producing symmetric stiffness matrices, the method is adequate when modeling most structures or conduction dominated heat transfer processes because it has the "best approximation" property. Without going into too much detail, this means that the error between the obtained solution and the analytic one goes to zero with a certain norm.

However, it is shown both in Brooks and Hughes [44], Donea and Huerta [45] and probably all sources that approach a FEM perspective to flow problems, that the Galerkin weighted residual formulation is not adequate to convection dominated problems (like in the cooling channels of the materials being studied) because this "best approximation" property is lost, with the matrix associated to the convective term being non-symmetric. This non-symmetry results in spurious oscillations in the solution field, also called "wiggles" in the world of computational fluid dynamics (CFD). The appearance of these "wiggles" for the Galerkin method is shown in detail Section 4.4 for both $1D$ and $2D$ cases but is also exemplified in Figure 4.1, which shows the analytic and Galerkin temperature distribution solutions for a convection dominated heat transfer problem in a $1D$ bar with temperature fixed at both ends.

Knowing that the Galerkin method is ineffective dealing with convection dominated heat transfer

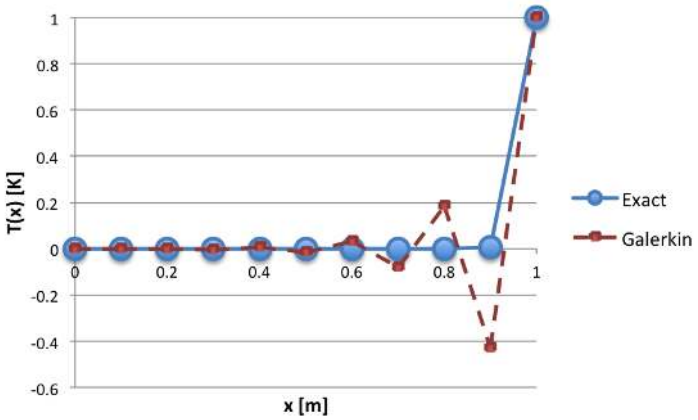


Figure 4.1: Temperature distribution solution in $1D$ bar to point out spurious oscillations.

problems, it is necessary to implement a weighted residual formulation that is not.

4.2 Streamline Upwind/Petrov-Galerkin method

The chosen alternative has first been presented in Brooks and Hughes [44] and is the Streamline Upwind/Petrov-Galerkin (SUPG) method, which is a modified weighted residual formulation that stabilizes the solution for problems that have highly convective effects present. If implemented correctly, this method must produce stable and accurate results.

Keeping to the essentials needed to explain the concept of SUPG, it is shown in literature (Brooks and Hughes [44]) that the standard Galerkin weighted residual formulation leads to a central difference scheme, which is known from CFD to introduce a truncation error in the form of diffusion (or conduction) operator. In other words, it actually provides a solution with wrong diffusivity (less than the actual). It is also shown that this negative numerical diffusion is actually the cause of numerical difficulties found in highly convective transport problems.

The main idea behind the SUPG method comes from earlier stabilization techniques that simply modified the weight functions used to obtain the solution in the form of added diffusion to counterbalance the negative numerical diffusion introduced by the Galerkin approximation. However, just introducing artificial diffusion can produce decreased accuracy in the form of overly diffuse solutions. According to Donea and Huerta [45], this interpretation has been source of extensive criticism.

From Soghrati [46], the amplitude of the fictitious oscillations or “wiggles” is a function of the directional grid Péclet number, the dimensionless number evaluated at each finite element in the direction of the flow that is relevant in the study of transport phenomena in a continuum. The keyword in the last sentence that dictates that the SUPG method is not merely the introduction of artificial diffusion is “directional”: in fact, the SUPG method also modifies the weight function to add diffusion (put more weight on the upstream nodes) but this diffusion is added only in the flow direction, which alleviates the phenomenon that is target to criticism (excessive crosswind diffusion). The stabilization introduced by SUPG is also a function of this directional grid Péclet number (presented in Subsection 4.2.2).

4.2.1 Problem Statement for the Conjugate Heat Transfer Problem

Differently from what is done for the last chapter where the IGFEM is presented, the SUPG method requires a generic conjugate heat transfer problem statement to be presented in order to explain its formulation. The information that is necessary for this explanation are than the strong and weak forms of an heat transfer process in the generic vascular domain represented in Figure 4.2.

To obtain the temperature distribution field in a vascular actively-cooled material, it is necessary to solve the convection-diffusion equations (heat equation) on a $2D$ domain $\Omega \subset \mathbb{R}^2$, pictured in Figure 4.2, that has two non-intersecting regions, the solid phase (Ω_s) and the fluid phase (Ω_f), which is where the coolant flows in the case of vascular materials. It is also possible to find the designations Γ_T, Γ_q and Γ_h (not used in this work), that are the boundaries to which each type of BC is applied, the first being Dirichlet, the second Neumann and third Robin boundaries.

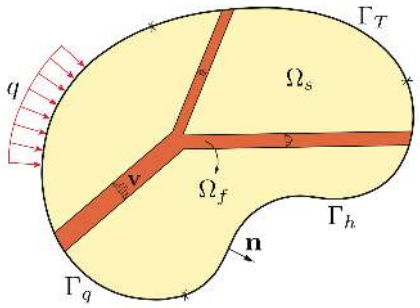


Figure 4.2: Sample 2D domain for conjugate heat transfer problem from Bergman et al. [6].

The strong form of the convection-diffusion equation for the domain and boundaries shown is:

$$\begin{aligned} &\textit{find the temperature field } T : \Omega \Rightarrow \mathbb{R} \textit{ such that} \\ -\nabla \cdot (\mathbf{k}\nabla T) &= f \text{ in } \Omega_s \\ -\nabla \cdot (\mathbf{k}\nabla T) + \rho c_p \mathbf{v} \cdot \nabla T &= f \text{ in } \Omega_f \\ T &= \overline{T} \text{ on } \Gamma_T \\ \mathbf{k}\nabla \cdot \mathbf{n} &= q \text{ on } \Gamma_q \,. \end{aligned} \tag{4.1}$$

Because the SUPG method is a modification of the weight functions used to obtain the FEM approximated solution, it is necessary to obtain the weak form of the problem to appreciate where the stabilization terms affect the governing equations,

$$\underbrace{\int_\Omega \nabla w^h \cdot \mathbf{k}\nabla u^h d\Omega + \int_{\Omega_f} w^h \rho c_p \mathbf{v} \cdot \nabla u^h d\Omega}_{\text{Standard convection}} = \int_{\Gamma_q} w^h \overline{q} d\Gamma \,, \tag{4.2}$$

where w^h are the trial weighting functions and u^h is the approximated solution. It is worth mentioning that before the work that is being described in this document, *hybdrida* did not have any formulation for advection or fluid flows (stabilized or otherwise), so the only term on the left side in Equation (4.2) implemented thus far was the first one. In the equation, the terms to the left of the equal sign correspond

to the diffusion and standard advection (without stabilization) phenomena, respectively. The term to the right of the equal sign corresponds to the heat source.

The stabilization that is introduced with the SUPG method is obtained with an additional term in the equation for the problem's weak form, which is presented in Subsection 4.2.2.

4.2.2 Formulation

As mentioned in the beginning of this section, both the amplitude of the "wiggles" and the stabilization parameter depend on the grid Péclet number. This dimensionless number is defined as

$$Pe_k = \frac{\text{rate of advection}}{\text{rate of conduction}} = \frac{|\mathbf{v_k}|h_k}{2k_k}\,, \tag{4.3}$$

where $|\mathbf{v_k}|$ is the norm of the vector quantity flow velocity, h_k is the element length in the flow direction and k_k is the thermal conductivity of the material in the flow direction.

In $2D$ domains, the thermal conductivity of a material is actually a tensor, which means that it can have different values for different directions. Taking this into consideration, the thermal conductivity in the direction of the flow can be computed as

$$k_k = \frac{||\mathbf{k}\cdot\mathbf{v_k}||}{||\mathbf{v_k}||}\,. \tag{4.4}$$

Finally, all of the necessary values are computed in order to evaluate the stabilization parameter. The formula for this stabilization factor is not element type independent and can be computed, for $2D$ elements (Fries and Matthies [47]), using

$$\tau = \frac{h_k}{2|\mathbf{v_k}|}\left(coth(Pe_k) - \frac{1}{Pe_k}\right)\,. \tag{4.5}$$

It is crucial to note (for implementation purposes) that, because the stabilization is dependent on the flow velocity, it may not the same for all elements composed of a fluid material or even constant inside the same element if the velocity profile is not constant (in reality, the flow velocity in the channels radially varies). This means that it has to be evaluated at each quadrature point.

As mentioned in the previous subsection, the SUPG method employs a modification of the weight functions used to obtain the FEM approximated solution, which means that it can be expressed as an additional stabilization term in the weak form of the problem as in

$$\underbrace{\int_\Omega \nabla w^h\cdot\mathbf{k}\nabla u^h d\Omega + \int_{\Omega_f} w^h\rho c_p\mathbf{v}\cdot\nabla u^h d\Omega}_{\text{Standard convection}} + \underbrace{\int_{\Omega_f}\tau(\mathbf{v}\cdot\nabla w^h)(\mathbf{v}\cdot\nabla u^h)d\Omega}_{\text{Stabilization term}} = \int_{\Gamma_q} w^h\overline{q}d\Gamma\,. \tag{4.6}$$

Concluding this section, the original *hybrida* code lacks the implementation of the second and third terms in Equation (4.6) in order to be able to effectively model the vascular cooled materials that are subject for this thesis.

4.3 Implementation

Having described the concept and formulation of the SUPG method, the next step is to present how it is implemented in the context of this project and in *hybrida*. As is mentioned above, there was no implementation of any convection or fluid flow formulation (stabilized or otherwise) in the code until this project. Having this opportunity to implement a phenomenon without having any previous structure behind means that some decisions are in order so that the functionality can be integrated in other projects with adequate level of encapsulation.

As mentioned and shown in the previous sections, the method consists of adding a stabilization term that alleviates the "wiggles", which means that its implementation appears at the end of the assembly routines (for both integration and standard elements) if the element being assembled is composed of a fluid material (refer to Algorithm 4 in Chapter 3).

The solution to detect the presence of a fluid material is the first decision that needs to be made for the implementation of convection. The initial thought was to change the element types if the element being analyzed was in the fluid phase. This solution proved to be to cumbersome and disruptive of the code's structure, as it demanded new element types (identical to standard elements but with additional information). The simplest solution that requires the minimum amount of modifications in other modules of the code is creating a new type of material, which is attributed to elements whose nodes have negative or negative and null level set values (refer to Chapter 3's Algorithms 1 and 3 to review this process), which is the same as saying that they are located in the fluid phase of the domain. This new type of material has the property of flow velocity, which is a function of the $2D$ spatial coordinates x and y.

At the end of both assembly routines that are presented in Chapter 3 (standard elements and integration elements) but still for each quadrature point is where the material type check takes place. If the material type created for convection is detected, the element enters a function that has the goal of adding the contribution of convection to the FEM model. For this process, it is irrelevant if the element is a standard or an integration element, which means that there is only one function to add the contribution of convection.

Despite a formulation that is not as simple as what has been found for IGFEM, a bit simplified by using Donea and Huerta [45], which provides some tips on the simpler $1D$ formulation for numerical computation, the algorithm for adding the contribution of convection to the assembly is straightforward, as it gets most of the matrices for numerical integration computed in the standard assembly function. In order to simplify the explanation of the procedure and point out some decisions, it is presented in Algorithm 6.

Fully understanding Algorithm 6 requires some observations in order to justify the decisions and clarify the necessary steps:

- The variables listed in line 4 are obtained using Equations (4.3) to (4.5) in Section 4.2;
- For implementing the contribution of convection using the standard Galerkin formulation, some additional matrices are needed (mentioned in line 5 of the pseudo-code). These matrices are computed in order to implement the second term of Equation (4.6);

Algorithm 6 Assemble convection algorithm

1: **Input:** element and assembly information data structures
2: **Output:** updated assembly data
3:
4: compute numerical values needed for convection formulations (v_k, h_k, k_k, Pe_k and τ)
5: compute matrices for Galerkin formulation of convection
6: choose convection formulation (user input)
7: **if** formulation is SUPG **then**
8: compute matrix for SUPG formulation of convection
9: **end if**
10: add convection contribution to assembly data (stiffness matrix, force vector, ...)

- Line 6 mentions choosing the formulation to employ. Despite being mostly about implementation verification, one of the goals of Section 4.4 is to apreciate the appearance of "wiggles" in the Galerkin solution field when problems become convection dominated, which means that it is useful to be able to simply alternate between the two formulations. This feature can also be an advantage for future projects;

- Only if the formulation chosen is SUPG does the matrix needed to implement the stabilization term in Equation (4.6) get computed in order to save computational resources. This matrix is computed based on the stabilization factor and flow velocity vector.

4.4 Verification

Again to gain some confidence in the results obtained using the code developed for this work requires verification of the implementation. In this section, two test cases are presented to accomplish this important task.

4.4.1 $1D$ Convection Approximation

The first test case consists of a simple $1D$ fluid bar of length $L = 1\,m$ made out of a material with a thermal conductivity of $k = 10^{-6}\,W/m \cdot K$ and a flow velocity of $\vec{v}$ with the Dirichlet BCs in Equation (4.7), represented in Figure 4.3. This test case is also useful to appreciate the influence of the flow velocity (which is related the Péclet number) in the appearance of "wiggles".

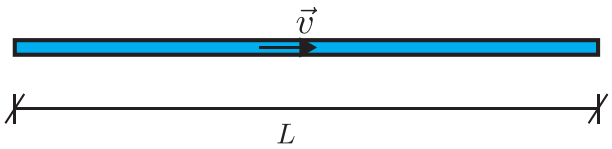


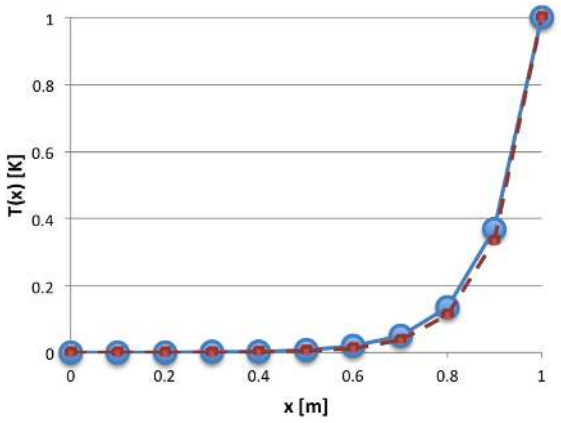
Figure 4.3: $1D$ fluid bar.

$$\begin{cases} T = 0K & @\, x = 0\,m \\ T = 1K & @\, x = 1\,m \end{cases}. \tag{4.7}$$

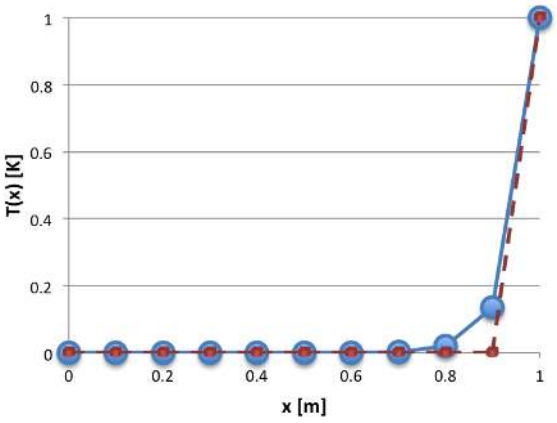
Despite this advantage to study the influence of the flow velocity in the appearance of “wiggles”, the main goal here is to do verification. Consequently, this example has been chosen because there is a simple analytic solution that can be used for this purpose (from Bergman et al. [6]). The governing equation for the temperature distribution in a bar, that derives from the convection-diffusion equation, is

$$T(x) = \frac{1 - e^{Pe\, x/L}}{1 - e^{Pe}}\,. \tag{4.8}$$

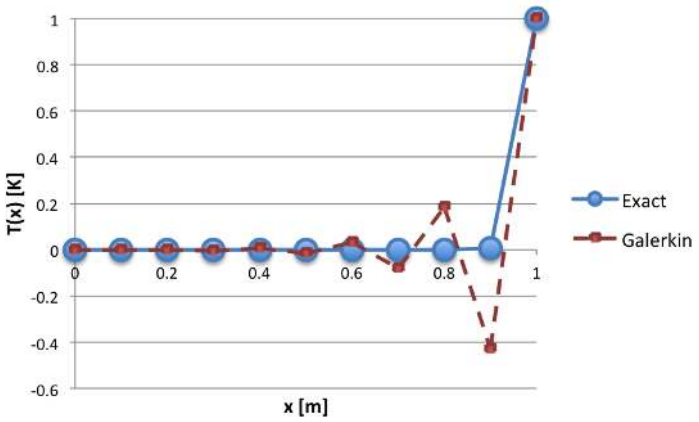
Because this is a $1D$ problem and the implementation of the SUPG assumes $2D$ conditions, the problem has been solved as an extremely sleek rectangular domain of the same length as the bar. Before verifying the results using the SUPG method, it is interesting to analyze the result using the standard Galerkin method. With this in mind, the temperature distribution for three Péclet numbers is presented in Figure 4.4 using a structured finite element mesh with ten rectangular elements and the analytic solution is sampled at each of the nodes in that mesh to provide better grounds for a comparison.



(a) $Pe = 10$, $Pe_k = 0.5$



(b) $Pe = 20$, $Pe_k = 1$



(c) $Pe = 50$, $Pe_k = 2.5$

Figure 4.4: Analytic and Galerkin temperature distribution solution in $1D$ bar for multiple Péclet numbers.

The limitations of the standard Galerkin method when modeling convection dominated flows that are mentioned in Section 4.1 become evident in the plots that are presented in Figure 4.4. For the first two cases ($Pe = 10$ and $Pe = 20$) where order of magnitude of diffusive phenomena is not negligible,

the observation is that the approximated solution appears to be under-diffuse when compared to the analytic ones (more noticeable in the second case). Increasing the Péclet number in the third example shows the appearance of "wiggles" with an amplitude of the same order of magnitude as the values of the solution, which renders the approximated results useless, confirming that looking into an alternative for the Galerkin formulation is necessary in this case. The results can be compared against what has been found in Brooks and Hughes [44], reproduced in Figure 4.5, and the conclusion is that they look identical (this comparison should be made taking into account that the temperature distribution is also a function the grid Péclet number, α in the source article, and that the Galerkin formulation leads to a central difference scheme).

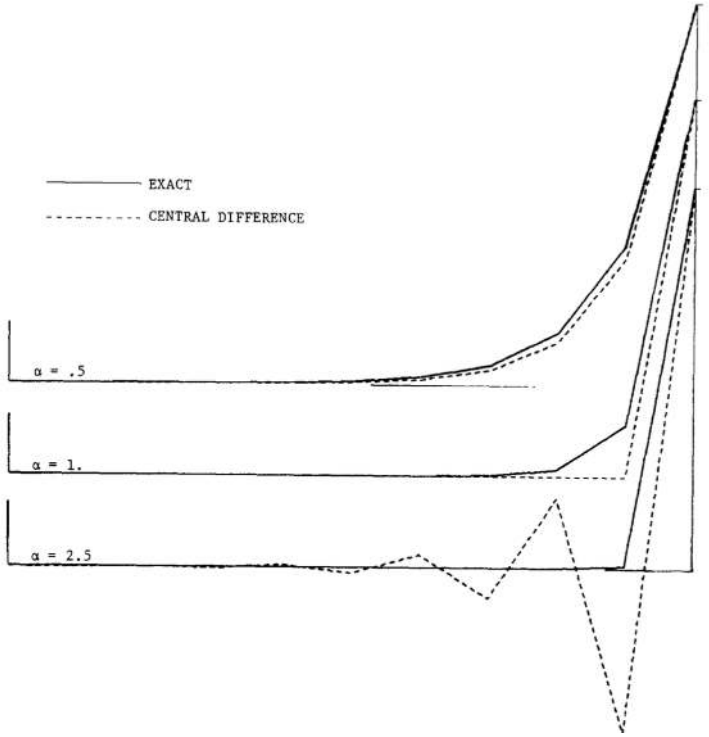


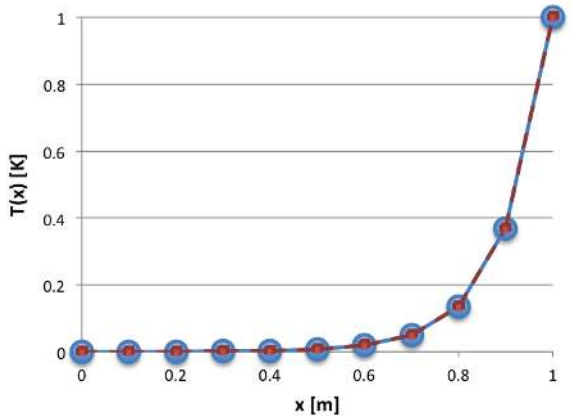
Figure 4.5: Exact and central difference solution for convection in $1D$ bar from Brooks and Hughes [44].

Applying the formulation of the SUPG method above gives way to obtain a different set of plots for the temperature distribution. Figure 4.6 presents these plots, where the conclusion is that the formulation behind them completely solves the appearance of under-diffuse solutions and "wiggles" in the temperature field, as the approximated solutions match the analytic ones to machine accuracy.

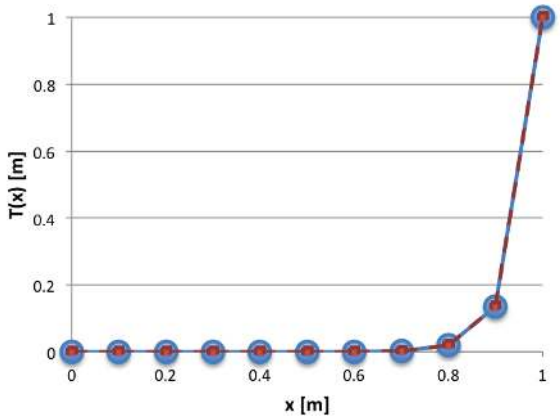
Despite the promising results present in Figure 4.6 regarding the effectiveness of the SUPG method obtaining the temperature field in a bar, this example is somewhat too simple to proof a correct implementation on its own.

4.4.2 Convection Skew to the Mesh

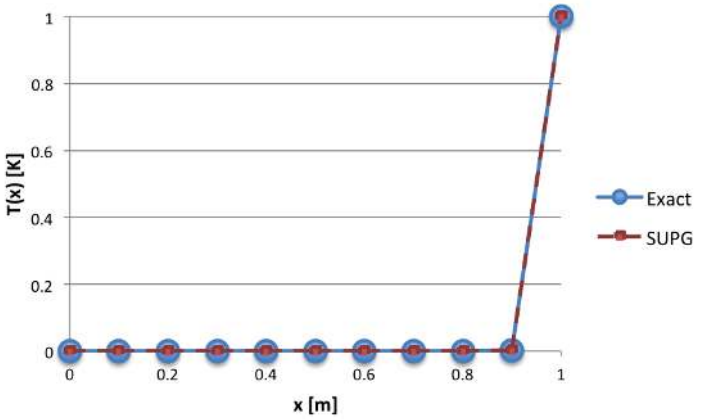
Continuing the work started in the previous subsection, this example is a square plate (inspired by an identical problem from Brooks and Hughes [44]) to add some complexity to the verification process with a $2D$ domain. The geometry and BCs that characterize the problem can be found in Figure 4.7, with a



(a) $Pe = 10$, $Pe_k = 0.5$



(b) $Pe = 20$, $Pe_k = 1$



(c) $Pe = 50$, $Pe_k = 2.5$

Figure 4.6: Analytic and SUPG temperature distribution solution in a bar.

unitary side length ($L = 1\ m$) and flow velocity ($|v| = 1\ m/s$ and direction θ). The problem is convection dominated with a negligible thermal conductivity of $k = 10^{-6}\ W/m \cdot K$, which results a Péclet number of $Pe = 10^6$.

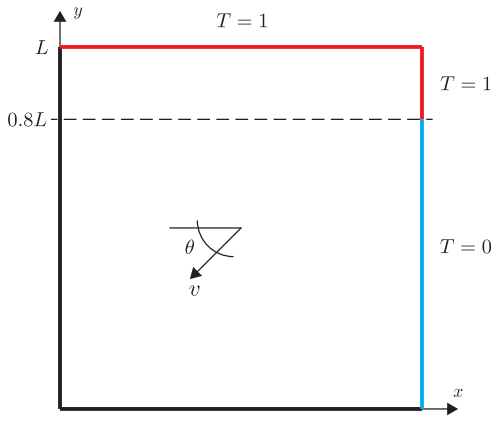
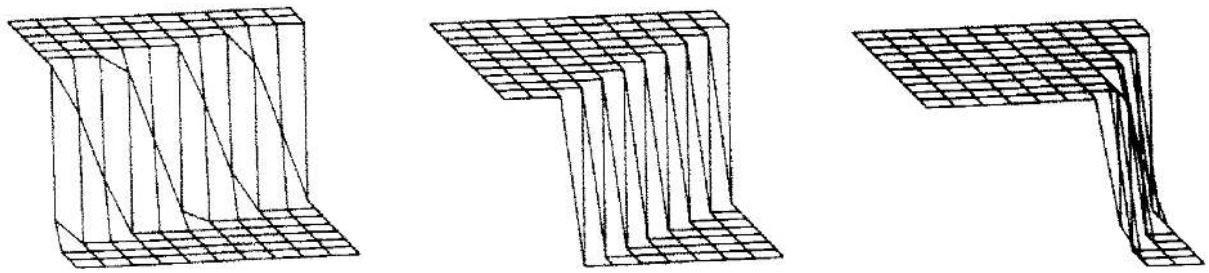


Figure 4.7: Convection skew to the mesh problem statement.

The mesh that has been created to model this problem is identical to the mesh that in the source of this example and consists of a structured uniform mesh of square elements (each side is composed

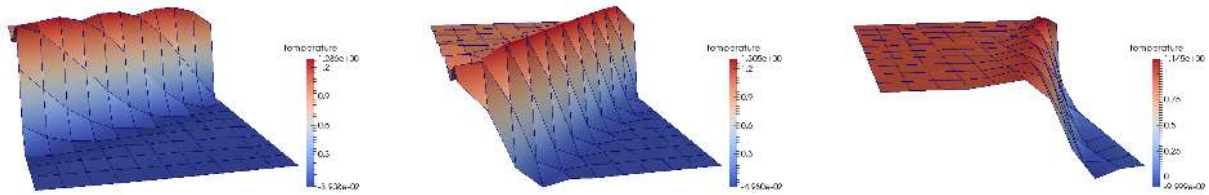
of ten square elements). Despite presenting a representation of the analytic temperature field, that has been reproduced in Figure 4.4, the source provides neither an explanation on how this solution is obtained nor its numerical values, which means this analysis has only a qualitative component.



(a) $\theta = 22.5^{o}$ (b) $\theta = 45^{o}$ (c) $\theta = 67.5^{o}$

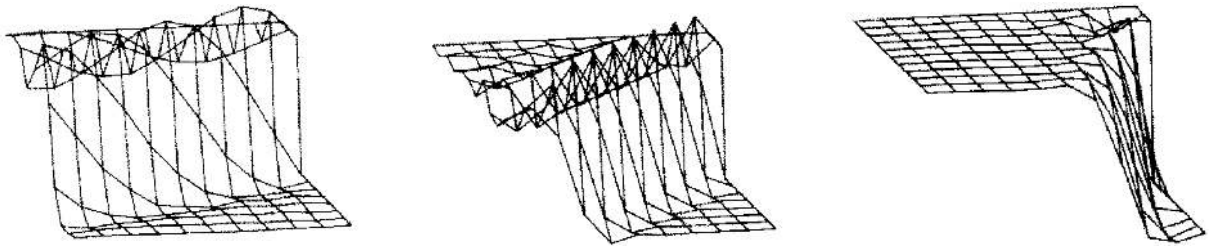
Figure 4.8: Analytic temperature distribution for skew convection from Brooks and Hughes [44].

The procedure for verification here is the same to what has been done in the previous section, starting with the comparison of the Galerkin solutions from the developed implementation and the source. Analyzing Figures 4.9 and 4.10 qualitatively (using the mesh outline as reference), it is possible to conclude that the results of each source are identical for every skew angle represented (refer to Figure 4.7). On the other hand, it is also observable that the amplitude of the “wiggles” is also related to the skew angle (flow velocity is the same for all plots).



(a) $\theta = 22.5^{o}$ (b) $\theta = 45^{o}$ (c) $\theta = 67.5^{o}$

Figure 4.9: Galerkin temperature distribution for skew convection from *hybrida*.



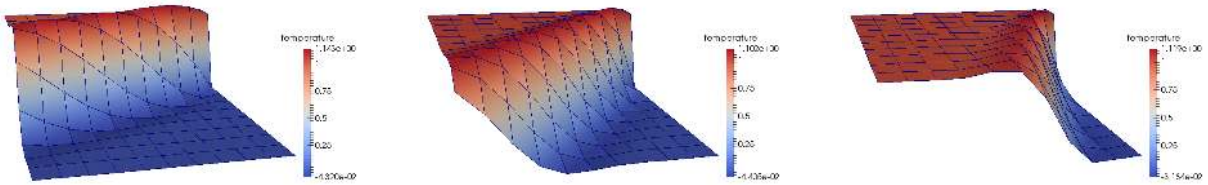
(a) $\theta = 22.5^{o}$ (b) $\theta = 45^{o}$ (c) $\theta = 67.5^{o}$

Figure 4.10: Galerkin temperature distribution for skew convection from Brooks and Hughes [44].

Moving on to the results using the SUPG method that are obtained with the developed implementation, represented in Figure 4.11, some observations are in order:

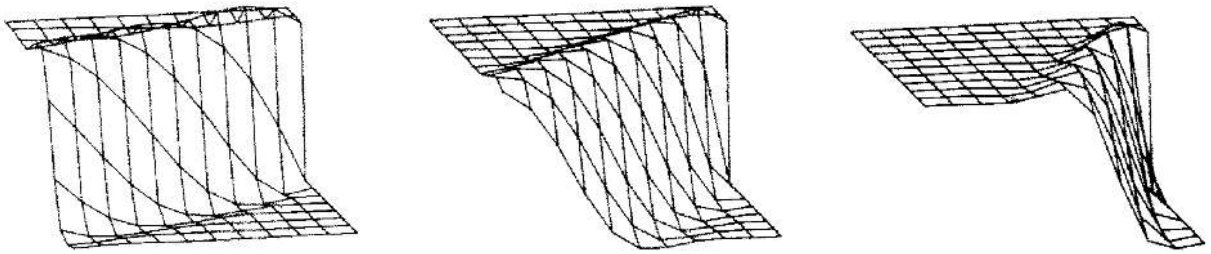
- As is the premise for this method, the results obtained for all skew angles are much smoother and the resulting “wiggle” amplitude is almost nonexistent;

- Using the mesh outline as a reference once again, it is possible to conclude that the results obtained in this implementation agree with the temperature field solution in the source of this example, reproduced in Figure 4.12.



(a) $\theta = 22.5^{o}$ (b) $\theta = 45^{o}$ (c) $\theta = 67.5^{o}$

Figure 4.11: SUPG temperature distribution for skew convection from *hybrida*.



(a) $\theta = 22.5^{o}$ (b) $\theta = 45^{o}$ (c) $\theta = 67.5^{o}$

Figure 4.12: SUPG temperature distribution for skew convection from Brooks and Hughes [44].

Combining the somewhat qualitative conclusions found here with the indisputable results obtained for the $1D$ bar (where the approximated solution matches the analytic ones to machine accuracy), it is possible to claim with confidence that, from what is observed with these examples, the implementation of the method is correct. Consequently, the path for this work can continue on to combine the advantages of both numerical tools discussed in this chapter and in Chapter 3 to tackle representative problems involving actively-cooled vascular materials.

Chapter 5

Test Cases Modeling Results

This chapter is where the two numerical methods discussed thus far are combined, showcasing the power of this union by modeling two test cases involving actively-cooled vascular material blocks with BCs and geometry that mimic a CPU cooler and a car engine cylinder. The CPU cooler test case (its geometry, BCs, baseline solution) and also an IGFEM efficiency study are presented and discussed in Section 5.1 while Section 5.2 discusses the car engine cylinder test case.

5.1 CPU Cooler Test Case

As already mentioned in other chapters of this document, the project that constitutes this thesis is being developed in an academic environment. Due to this and also due to the fact that there was no specific optimization goal, the problem (geometry subject to BCs) to optimize has to be created with the goal of showcasing the power of combining the numerical tools that have been implemented. Despite this, the objective here is not to select a completely random geometry and BCs where the result of the optimization process cannot be fully appreciated, but instead to get inspiration from a real life problem. Being inspired by a real life problem, the results of the optimization process can be analyzed in an easier way, since the order magnitude of the usual temperatures is empirically known.

5.1.1 Problem Description

Before presenting the initial geometry, BCs and remaining data, it is relevant to recall the problem that is being solved using the IGFEM and SUPG. The problem at hand is a conjugate heat transfer problem with the generic information that is presented in Subsections 4.2.1 and 4.2.2, namely a domain schematic in Figure 4.2, the strong form in Equation (4.1) and the weak form in Equation (4.6).

Moving on to the active cooling vascular problem of the CPU cooler that was briefly mentioned before, the first thing to note is that both the geometry and data are inspired in the real values that can be found in multiple references for the subject, for example Vogel [48] or Khonsue [49].

The CPU cooler's initial geometry consists of the $2D$ rectangular domain intersected by an horizontal straight cooling channel that is represented in Figure 5.1, where the effect of the IGFEM is clearly visible

in the zoomed view of a sample mesh also presented in the figure. The matrix (solid phase) of this cooler is made out of copper ($k = 401\ W/m \cdot K$, $c_p = 0.39\ J/kg \cdot K$ and $\rho = 8960\ kg/m^3$) and the circulating cooling liquid (fluid phase) is water ($k = 0.6\ W/m \cdot K$, $c_p = 4.182\ J/kg \cdot K$ and $\rho = 998.3\ kg/m^3$).

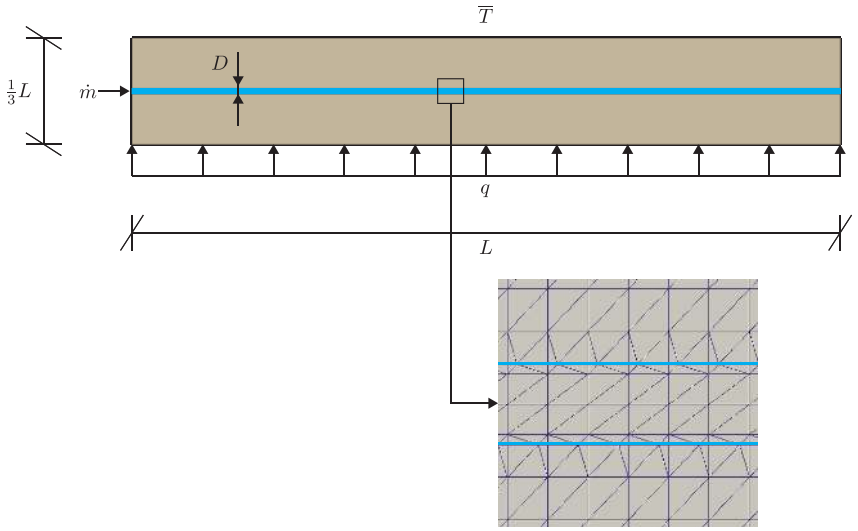


Figure 5.1: CPU cooler test case baseline geometry and sample mesh (side view).

The BCs pictured in Figure 5.1 consist of uniform heat flux of q at the bottom boundary (CPU's thermal energy output) and a fixed temperature of $\overline{T}$ at the top boundary (considered ambient temperature) as well as at the forced inlet flow. The dimensions and other data represented in the figure are

$$\begin{aligned} L &= 45\ mm \\ D &= 0.8\ mm \\ \dot{m} &= 10\ g/min \\ T_{inlet} &= \overline{T} = 20\ ^{\circ}C \\ q &= 150\ W \text{ (uniform heat flux)}, \end{aligned} \tag{5.1}$$

where $\dot{m}$ is the mass flow rate that enters the cooling channel.

The configuration of the cooling channels needs to be defined using some sort of mathematical functions (the IGFEM's *geometric engine* works with level set functions). Because there are at least two material interfaces for the case of vascular material (one channel is defined by two interfaces), there need to be at least two level set functions.

One important aspect to consider regarding the modeling of the cooling channels is the velocity profile inside them. Regardless of the channel configuration, a fully developed flow (this means that the effects of the velocity boundary layer entering the channel are neglected) with a parabolic Hagen–Poiseuille velocity profile given by Equation (5.2) and exemplified in Figure 5.2 is considered,

$$v = 1.5 \frac{\dot{m}}{\rho\, A} \left(1 - \frac{2\, r}{D}\right)^2, \tag{5.2}$$

where r is the distance to the channel's centerline.

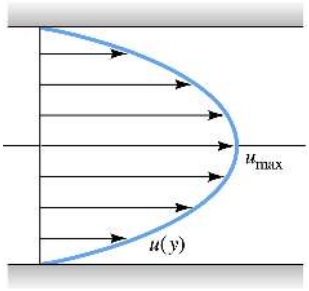


Figure 5.2: Hagen–Poiseuille velocity profile from White [18].

For the purpose of this thesis, the considered cooling channel geometry is defined by two offsets to a sine wave shaped centerline, given by

$$y = a\ sin(k\ x) = a\ sin\left(\frac{2n_{sw}\pi}{L}\ x\right) \tag{5.3}$$

The parameters that define the channel geometry and that need to be tweaked are defined in relation to the centerline and are the amplitude of the channel (A) and number of sine waves in the horizontal direction of the domain (n_{sw}, related to the wavelength of that sine wave). As an example, Figure 5.3 is the representation of a mesh where the channel geometry is defined by ($A = 2\,mm$, $n_{sw} = 2$) and the initial configuration in Figure 5.1 is also considered a sine wave with ($A = 0\,mm$, $n_{sw} = 0$),

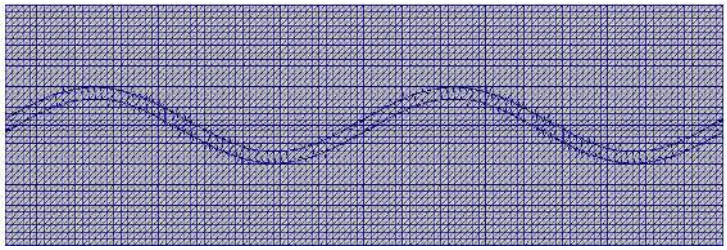


Figure 5.3: Sample mesh for $A = 2\,mm$, $n_{sw} = 2$

Before going any further, it is necessary to study the convergence of the solution in relation to the size of the finite elements. To do this, a random set of parameters of ($A = 4\,mm$, $n_{sw} = 2$) can be used to obtain value the temperature in the far right bottom corner node (which is also the maximum temperature) for different mesh refinements. Plotting these values for six mesh refinement levels originates the graph in Figure 5.4, corresponding to the numbers of elements summarized in Table 5.1.

As further discussed in Chapter 7, being the goal to perform an optimization process with the objective function being the temperature in node four (what is represented in the graph of Figure 5.4), it is desirable to reduce computation time as much as possible, which means to use a mesh refinement that provides a result close to the converged value but still with a relatively low processing time. Taking these premises into consideration, the chosen mesh is the fourth one, with 128134 finite elements. Despite not presenting exactly the same result considered to be the converged value (data point to the far right of the plot), it yields a relative error of 1% with only 55% of the computation time, which is considered to

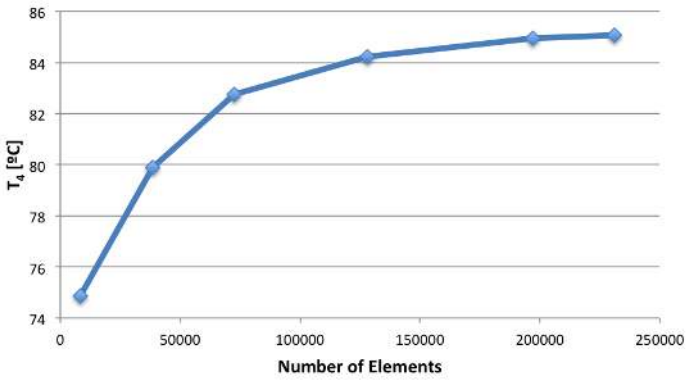


Figure 5.4: Mesh convergence study for $A = 4\,mm$ and $n_{sw} = 2$.

be adequate for the optimization process. After obtaining the optimized geometry, the maximum temperature value for the chosen mesh is going to be compared against the result for the finer mesh to ensure that there isn't an odd discrepancy.

5.1.2 Implementation Efficiency

Another relevant aspect that has not yet been approached is the efficiency of the algorithms implemented for the chosen numerical methods (IGFEM and SUPG). To do this, Table 5.1 can be used to compare the computation times between the vascular material geometry that is used in the previous paragraphs for the mesh convergence study ($A = 4\,mm$, $n_{sw} = 2$) and a similar domain without the cooling channel (solid copper block with the same dimensions and BCs).

$NELEM$	$T_{\text{IGFEM+SUPG}}\,[s]$	$T_{\text{solid}}\,[s]$	$\frac{\Delta t}{T_{\text{IGFEM+SUPG}}}\,[\%]$
8290	13	7	43
38540	43	28	34
72591	96	63	35
128134	163	114	30
197127	255	176	31
231177	294	220	25

Table 5.1: Computation times *vs.* number of finite elements.

From the values in the last column of Table 5.1, which correspond to the percentage of time that is used by IGFEM and SUPG to create the vascular cooling channel and add the contribution of convection to the heat transfer process, some observations can be pointed out:

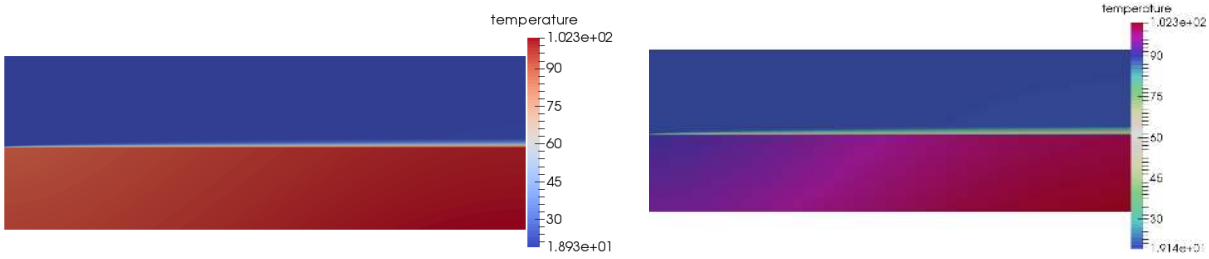
- The percentage of time that is used by IGFEM and SUPG is reduced as the mesh gets finer;
- Taking into account the complexity that is added to the problem with the introduction of a cooling channel with circulating fluid, the relatively low time percentages used to do it show the choice of numerical tools and respective implementation are efficient dealing with this test case, certainly when compared to the computational effort that would be needed to remesh the problem for each

channel geometry. As an example, for the mesh that has been chosen for the optimization, the time used to implement the vascular active cooling is only 30% of the total time.

Regarding the first point above, profiling a bit deeper in the code (using built-in *Python* functionality to check the computation time that each function uses) reveals that the most time consuming step is not the geometric operations taken care of by the *geometric engine* but the assembly functions, more specifically doing the tasks of checking if the "special" assembly function for integration elements is needed or if the material that constitutes the element is fluid and using the correspondent assembly functions (refer back to Algorithm 4 in Chapter 3 to recall the location of this functions within the code). This means that if code optimization is desired it should start in the way these assertions are being made, which is programming language specific, and not in the operations in themselves.

5.1.3 Solution

The first simulation to run has the goal of obtaining the baseline solution, which is the result that is obtained using the initial geometry. This geometry is represented in Figure 5.1, the CPU cooler intersected by a straight channel ($A = 0\,mm$, $n_{sw} = 0$). The result of this simulation is presented in Figure 5.5.



(a) (b)

Figure 5.5: CPU cooler test case baseline geometry solution.

From Figures 5.5(a) and (b), it is possible to observe that the maximum temperature in the CPU cooler's domain is $102.3^{o}C$. This temperature is above $95^{o}C$, which is known from experience to be a threshold above which normal CPUs tend to thermal throttle (reduce computational performance in order to lower the temperature and avoid damages). This result, obtained using this straight channel configuration and data from Vogel [48] and Khonsue [49] is consistent with the temperature values found for real CPUs and fortunate because it gives some growing room to see the effect of the optimization (check if modifying the configuration of a cooling channel to a configuration as simple as a sinusoidal is enough to alleviate this issue).

Furthermore, it is possible to observe from Figure 5.5(b) that:

- The temperature in the top boundary of the cooler appears to be constant, which indicates that the fixed temperature boundary condition is correctly applied;
- The heat source is located at the bottom boundary of the cooler, which indicates that (at least the location of) the heat flux boundary condition is correctly applied;

- Below the cooling channel, there is an increase of the temperature from left to right consistent with direction of the circulating coolant. Taking the bottom boundary of the domain as the horizontal reference axis, this increase goes from approximately $94^{o}C$ in the far left bottom corner to the maximum temperature of $102.3^{o}C$ in the far right corner. This is due to the fact that the fluid enters the channel at a lower temperature and is then warmed as it circulates, increasing its internal energy and reducing its ability to cool the surroundings;

- A vertical temperature gradient (below the channel as well) it is also noticeable, with the temperatures closer to the cooling channel being lower (fluid absorbs energy from surroundings first);

- Above the cooling channel, the temperature appears to have negligible variation, showing a constant color in the graph.

Having presented the baseline temperature distribution resultant from the straight channel configuration of the CPU cooler and made the comments that are deemed relevant, it is adequate to move on to the next test case.

5.2 Engine Cylinder Test Case

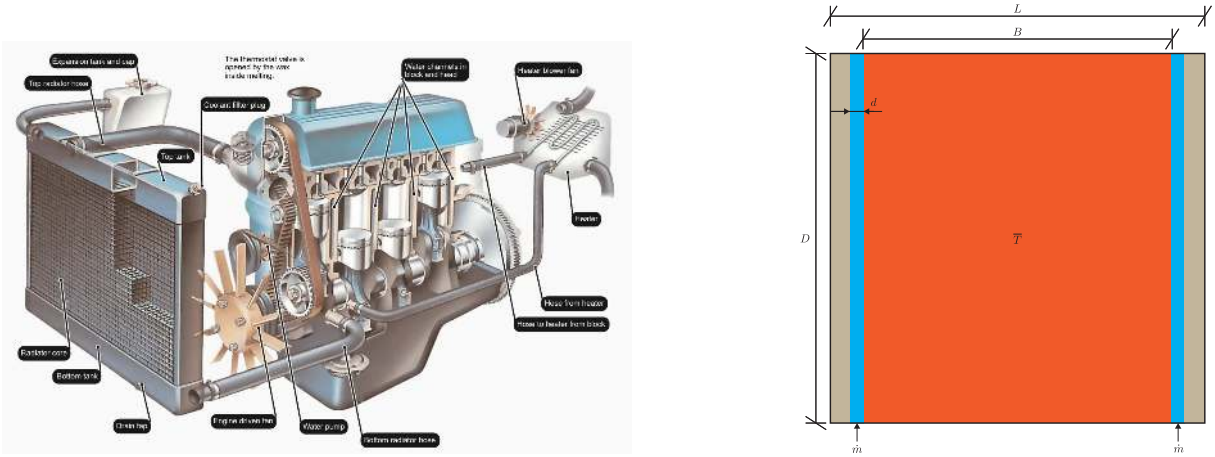
Much like the test case presented in Section 5.1, the goal is to select a set of geometry and BCs inspired by a real life problem in order to check if the order of magnitude of the resulting solution makes sense.

5.2.1 Problem Description

Again falling into the same premises as the previous test case, the problem at hand is a conjugate heat transfer problem with the generic information that is presented in Subsections 4.2.1 and 4.2.2, namely a domain schematic in Figure 4.2, the strong form in Equation (4.1) and the weak form in Equation (4.6).

This time, the goal is to model the cooling system in a car's engine using Figure 5.6(a) as inspiration. Due to symmetry and also with the goal of simplification, the domain to model using the implemented thermal solver is limited to one cylinder, as can be seen in Figure 5.6(b). This test case also proves that the developed IGFEM implementation is capable of handling multiple channels.

Using for inspiration the usual values for engine displacement (D) and cylinder bore (B) from a $1600cc$



(a) Inspiration from Cogniso [50]

(b) Simplification

Figure 5.6: Engine cylinder test case geometry.

engine and Patel et al. [51] for information on the mass flow rates, the dimensions and other data are

$$
\begin{aligned}
B &= 75\,mm \\
D &= 90\,mm \\
d &= 3.2\,mm \\
L &= 90\,mm \\
\dot{m} &= 100\,g/min \\
T_{inlet} &= 20\,^{\mathrm{o}}C \\
\overline{T} &= 180^{\mathrm{o}}C\,,
\end{aligned}
\tag{5.4}
$$

where d is the diameter of the channels, L is the total width of the domain, T_{inlet} is the temperature of the fluid entering the cooling channels and $\overline{T}$ is the fixed temperature in the cylinder. The boundary conditions applied to this test case are the fixed temperature in the cylinder (red in Figure 5.6(b)) which, at $180^{\mathrm{o}}C$, is said in Patel et al. [51] to be the maximum temperature to ensure that the oil film between the cylinder and cylinder wall does not deteriorate and an external temperature of $90^{\mathrm{o}}C$ on the side walls of the engine, which is also said to be the average external engine temperature.

Being described similarly to what has been done in the previous test case, the cooling channel's configuration is also defined using level set functions and the flow inside them given by a parabolic Hagen–Poiseuille velocity profile described by Equation (5.2) and exemplified in Figure 5.2.

The mesh refinement choice to model this test case has been inspired by the convergence study made for the previous test case. The number of elements per unit of area is similar to the one found in the mesh used for the results of that case, which results in another structured triangular (similar to the one in Figure 5.1) mesh with 148047 finite elements.

5.2.2 Solution

With the additional goal of observing the cooling capacity difference using different coolants, this test case has been modeled using one material for the solid phase (cast iron, $k = 55\ W/m \cdot K$, $c_p = 0.46 J/kg{\cdot}K$ and $\rho = 7800 kg/m^3$) and two materials (one at a time) for the coolant (water, $k = 0.6 W/m{\cdot}K$, $c_p = 4.182\ J/kg \cdot K$, $\rho = 998.3\ kg/m^3$ and engine coolant, $k = 0.4153\ W/m \cdot K$, $c_p = 3.6\ J/kg \cdot K$, $\rho = 1030\ kg/m^3$, from Patel et al. [51]).

Despite a mesh refinement that does not differ from the ones used for the previous test case, the geometry modeled here and the nature of the BCs (applying Dirichlet BCs to a surface is equivalent to fix the temperature in all the nodes that are contained in it, which makes the computation more expensive) result in a much higher run time. In this case, the computation time is $24m46s$ and the percentage of time used by the IGFEM SUPG combined solver is only 17%.

Having covered all relevant aspects, it is now adequate to show the results of the simulation, which are presented in Figure 5.7[1]. The figure only represents the solution for the case where water is the coolant, as they visually look the same (only difference is in the temperature of the coolant, which is not noticeable in the figure and is discussed below).

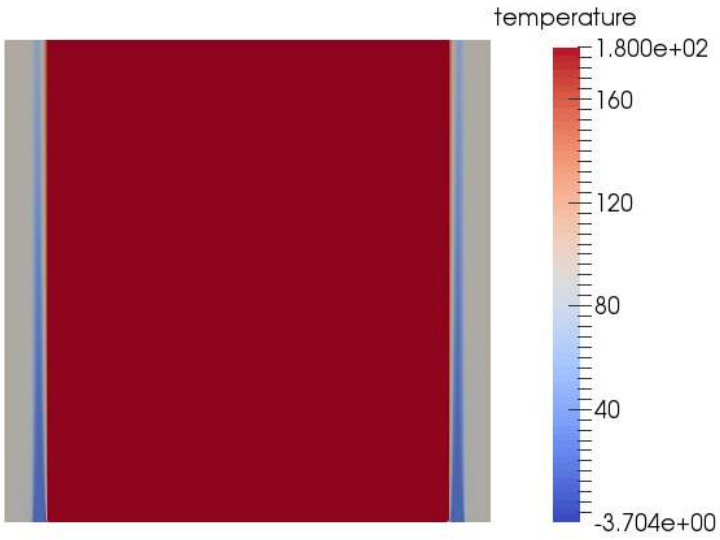


Figure 5.7: Engine cylinder test case solution.

Analyzing the solution presented in Figure 5.7 does not allow for much quantitative comments on the result. The only thing that can be said is that the temperature distribution along the vertical direction of the channel appears to make sense, as it is higher near the top of the channel and lower at the inlet. Additionally, it is higher near the cylinder and lower in the middle (maximum flow velocity).

In order to compare the results obtained for water and engine coolant as circulating fluid and also to validate the order of magnitude of the results in obtained solutions, the chosen metric is the temperature along the outlet of the cooling channels. Being identical for both cooling channels, this temperature distribution for both water and engine coolant as circulating fluid can be found in Figure 5.8.

[1] The minimum temperature in the scale does not make sense, as the minimum value of the solution vector has been searched for computationally and is 20, the temperature of the inlet flow. This mistake could not be fixed in the visualization tool *Paraview*, as the scale appears to be automatic.

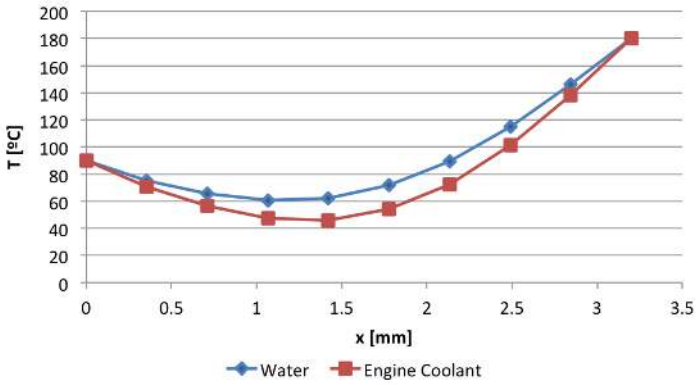


Figure 5.8: Temperature distribution along exit of cooling channel for engine cylinder test case.

Regarding Figure 5.8, the analysis can be summed in the following points:

- The left and right limits are well implemented, $90^{o}C$ being the temperature of the "engine casing" and $180^{o}C$ the temperature emitted by the cylinder;
- The line that corresponds to the simulation done using engine coolant as circulating fluid is lower than the line correspondent to water which indicates, as expected, a better cooling performance. The average temperatures in the outlet are $85.6^{o}C$ for the engine coolant and $95.7^{o}C$ for the water, which means a $10^{o}C$ temperature drop obtained by just changing the coolant;
- The order of magnitude of the average temperature for outlet flows is consistent with real engine running temperatures of about $90^{o}C$ known from experience.

With everything that needed to be observed regarding this test case documented, it is possible to conclude that, apart from the use of multiple channels, the two test cases do not present a lot of relevant differences. Taking that into account, only the CPU cooler continues to undergo an optimization process, as it is better defined and has subjectively more growing room to change the geometry (the cooling channels in the engine cylinder don't have a lot of room to be shaped differently).

The next chapter approaches the choice of optimization tool and method to obtain an optimal configuration for the sinusoidal shaped cooling channel in the CPU cooler test case and Chapter 7 presents the results of this process.

Chapter 6

Optimization Fundamentals

Despite being discussed in Chapter 1 as one of the milestones of this thesis, the optimization process has not yet been approached. This is justified by the fact that first it was necessary to obtain modeling computational tools that present trustworthy results for problems similar to those that need to be optimized. Now that the IGFEM and SUPG have been presented, their implementation verified and a test case and baseline solution analyzed, it is adequate to discuss optimization.

Before getting to the optimization relevant aspects, it is important to mention that despite not developing the optimization loops themselves and using standard optimization packages in the programming language *Python*, an explanation of the choice of the optimization technique and some information about it is presented in Section 6.2.

6.1 Problem Formulation

The general form of a constrained optimization problem (process of minimizing a function with respect to some variables in the presence of constraints on those variables) is

$$\begin{aligned} &\text{Minimize: } f(\mathbf{x}, \mathbf{b}) \\ &\text{Subject to: } x_L < \mathbf{x} < x_U \\ &\qquad\qquad A\,\mathbf{x} \leq \mathbf{b} \text{ (Linear)} \\ &\qquad\qquad h(\mathbf{x}) \leq 0 \text{ (Non-Linear)}\,, \end{aligned} \tag{6.1}$$

where $f(\mathbf{x}, \mathbf{b})$ is the objective function (function that undergoes the optimization process) and the equations under the category "Subject to:" are the constraints. The first equation under this category represents the bounds for the variables $\mathbf{x}$. It is also important to mention that constraints are not limited to linear relations, they can also be non-linear, as exemplified by the last inequality.

For the specific case of this work and the optimization of the conjugate heat transfer problem of the vascular cooled CPU, the objective function is the maximum temperature in the domain and the variables

define the cooling channel's configuration. All of this is further discussed in Chapter 7.

6.2 Optimization Tool

As mentioned in this chapter's introductory notes, this project does not envision developing the optimization loops to be used in vascular cooled materials. Instead, it relies on an optimization package already implemented in a module of the programming language *Python*. This optimization package is found under the *SciPy* module, an open source library of scientific tools, is called *optimize.minimize* and does minimization of a scalar function of one or more variables (see Varoquaux [52]).

6.3 Optimization Methods

Now that the formulation for an optimization problem has been presented, it is adequate timing to discuss the choice of the optimization method for this project. As is mentioned before, the goal is to choose one of the optimization methods implemented in the *SciPy optimize.minimize* module, preferably one with a reduced amount of user input required. The following list provides a basic understanding on the methods present in the *optimize.minimize* module, as well as some advantages and disadvantages of those methods (obtained from the module's documentation and Bonnans et al. [53]):

- The *Nelder-Mead* (or *simplex*) search algorithm is one of the best known algorithms for unconstrained optimization without derivatives. Aside from its simplicity and ease of use, its advantages are related to the fact that the method often gives significant improvements in the first few iterations, quickly producing satisfactory results. On the other hand, the lack of convergence theory can lead to it taking an enormous amount of iterations with negligible improvement in the function value, despite being nowhere near a minimum. This can result in premature termination of iterations;

- The Powell method (or Powell's conjugate direction method) is an algorithm for finding the local minimum of a function using a linear search. The objective function does not need to be differentiable and no derivatives are taken. Being simple, this method is useful in calculating the local minimum of a continuous but complex function, especially one without an underlying mathematical definition, because it is not necessary to take derivatives. On the other hand, the linear searches that are used to obtain the position of the local minimum are more complex, requiring additional methods;

- The Conjugate Gradient (CG) methods are a class of numerical methods that can be used for both solving linear equations and nonlinear optimization. As optimization is concerned, these methods present low memory requirements fast convergence. Despite a convergence speed decrease for ill-conditioned problems, the final result is not affected a great deal by this ill-conditioning. On the other hand, these methods are more suitable for unconstrained optimization;

- The Broyden-Fletcher-Goldfarb-Shanno (BFGS) method is the most commonly used update strategy for implementing Quasi-Newton optimization technique, as it is considered the most efficient one (low number of function evaluations per iteration when compared to similar methods). Unlike Newton optimization techniques, it does not require computing the second derivatives of the objective function. The disadvantages lie mostly on the fact that the quadratic convergence of Newton's method is lost;

- The Newton-CG method combines the moderate memory requirement of the conjugate gradient method with the fast convergence of a quasi-Newton method. It has the inconvenience of requiring the Jacobian (gradient) of the objective function, which is what one of the things trying to be avoided;

- The Limited-memory BFGS-B (L-BFGS-B) is also an optimization algorithm in the family of quasi-Newton methods that approximates the BFGS methods by limiting the amount of computer memory that can be used for parameter estimation and then extends it to handle simple box constraints on variables (example $l_i \leq x_i \leq u_i$). This method is good at evaluating computationally expensive functions because, as the standard BFGS method, it requires a low amount of function evaluation. It presents the same problem as the standard BFGS method, which is that it degenerates into steepest descent method for ill-conditioned problems, requiring a good matrix preconditioner to remedy this situation;

- The Truncated Newton (TNC) methods are a family of optimization algorithms designed for optimization with a large numbers of variables. The only proved advantages are that it sometimes converges in fewer iterations than similar methods and requires less parameter tweaking. Looking at the disadvantages, these methods require that the gradient is suggested and good preconditioning exists in order to work correctly;

- The Constrained optimization by linear approximation (COBYLA) is another optimization method for constrained problems where the derivative of the objective functions is not known. The advantages of this method lie in the very few function evaluations needed for convergence and the disadvantages in the dependency on the starting point (early convergence in local minima when starting point is far from optimal);

- The Sequential Least SQuares Programming (SLSQP) is an iterative method for nonlinear optimization. This method is considered to be one of the most efficient methods for problem minimisation with equality and inequality constraints. Furthermore, it does not require second derivatives to be computed and satisfies the bounds in all iterations;

- Finally, the computational package being discussed also provides implementations of the dog-leg trust-region and of Newton conjugate gradient trust-region algorithms. Because these are meant for unconstrained optimization, they aren't really relevant for the problem of optimizing the CPU cooler's geometry.

6.4 Selected Method

Considering all of the advantages and disadvantages of the optimization algorithms in the list above, which can be directly applied using the mentioned *SciPy* module, the method to be used in the optimization of the CPU cooler is the SLSQP as it is the most well-rounded one in the advantages and disadvantages that matter for the mentioned problem. It may not be the fastest, but it is straightforward to use and there has been no problem in obtaining reliable results.

The usage of the discussed optimization tool and method in a *Python* environment is presents no difficulty very straightforward, which can be demonstrated using a simple example: consider the objective function and single constraint (bound for variable x)

$$\begin{aligned} &\text{Minimize: } f(\mathbf{x}) = sin(x) \\ &\text{Subject to: } 0 < \mathbf{x} < 3\pi\,. \end{aligned} \tag{6.2}$$

The result of this optimization problem is trivial, being -1 the minimum value of $f(x)$ in the indicated domain for an x value of $\frac{3\pi}{2}$.

The implementation of this sample optimization problem using the *SciPy* package in *Python* can be illustrated by Algorithm 7 (x_0 is the user's initial guess for the value of the variable that corresponds to the function minimum in the chosen domain).

Algorithm 7 Sample optimization problem implementation

1: fun = $sin(x)$
2: guess = x_0
3: method = 'SLSQP'
4: bnds = $(0, 3\pi)$
5: cons = ()
6: result = minimize(fun, guess, method, bnds, cons)
7: print result (if successful, returns the final value of the variables and function, the number of iterations and elapsed time)

Chapter 7

Test Case Optimization Results

It is in this final results chapter that the ability to model multiphase problems provided by the union of IGFEM and SUPG is combined with an optimization algorithm provided by the *SciPy optimize.minimze* package. While Section 7.1 approaches the specifics of the optimization process (objective, constraints and design variables), Section 7.2 presents the optimization results (obtained using the optimization method selected in Chapter 6) that are deemed necessary and interesting.

The goal is to optimize a vascular rectangular domain with a cooling channel that mimics the cooling block of a computer's processor (central processing unit, CPU). The data is inspired by a real life problem, namely the heat flux produced by the CPU and the temperatures involved and a first baseline geometry has already been modeled in Chapter 5. Being a real life problem, the results of the optimization process can be analyzed in an easier way, since the order magnitude of the usual temperatures is empirically known.

7.1 Bounds and Constraints

Before presenting results of the optimization, it is important to formally present its formulation, similarly to what has been done in Chapter 6 for a general optimization problem. The requirements to identify are the objective function, bounds and constraints. The objective function is, as already mentioned, the maximum value of the temperature in the domain. As for the bounds and constraints, they are the first two expressions in Equation (7.1) and the last expression, respectively, and they regard the optimization parameters that are the amplitude of the centerline sine wave and number of "waves" in the horizontal direction of the domain), resulting

$$\begin{aligned} \text{Minimize: } & max[T(\mathbf{a}, \mathbf{n_{sw}})] \\ \text{Subject to: } & 2 < \mathbf{a} < 4 \; [mm] \\ & 2 < \mathbf{n_{sw}} < 4 \\ & \Delta p \leq 1.5 \times \Delta p_{\text{straight interface}} \; , \end{aligned} \tag{7.1}$$

where Δp and $\Delta p_{\text{straight interface}}$ are the pressure drop in the channel due to viscous effects for the selected interface and straight interface, respectively. Refering back to White [18] once again, the pressure drop in a channel due to viscous effects with an Hagen-Poisseuile velocity profile is given by

$$\Delta p = \frac{8\mu s Q}{\pi R^4} \propto s \Rightarrow \Delta p \leq 1.5 \times \Delta p_{\text{straight interface}} \Leftrightarrow s \leq 1.5 \times L\,, \tag{7.2}$$

where μ is the dynamic viscosity of the fluid, s is the length of the channel, Q is the volumetric flow and R the radius of the channel. Taking into account that all parameters remain constant but the length of the channel, the constraint presented in Equation (7.1) is equivalent to the one presented here. The length of the channel defined by a sine wave shaped centerline is given by the arc length of that same sine

$$s = \int_0^L \sqrt{1 + (A\,k\,cos(k\,x))^2} dx\,. \tag{7.3}$$

Choosing these values for the bounds of the optimization parameters has two reasons. The most relevant reason is geometrical: using two offset curves from a sine wave shaped centerline does not normally produce a constant width channel. To remedy this situation (too solve it completely the channel would have to be modeled in some other way, for example using sine generated curves with constant radius), an empiric scaling function is used, but this function is not effective for very high amplitudes or very short wavelengths. The second reason is that using high amplitudes for the sinusoidal shaped centerline leads to uneven cooling, something that might not be desirable in the case of a CPU.

Finally, the pressure drop ratio constraint is introduced to add some real life restriction to the problem by limiting the required driving force of an hypothetical pump that would have to drive the coolant through the channel. This constraint is what generates interesting results of the optimization, aside from the temperature differences resultant for different channel configurations and if it is not introduced, the result of the optimization is that the parameters take the value of their upper bounds (verified using this formulation and optimizer without this constraint). This means that the resulting geometry would have the largest possible amplitude value and number of waves in the domain, as this means more fluid inside it, which leads to a higher cooling capacity (*via* transferring energy from the solid phase to the fluid, which keeps on being renovated with new fluid at ambient temperature).

7.2 Enhanced Solution

Now that all the premises for the optimization are clearly and formally presented, it is adequate to obtain the results of the optimization using the developed numerical tools and the chosen built-in package for optimization method (SLSQP). In order to draw some comments on the effects of the different channel geometries in the maximum temperature in the CPU cooler, Table 7.1 can be built with the results of maximum temperature and pressure drop for four different geometries: the straight interface configuration that has already been analyzed in Subsection 5.1.3, the lower and upper limits of the optimization parameters' bounds and the optimized result.

Before presenting the graphs for the resulting temperature fields for each geometry, it is convenient

$a\ [mm]$	n_{sw}	$max[T(A, n_{sw})]\ [^{o}C]$	$\Delta p/\Delta p_{\text{s.i.}}$	Remarks
0	0	102.3	1	Straight channel (baseline)
2	2	84.9	1.1	Design variables at lower bound
4	4	59.4	1.8	Unconstrained result (violated constraint)
3.15	3.7	66.9	1.5	Optimal design (active constraint)

Table 7.1: Results of CPU cooler's optimization problem.

to draw some observations from the most important metric in this optimization, which is the maximum temperature within the CPU cooler's domain:

- The result for the straight channel geometry serves merely to evaluate the results for the remaining geometries. As would be expected, the ratio between pressure drops is exactly one, as the value being tested and reference are the same;
- The result in the second data row of the table (for $A = 2\,mm$ and $n_{sw} = 2$) corresponds to the case where both lower bounds of the optimization parameters are used. As is clear with a maximum temperature of $84.9\,^{o}C$, the temperature of the domain is now below the aforementioned thermal throttle threshold of $95^{o}C$, which means that any geometries that improve on this result already complies with the minimum requirements. Presenting a pressure drop due to viscous effects ratio of 1.1 in relation to the straight channel configuration means that the pressure drop constraint presented in Equation (7.1) is not yet active and that there is still room for improvement;
- The next row of the table (for $A = 4\ mm$ and $n_{sw} = 4$) corresponds to the upper bounds of the optimization parameters. With a resulting maximum temperature of $59.4^{o}C$, it is the most favorable geometry for reducing the maximum domain temperature considering the bounds chosen for the optimization parameters. It is very interesting to observe that changing the geometry from a straight channel to a simple sinusoidal shaped channel leads to a maximum temperature difference of $43^{o}C$. On the other hand, looking at the last value in this row indicates that the pressure drop due to viscous effects ratio is higher than 1.5, which means that this constraint is violated and this result is not valid for this optimization problem (it corresponds to the unconstrained optimization previously mentioned);
- Finally, the fourth and last data row of the table (for $A = 3.15\,mm$ and $n_{sw} = 3.7$) corresponds to the final result of the optimization. Regarding this result, several comments are in order:
  - The maximum temperature resulting from the optimized geometry is $67.1^{o}C$ and falls, as expected, between the results obtained for the geometries defined by the lower and upper bounds;
  - It is noticeable that the constraint regarding the pressure drop due to viscous effects is active since the optimized geometry yields a value of 1.5, the maximum allowed in Equation (7.1);
  - The geometry that results from this optimization has been tested with the finest mesh from the convergence study in Figure 5.4 and the resulting temperature is $66.9^{o}C$. This means that

the difference between the temperatures obtained for both meshes is approximately 0.15%, which is considered negligible, reassuring confidence in the results that are presented;

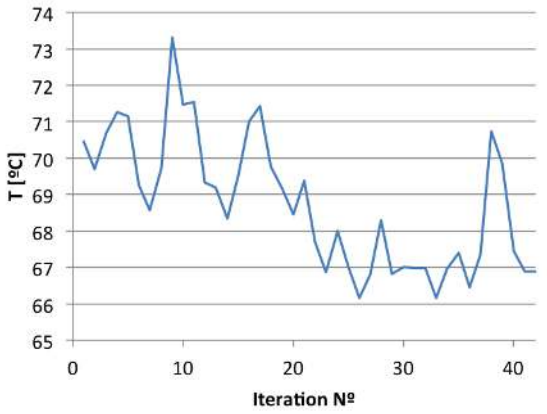
- This optimization's result is that the temperature of this CPU cooler block under heavy load goes from a value way above the optimal performance and integral safety threshold to a very favorable temperature for the same conditions (temperatures under $80^{o}C - 85^{o}C$ are considered to be a synonym of adequate cooling performance in various sources such as computer forums);
- From analyzing the resulting geometry, it is possible to conclude that the maximum temperature is more sensitive to the parameter n_{sw} than to the amplitude of the sinusoidal centerline, as the "selected" geometry has a higher value for n_{sw}. This fact cannot be explained by the sensitivity of the channel's length to the same parameters, as equal increments for each parameter produce the same exact result (for example, the length of the channel for the geometry $A = 3\,mm$ and $n_{sw} = 2$ is exactly the same as for $A = 2\,mm$ and $n_{sw} = 3$);
- The use of the chosen numerical techniques allowed this geometry to be optimized in just (about) $1h40$ using a personal system running an *Intel Core i7* processor clocked at $2.2GHz$ with $16GB$ of memory and on *OS X Yosemite*, despite the forty two iterations that the optimizer needed to obtain the result. A special emphasis must be given to IGFEM in problems such as the one being analyzed here, as it is a very interesting method that efficiently deals with the discontinuities and without which an optimization process to this problem would be much more user time consuming and cumbersome (with a constant need for remeshing).

With all the comments deemed necessary regarding the concrete result of the maximum domain temperatures, it is also interesting to look at the evolution of relevant optimization data with iteration number. The evolution of the maximum domain temperature, pressure drop due to viscous effects ratio and design variables evolution with the iteration number is represented in Figure 7.1. Rounding up the observations regarding Figures 7.1(a) and (b), it is necessary to point out that there are iterations with a lower maximum domain temperature than the final result of the optimization. These values correspond to the pressure drop ratios that are above the dashed line limit in Figure 7.1(b). On the other hand, the dashed lines labeled as xL and xU in Figure 7.1(c) correspond to the lower and upper bounds, respectively, of the design variables a and n_{sw}.

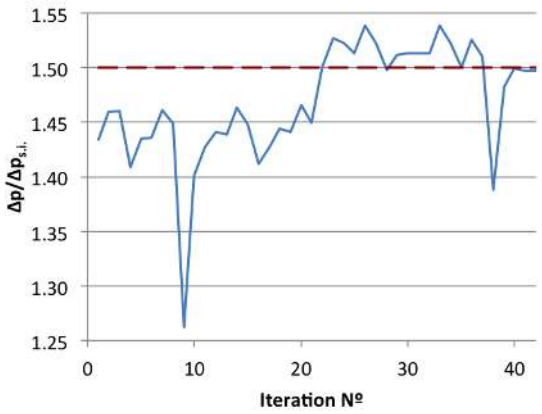
It is now adequate to analyze the domain temperature distributions for the geometries that are being considered. The result for the first geometry, which corresponds to the design variables at their lower bounds ($A = 2\,mm$, $n_{sw} = 2$), is represented in Figure 7.2.

When comparing the temperature distribution obtained in Figure 7.2(b) with what has been obtained for the straight interface in Figure 5.5, it is noticeable that:

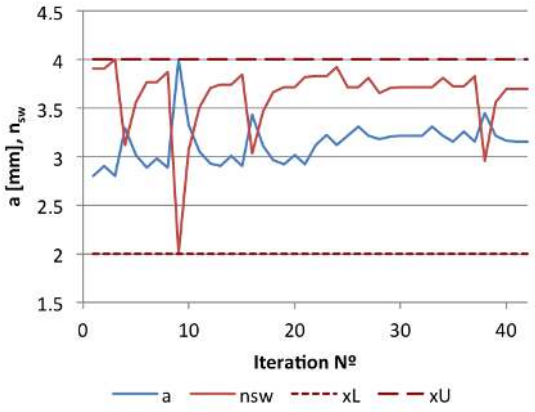
- Below the cooling channel, the difference between far right and far left bottom corner nodes' temperatures is $10.5^{o}C$, which is higher than the resulting $8.2^{o}C$ from the straight interface geometry. This means that the greater quantity of fluid in the channel and the added contact area contributes



(a) Maximum domain temperature

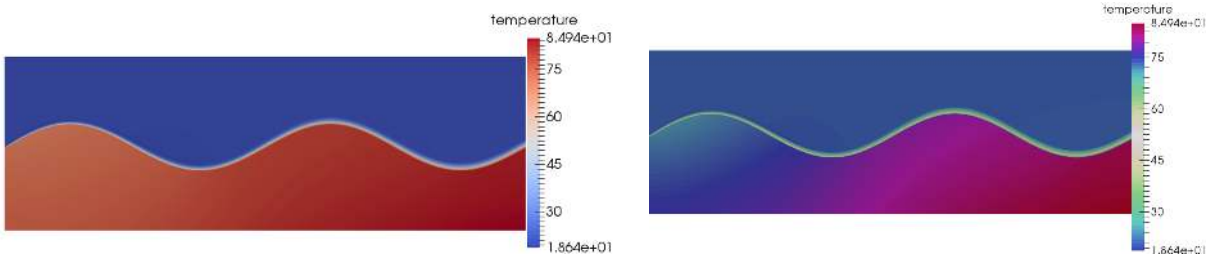


(b) Pressure drop due to viscous effects ratio



(c) Design variables

Figure 7.1: Evolution of relevant optimization data with iteration number.



(a) (b)

Figure 7.2: CPU cooler optimization problem's result for $A = 2\ mm$ and $n_{sw} = 2$.

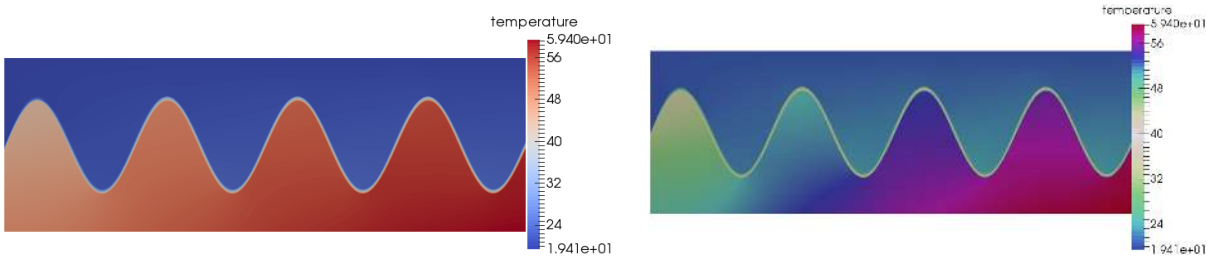
to a more efficient cooling (lower maximum temperature) but also to a bigger temperature gradient in the bottom boundary of the domain;

- Despite not being entirely translated into vertical temperature gradients, the stratification of the temperature below the cooling is much more visible in this geometry when compared to the straight interface. This is noticeable due to the well defined color zones, with the transitions happening near the regions of the peaks and valleys of the sinusoidal channel;

- Above the cooling channel, close to the flow outlet, it is possible to identify a higher temperature

zone with a slightly different color (this is even more noticeable in the geometries presented below).

Before presenting the temperature field for the optimized geometry, it is also interesting to show the result of the unconstrained optimization (Figure 7.3), which means a geometry with $A = 4\ mm$ and $n_{sw} = 4$.



(a) (b)

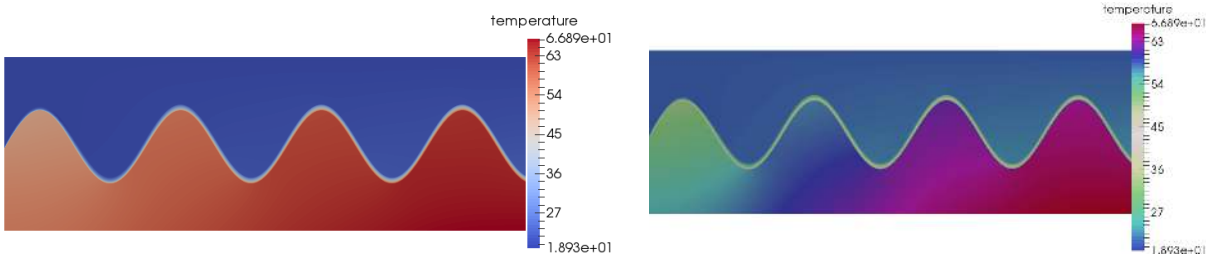
Figure 7.3: CPU cooler optimization problem's result for $A = 4\ mm$ and $n_{sw} = 4$.

Commenting the resulting temperature field falls in the same topics that have already been approached for the previous geometries:

- The temperature gradient along the bottom boundary of the domain is once again bigger than in the previous geometry (Figure 7.2), revealing a value of $10.8^{o}C$;
- Under the cooling channel, the different temperature zones are even more noticeable in this case, as there is clear stratification of the temperature depending on the position under which "wave" that is being considered;
- The warming of the upper part of the domain is also more noticeable for this geometry, as is clearly observable that the temperature above the cooling channel tends to be clearly higher starting on the second "wave".

Furthermore, it is possible to conclude that the bigger is the length of channel inside the cooler, the better distributed will the heat be within the domain (zones that are clearly at a higher temperature and zones clearly colder tend to vanish).

Finally, it is time to present the temperature field for the optimized geometry, which can be found in Figure 7.4. In order to better evaluate this temperature distribution, four different perspective $3D$ views have been obtained and are presented in Figure 7.5.



(a) (b)

Figure 7.4: CPU cooler optimization problem's result final design in $2D$.

Presenting an optimized geometry that is not a lot different from what is presented for the previous case in Figure 7.3, the comments that have been made for it generally apply to the current case of Figure 7.4, except for the bottom interface temperature gradient which is slightly bigger here, $11.7^{o}C$, perhaps due to the smaller amplitude of the sinusoidal channel.

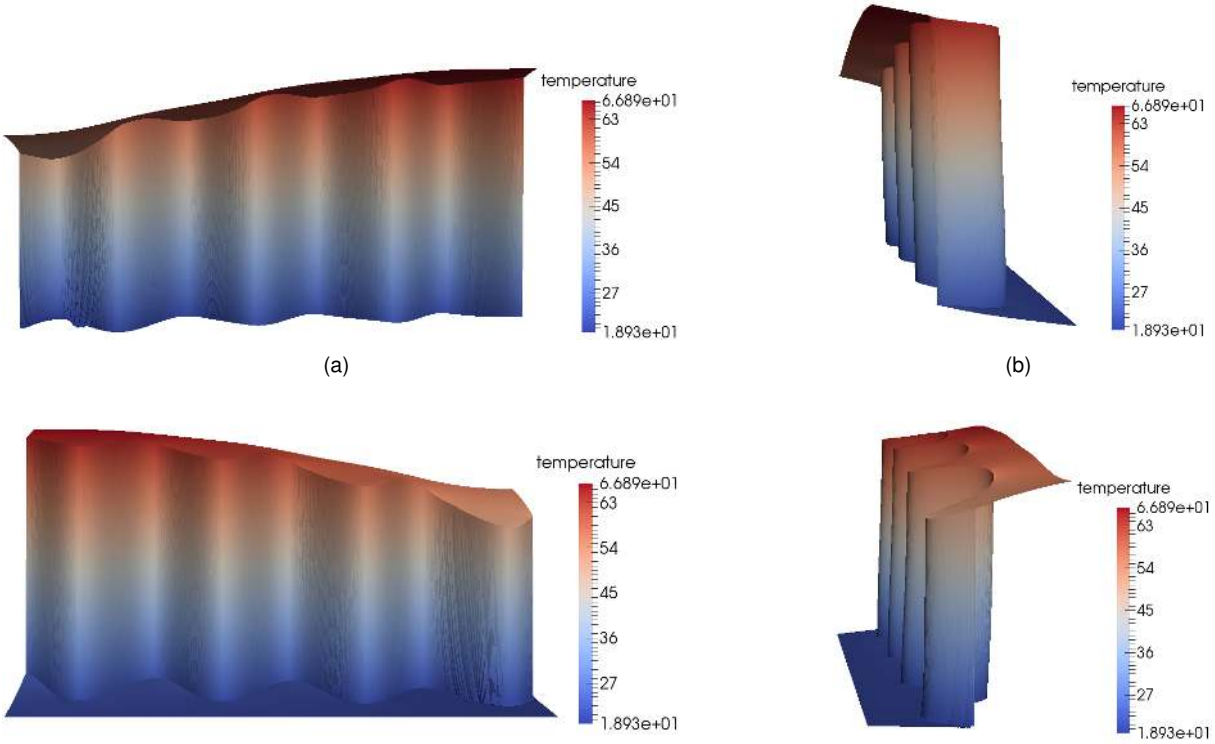


Figure 7.5: CPU cooler optimization problem's result final design $3D$ views.

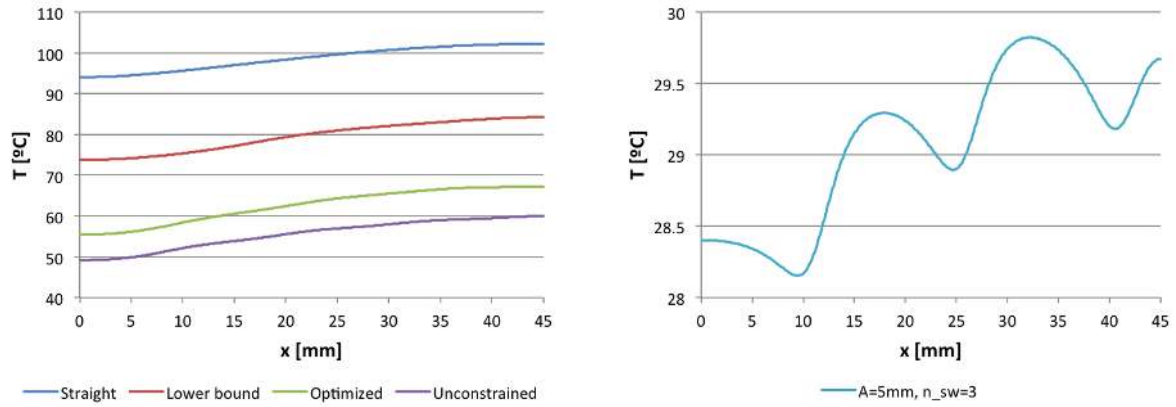
Regarding the $3D$ extrusion of the temperature field for the resulting optimized geometry, the intuitive observations that can be listed are:

- The IGFEM proves once again to be a very powerful tool. It is very advantageous to be able to use nonconforming structured mesh such as the ones being used here and obtain the smooth resolution on the interface that can be observed in the figure for any desired channel geometry;
- The assertion above regarding the constant temperature in the top boundary of the cooler is now obvious, confirming that the fixed temperature BC is applied correctly;
- Analyzing the horizontal slope of the temperature distribution decrease in the part of the domain below the cooling channel clearly leads to the conclusion that the cooling capacity of the fluid reduces (the temperature tends to stabilize). This is due to the increase in temperature of the coolant. On the other hand, even if less noticeable, this slope in the area above the cooling channel tends to increase due to the same reason, which is the higher temperature of the fluid that this time warms its colder surroundings;
- The vertical temperature gradient for different x positions follows the same trend that is reported in the previous bullet point and due to the same reasons: it decreases with x in the area below

the cooling channel and increases with x in the area above it, both due to the increase in the temperature of the flow.

One thing to note again is that if the maximum temperature result obtained with the considered mesh has a negligible difference to the finer mesh of the convergence analysis, the temperature distributions (Figures 7.4 and 7.5) are indistinguishable regardless of the mesh used.

As a complement, it is also interesting to observe the temperature distribution along the bottom boundary of the cooler, which would be the temperature of the CPU. In Figure 7.6, this graph is plotted for both the tested geometries and an alternative geometry with $A = 5\,mm$ and $n_{sw} = 4$, where the goal is to point out the uneven cooling mentioned in Section 7.1.



(a) Tested geometries

(b) Alternative geometry with $A = 4\,mm$ and $n_{sw} = 3$

Figure 7.6: Temperature distribution along bottom boundary of the cooler.

Due to the moderate amplitudes of the sinusoidal channels from the geometries that fall in the bounds and constraints of the optimization problem, the resulting temperature distributions along the bottom boundary of the cooler depicted in Figure 7.6(a) have a linear trend, which means that the proximity of the cooling channel to the boundary doesn't affect the temperature distribution.

On the other hand, 7.6(b) represents a zoomed view of the same temperature distribution for a geometry with a a larger sinusoidal channel amplitude where, despite the even lower maximum temperature obtained, the influence of the proximity of the cooling channel to the boundary clearly affects the temperature distribution. This happens due to the different conductivity of the materials: the fact that the cooling channels do not have a negligible diameter and the thermal conductivity of the coolant is much lower than the one in the solid leads to a greater difficulty in the heat transferal process, which results in a temperature "increase" when the channel gets close to the boundary. To avoid this is one of the reasons to choose a conservative upper bound for the amplitude of the cooling channel as this might not be desired if the cooler is going to be used for several CPU models. If the position of the computing cores of the CPU was constant, the geometry could be optimized with this considerations to obtain an even temperature distribution.

Chapter 8

Conclusions

8.1 Achievements

The work presented in this master thesis has been developed in the Structural Optimization group of *TU Delft* and all computational implementations done within their recent finite element computational tool *hybrida*. This means that all progress has been done with careful organization considerations and had to follow the previous code's structure and implementation techniques. If using the already implemented functionality of the computational tool simplified implementation, taking all of that is mentioned above and the fact that the code developed in this work is also meant for other applications (it is already being used in the domain of crack propagation and contact mechanics) into account means that the list of achievements below required not only the understanding of the computational methods but also a lot of "design" thinking:

- The basic formulation of a state-of-the-art discretization technique described by the Interface-enriched Generalized Finite Element Method (IGFEM) has successfully been implemented for efficient mesh-independent modeling of $2D$ problems with discontinuous gradient fields. This efficiency is shown not only in terms of low computational cost but also in the ease of use, the implementation of a problem's boundary conditions and interface(s) is straightforward and does not require extensive knowledge on the computational tool being used (it can almost be used as a *black box*);

- Modifying the original method by using modifications studied in different literature sources, the developed computational tools have the potential to be very helpful in optimization studies, removing the burden and computational cost of creating conforming finite element meshes. Furthermore, the implementation of this method is advantageous over standard Generalized Finite Element Methods (GFEM), since it is said to have lower computational cost and easier enforcing of Dirichlet boundary conditions and evaluation of enrichment functions;

- Even though the formulation of the IGFEM proved to be straightforward to understand and even implement for simple situations, the variety of geometries resultant from slicing the various $2D$

finite elements that are intersected by the interfaces demands a general implementation to avoid problems. This has required imaginative solutions compatible with the remaining code's structure and an iterative process to obtain algorithms up to the task of dealing with these solutions;

- The implementation of these computational tools has proved to be successful by verification using test cases from literature where optimal convergence rates have been obtained;

- The IGFEM solver that has been implemented in this work has been used to solve the problem of creating models of the vascular actively-cooled materials that serve as motivation to this thesis using level set functions to describe the material interfaces and nonconforming meshes. This process would have otherwise been more computationally expensive and required much more user intervention;

- Having solved the geometrical problem, the IGFEM still presents oscillatory solution fields for convection dominated problems (characteristic of the methods that use a Galerkin formulation of convection). As a solution for this problem, the Streamline Upwind Petrov-Galerkin (SUPG) method has been implemented for heat transfer processes where heat convection is not negligible, stabilizing the solution and reducing the amplitude of fictitious oscillations;

- Regarding the complete process of understanding and implementing the SUPG, the situation is opposite to what is reported for the IGFEM: the method does not require much additional implementation related effort, with almost all required variables and matrices already computed for the standard Galerkin technique (the SUPG adds a correction term to it), but its formulation is theoretically demanding and not quite so straight-forward, especially for two dimensions ($3D$ should present some additional complexity, but this case has not been investigated in this thesis);

- The implementation of SUPG has proven once again to have been successful, with results for identical problems matching the ones found in literature for the initial method's formulation;

- Combining both the IGFEM and SUPG results in an efficient and stable IGFEM solver that can be used to solve the conjugate heat transfer problems of vascular actively-cooled materials. The use of this thermal solver in a vascular material with convection in its channel only results in a computation time that is (on average) 30% slower when compared to the solution of a solid material block where only conduction is present;

- The stable IGFEM thermal solver has been used to optimize a $2D$ vascular CPU cooler with a uniform heat flux boundary condition and fixed temperature boundary condition. Without the need to remesh and using a nonconforming structured mesh, a standard *Python* optimizer has been used to obtain a geometry that reduced the temperature of the CPU cooler from $102.3^{\circ}C$ for a straight line channel geometry (clearly above the limits for optimal performance and physical integrity) to $66.9^{\circ}C$ for a sinusoidal shaped channel (good running maximum temperature under load) with a pressure drop due to viscous 1.5 times bigger;

8.2 Future Work

As of future work that can be developed in the subject of this thesis, there are a few interesting ideas that might provide inspiration for additional results:

- Implement specially developed optimization routines for the problems being solved. The optimization method that has been combined with the stable IGFEM solver is contained in the standard optimization packages of *Python* and, not having a lot of information on the objective function it is optimizing, it is not as fast as it could be. Developing special optimization routines can help improve this and even lead to interesting academic results, as there is not a lot of research on the topic;

- Generalize methods implemented to enable $3D$ modeling and optimization. This topic can lead to very promising results in a finite element computational tool such as *hybrida* (that does not have mesh generating capabilities), as it has the potential of helping to solve complex geometries without conforming meshes. However, slicing $3D$ elements is even more computationally demanding than $2D$ elements and coming up with the algorithms for this implementation would probably constitute an entire master thesis;

- Finally, the last suggestion for future work is to apply the developed tools to real optimization problems that have well defined bounds, constraints and goals. With this, one would also be accessing the validity of what has been implemented.

Bibliography

[1] Y. Bar-Cohen. Biomimetics—using nature to inspire human innovation. *Institute of Physics Publishing*, 1 (2006):1–12, Apr. 2006. doi: 10.1088/1748-3182/1/1/P01.

[2] M. Stone. Applications of biomimetics. *The Bridge*, 36(4):14–18, 2006. USPS 551-240.

[3] P. Fratzl. Biomimetic materials research: what can we really learn from nature's structural materials? *Journal of the Royal Society*, 4 (2007):637–642, July 2007. doi: 10.1098/rsif.2007.0218.

[4] L. Wahl, S. Maas, D. Waldmann, A. Zurbes, and P. Freres. Shear stresses in honeycomb sandwich plates: Analytical solution, ,finite element method and experimental verification. *Journal of Sandwich Structures and Materials*, 14(4):449–468, July 2012. doi: 10.1177/1099636212444655.

[5] K. Toohey, N. Sottos, J. Lewis, J. Moore, and S. White. Self-healing materials with microvascular networks. *Nature Materials*, 6:581–585, Aug. 2007. doi: 10.1038/nmat1934.

[6] T. L. Bergman, A. S. Lavine, F. P. Incropera, and D. P. Dewitt. *Fundamentals of Heat and Mass Transfer*. John Wiley and Sons, 7^{th} edition, 2011. ISBN: 0470501979.

[7] W. Lucas, A. Groover, R. Lichtenberger, K. Furuta, S.-R. Yadav, Y. Helariutta, X.-Q. He, H. Fukuda, J. Kang, S. Brady, J. Patrick, J. Sperry, A. Yoshida, A.-F. López-Millán, M. Grusak, and P. Kachroo. The plant vascular system: evolution, development and functions. *Journal of Integrative Plant Biology*, 55(6):294–388, Mar. 2013. doi: 10.1111/jipb.12041.

[8] A. Nilsson. Blood flow, temperature and heat loss of skin exposed to local radiative and convective cooling. *Journal for Investigative Dermatology*, 88:586–593, 1987.

[9] S. Soghrati and P. Geubelle. A 3D interface-enriched generalized finite element method for weakly discontinuous problems with complex internal geometries. *Computational Methods Applied to Mechanical Engineering*, 217-220:46–57, Jan. 2012. doi: 10.1016/j.cma.2011.12.010.

[10] S. Soghrati, A. M. Aragón, C. A. Duarte, and P. H. Geubelle. An interface-enriched generalized FEM for problems with discontinuous gradient fields. *International Journal of Numerical Methods in Engineering*, 89:991–1008, Aug. 2011. doi: 10.1002/nme.3273.

[11] L. Valdevit, N. Vermaak, F. Zok, and A. Evans. A Materials Selection Protocol for Lightweight Actively Cooled Panels. *Journal of Applied Mechanics*, 75, Nov. 2008. doi: 10.1115/1.2966270.

[12] N. Vermaak, L. Valdevit, and A. Evans. Materials property profiles for actively cooled panels: An illustration for scramjet applications. *The Minerals, Metals and Materials Society Journal*, 40:877–890, Apr. 2009. doi: 10.1007/s11661-008-9768-y.

[13] H. Saravanamuttoo, G. Rogers, H. Cohen, and P. Straznicky. *Gas Turbine Theory*. Prentice Hall, 6^{th} edition, Sept. 2008. ISBN: 0132224372.

[14] A. Aragón. *Computational Design of Microvascular Biomimetic Materials*. PhD thesis, University of Illinois at Urbana-Champaign, 2010.

[15] A. Aragón. Website, May 2016. `https://svn.3me.tudelft.nl/trac/hybrida`.

[16] S. Soghgrati, P. Thakre, S. White, N. Sottos, and P. Geubelle. Computational modeling and design of actively-cooled microvascular materials. *International Journal of Heat and Mass Transfer*, 55: 5309–5321, June 2012. doi: 10.1016/j.ijheatmasstransfer.2012.05.041.

[17] A. Bejan. *Convection Heat Transfer*. John Wiley $\&$ Sons, 4^{th} edition, 2013. ISBN: 0470900376.

[18] F. White. *Fluid Mechanics*. McGraw-Hill, 4^{th} edition, Dec. 1998. ISBN: 007069673.

[19] J. Reddy. *An Introduction to the Finite Element Method*. McGraw-Hill, 3^{rd} edition, 2006. ISBN: 0071244735.

[20] P. Gould. *Introduction to linear elasticity*. Springer, 2^{nd} edition, 1994. ISBN: 1461287285.

[21] N. Sukumar, D. Chopp, N. Moes, and T. Belytschko. Modeling holes and inclusion by level sets in the extended finite element method. *Computer Methods in Applied Mechanics and Engineering*, 190(46-47):6183–6200, 2001.

[22] M. Schweitzer. *Meshfree and Generalized Finite Element Methods*. PhD thesis, Mathematisch–Naturwissenschaftlichen Fakultät der Rheinischen Friedrich–Wilhelms–Universität Bonn, 2008.

[23] A. Hrennikoff. Framework Method and its technique for solving plane stress problems. *IVBH Abhandlungen*, 9:217–248, 1949.

[24] J. Cuillière, V. Francois, and J. Drouet. Automatic mesh generation and transformation for topology optimization methods. *Computer-Aided Design*, 45(12):1489–1506, Dec. 2013.

[25] K. Ho-Le. Finite element mesh generation methods: a review and classification. *Computer Aided Design*, 20(1):27–38, Jan. 1988. doi: 0010-4485/88.

[26] Z. Yue and D. Robbins. Adaptive superposition of finite element meshes in non-linear transient solid mechanics problems. *International Journal for Numerical Methods in Engineering*, (72):1063–1094, May 2007. doi: 10.1002/nme.2067.

[27] M. Beall, J. Walsh, and M. Shephard. Accessing CAD geometry for mesh generation. *Proceedings, 12th International Meshing Roundtable*, pages 33–42, Sept. 2003.

[28] S. Soghrati. Hierarchical interface-enriched finite element method: An automated technique for mesh-independent simulations. *Journal of Computational Physics*, 275:41–52, July 2014. doi: 10.1016/j.jcp.2014.06.016.

[29] T. Belytschko, R. Gracie, and G. Ventura. *A Review of Extended Generalized Finite Element Methods for Material Modelling*. Modeling and Simulation in Materials Science and Engineering, 17 edition, 2009. 043001.

[30] K. Schwebke and S. Holzer. Some Remarks on Generalized Finite Element Methods (GFEM) in Solid Mechanics. *WCCM V - Fifth World Congress on Computational Mechanics*, July 2002.

[31] J. Melenk. *On Generalized Finite Element Methods*. PhD thesis, The University of Maryland, 1995.

[32] I. Babuska, U. Banerjee, and J. Osborn. Generalized Finite Element Methods: Main Ideas, Results, and Perspective. *International Journal of Computational Methods*, 01(67), June 2004. doi: 10.1142/S0219876204000083.

[33] J. Melnek and I. Babuska. The partition of unity finite element method: basic theory and applications. *Computer Methods in Applied Mechanics and Engineering*, 139:289–314, 1996.

[34] S. Soghrati and P. Geubelle. A 3D interface-enriched generalized finite element method for weakly discontinuous problems with complex internal geometries. *Comput. Methods Appl. Mech. Engrg.*, 217(220):46–57, Jan. 2012. doi: 10.1016/j.cma.2011.12.010.

[35] M. Tan, M. Safdari, A. Najafi, and P. Geubelle. A NURBS-based interface-enriched generalized finite element scheme for the thermal analysis and design of microvascular composites. *Computer Methods in Applied Mechanics and Engineering*, 283:1382–1400, 2015.

[36] S. Soghrati, C. Duarte, and P. Geubelle. An adaptive interface-enriched generalized FEM for the treatment of problems with curved interfaces. *International Journal for Numerical Methods in Engineering*, 102:1352–1370, Jan. 2015. doi: 10.1002/nme.4860.

[37] S. Soghrati and J. Barrera. On the application of higher-order elements in the hierarchical interface-enriched finite element method. *International Journal for Numerical Methods in Engineering*, 105 (6):403–415, June 2015. doi: 10.1002/nme.4973.

[38] O. Zienkiewicz and R. Taylor. *The Finite Element Method*, volume Volume 1 The Basis. Butterworth Heinemann, 5^{th} edition, 2000. ISBN: 0750650494.

[39] E. Abbena, S. Salamon, and A. Gray. *Modern Differential Geometry of Curves and Surfaces with Mathematica*. Textbooks in Mathematics. Chapman and Hall, 3^{rd} edition, June 2006. ISBN: 1584884487.

[40] C. Ericson. *Real Time Collision Detection*. Elsevier Inc., 2005. ISBN: 1558607323.

[41] C. Geuzaine and J. Remacle. Gmsh: A three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. Website, May 2016. `http://gmsh.info/`.

[42] I. Babuska and B. Szabo. On the rates of convergence of the finite element method. *International Journal for Numerical Methods in Engineering*, 18:323–341, 1982.

[43] A. Ramos, A. M. Aragón, S. Soghrati, P. H. Geubelle, and J. Molinari. A new formulation for imposing Dirichlet boundary conditions on non-matching meshes. *International Journal of Numerical Methods in Engineering*, 2015. doi: 10.1002/nme.4898.

[44] A. Brooks and T. Hughes. Stramline Upwind/Petrov-Galerkin formulations for convection dominated flows with particular enphasis on the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 32:199–259, 1982. doi: 0045-7825/82/0000-0000/02.75.

[45] J. Donea and A. Huerta. *Finite element methods for flow problems*. John Wiley and Sons, Ltd, 2003. ISBN: 0471496669.

[46] S. Soghrati. *An Interface-enriched Generalized Finite Element Method for the design of actively-cooled microvascular composites*. PhD thesis, University of Illinois at Urbana-Champaign, 2013.

[47] T. Fries and H. Matthies. A review of Petrov-Galerkin stabilization approaches and an extension to meshfree methods. *Scientific Computing*, 2004.

[48] M. Vogel. Low profile heat sink cooling for next generation CPU thermal designs. *Electronics Cooling*, Feb. 2005.

[49] O. Khonsue. Experimental on the liquid cooling system with thermoelectric for personal computer. *Heat Mass Transfer*, 48:1767–1771, June 2012. doi: 10.1007/s00231-012-1022-x.

[50] A. Cogniso. How an engine cooling system works. Website, May 2016. `http://www.howacarworks.com/basics/how-an-engine-cooling-system-works`.

[51] B. Patel, A. Modi, and P. Rathod. Analysis of engine cooling waterpump of car and significance of its geomtery. *International Journal of Mechanical Engineerging and Technology*, 4(3):100–107, June 2013. ISSN: 09766340.

[52] G. Varoquaux. scipy.optimize.minimize. Website, May 2016. `http://docs.scipy.org/doc/scipy-0.17.0/reference/generated/scipy.optimize.minimize.html`.

[53] J. Bonnans, J. Gilbert, C. Lemaréchal, and C. Sagastizábal. *Numerical Optimization: Theoretical and Practical Aspects*. Springer, 2^{nd} edition, 2006. ISBN: 354035445.

Appendix A

Inclusion Problem Convergence

A.1 Analytic Derivatives of the Displacement Field

A.1.1 Inside the Inclusion

$$\frac{du_x}{dx} = \frac{r_u^2 + \alpha(r_i - r_u)(r_i + r_u)}{r_i^2} \tag{A.1}$$

$$\frac{du_x}{dy} = 0 \tag{A.2}$$

$$\frac{du_y}{dx} = 0 \tag{A.3}$$

$$\frac{du_y}{dy} = \frac{r_u^2 + \alpha(r_i - r_u)(r_i + r_u)}{r_i^2} \tag{A.4}$$

A.1.2 Outside the Inclusion (matrix)

$$\frac{du_x}{dx} = \alpha + \frac{2(-1+\alpha)r_u^2 x^2}{(x^2+y^2)^2} - \frac{(\alpha-1)r_u^2}{x^2+y^2} \tag{A.5}$$

$$\frac{du_x}{dy} = \frac{2(\alpha-1)r_u^2 xy}{(x^2+y^2)^2} \tag{A.6}$$

$$\frac{du_y}{dx} = \frac{2(\alpha-1)r_u^2 xy}{(x^2+y^2)^2} \tag{A.7}$$

$$\frac{du_y}{dy} = \alpha + \frac{2(-1+\alpha)r_u^2 y^2}{(x^2+y^2)^2} - \frac{(\alpha-1)r_u^2}{x^2+y^2} \tag{A.8}$$