Control and optimization of high magnetic fields

Romain Hild

Thèse

INSTITUT DE RECHERCHE MATHÉMATIQUE AVANCÉE

UMR 7501

Strasbourg

présentée pour obtenir le grade de docteur de
l'Université de Strasbourg
Spécialité MATHÉMATIQUES APPLIQUÉES

Romain Hild

Control and Optimization of High Magnetic Fields

Soutenue le 27 novembre 2020
devant la commission d'examen

Frédéric Hecht, rapporteur
Anthony Nouy, rapporteur
Zakaria Belhachmi, examinateur
Yannick Privat, examinateur
Philippe Fazilleau, examinateur
Christophe Prud'homme, directeur de thèse
Christophe Trophime, co-encadrant de thèse, invité

irma.math.unistra.fr

IRMA
Institut de Recherche
Mathématique Avancée

cnrs
dépasser les frontières

Université
de Strasbourg

UNIVERSITÉ DE STRASBOURG

École Doctorale Mathématiques, Sciences de l'Information et de l'Ingénieur
Institut de Recherche Mathématique Avancée, UMR 7501

THÈSE présentée par

Romain HILD

soutenue le 27 novembre 2020

pour obtenir le grade de Docteur de l'Université de Strasbourg
Spécialité : Mathématiques Appliquées

Control and Optimization of High Magentic Fields

THÈSE DIRIGÉE PAR :

Christophe PRUD'HOMME	Directeur de thèse, Université de Strasbourg
Christophe TROPHIME	Co-encadrant de thèse, LNCMI, Grenoble

RAPPORTEURS :

Frédéric HECHT	Professeur, Sorbonne Université
Anthony NOUY	Professeur, École Centrale Nantes

EXAMINATEURS :

Zakaria BELHACHMI	Professeur, Université de Haute-Alsace
Philippe FAZILLEAU	Ingénieur-Chercheur, CEA
Yannick PRIVAT	Professeur, Université de Strasbourg

I... a universe of atoms, an atom in the universe

RICHARD P. FEYNMAN

Remerciements

Je voudrais commencer naturellement par remercier mon directeur de thèse Christophe Prud'homme qui m'a offert l'opportunité de travailler sur ce sujet et qui m'a fait confiance tout au long de ces années. Merci de m'avoir donné la chance de travailler avec quelqu'un d'aussi passionné et d'avoir pu apprendre autant.
Un grand merci également à Christophe Trophime pour m'avoir aidé d'innombrable fois à comprendre le fonctionnement des aimants, ainsi que pour son accueil chaleureux lors de mes passages à Grenoble. Ça a toujours été un plaisir de travailler et de discuter avec toi.

Je remercie Frédéric Hecht et Anthony Nouy d'avoir accepté d'être les rapporteurs de ma thèse. Merci également à Yannick Privat, Zakaria Belhachmi et Philippe Fazilleau d'avoir accepté de faire partie de mon jury.

Merci au personnel du laboratoire de l'IRMA, de l'UFR Math-Info et de l'Université de Strasbourg de faciliter la vie des chercheurs. Merci également à l'équipe MOCO pour m'avoir accueilli et en particulier à l'équipe Cemosis, Vincent H et Alexandre qui m'ont bien aidé à prendre en main `Feel++` à un moment où je n'y comprenais pas grand-chose. Et merci à Lukas, Philippe et François pour les discussions et les quelques sorties ou voyages que l'on a pu partager.
Je tiens aussi à remercier les doctorants: Cécile pour m'avoir aidé à prendre en main le sujet et son accueil à Grenoble, Ranine pour les soirées au Cemracs et les restaus libanais, Fred et Mohamad pour m'avoir supporté et pour les fous rires dans le bureau, Nicolas pour les soirées aux bars et à tous les autres que j'oublis certainement.

Plus particulièrement, un énorme merci à Lorenzo, Céline et Jean-Baptiste pour leur bonne humeur et toutes les discussions plus ou moins sérieuses dans un bureau, lors de voyages ou autour d'un verre. Merci à Vincent C. qui supporte toutes mes questions sur `Feel++` et sans qui il y aurait beaucoup moins de lignes de code dans cette thèse, c'est un plaisir de partager ce bureau avec toi.
Et bien sûr, merci à Guillaume, Laura et Manu qui m'ont aidé à faire passer ces années plus vite. Que ce soit grâce aux débats sans fin sur tout et (surtout) n'importe quoi, aux voyages et aux soirées qui n'en finissent pas, j'ai hâte de pouvoir continuer à partager ces moments avec vous.

Enfin, je tiens à remercier mes proches et ma famille, ma mère, Saïd, ma grand-mère, Antoine, Chrystelle, Arnaud, Laure, Maud, Mike et tous mes neveux et nièces qui me font sourire et rire à chaque fois que je les vois.

Contents

Notations

Methods

FE	:	Finite Element
HDG	:	Hybrid Discontinuous Galerkin
RB	:	Reduced Basis
EIM	:	Empirical Interpolation Method
SER	:	Simultaneous EIM and RB

Functional spaces

Ω	:	Regular bounded domain	$\Omega \subset \mathbb{R}^d,\ d = 1, 2, 3$
$X(\Omega)$	:	Continuous functional space on Ω	
$[X(\Omega)]^v$	:	Vectorial functional space with v components	
$(\cdot, \cdot)_X$	:	Scalar product associated with X	
$\|\cdot\|_X$	:	Norm associated with X	

∇f	:	Gradient of a scalar function f	$\left(\frac{\partial f}{\partial x_i}\right)_{i=1,\dots,d}$
$\nabla \cdot \mathbf{f}$	:	Divergence of a vectorial function $\mathbf{f}$	$\sum_{i=1}^{d} \frac{\partial \mathbf{f}_i}{\partial x_i}$
Δf	:	Laplacian of a scalar function f	$\sum_{i=1}^{d} \frac{\partial^2 f}{\partial^2 x_i}$
$\nabla \times \mathbf{f}$	:	Curl of a vectorial function $\mathbf{f}$	$\frac{\partial \mathbf{f}_2}{\partial x_1} - \frac{\partial \mathbf{f}_3}{\partial x_2}$ if $d = 2$
			$\left(\frac{\partial \mathbf{f}_3}{\partial x_2} - \frac{\partial \mathbf{f}_2}{\partial x_3}, \frac{\partial \mathbf{f}_3}{\partial x_1} - \frac{\partial \mathbf{f}_1}{\partial x_3}, \frac{\partial \mathbf{f}_2}{\partial x_1} - \frac{\partial \mathbf{f}_3}{\partial x_2}\right)^T$ if $d = 3$

$L_2(\Omega)$	:	$\{f \mid \int f^2 < \infty\}$	
$H_1(\Omega)$	:	$\{f \in L_2(\Omega) \mid \nabla f \in [L_2(\Omega)]^d\}$	
$H_{\text{div}}(\Omega)$	:	$\{\mathbf{f} \in [L_2(\Omega)]^d \mid \nabla \cdot \mathbf{f} \in L_2(\Omega)\}$	
$H_{\text{curl}}(\Omega)$	:	$\{\mathbf{f} \in [L_2(\Omega)]^d \mid \nabla \times \mathbf{f} \in [L_2(\Omega)]^v\}$	with $v = 1$ if $d = 2$ or $v = 3$ if $d = 3$

Mesh

d	:	Geometrical dimension d=1,2 or 3
(K, P_K, Σ_K)	:	Finite Element tuple
Ω_h	:	Mesh of characteristic size h

K	:	Geometrical element $K \in \Omega_h$
$\widehat{K}$	:	Reference geometrical element
ϕ_K^{geo}	:	Geometrical transformation $\phi_K^{geo} : \widehat{K} \to K$
$\mathbf{n}$	:	Unit outward normal
$\mathbf{t}$	:	Unit tangent
$\mathcal{E}_h$	:	Set of faces of the geometrical elements
$\mathcal{E}_h^0$	:	Set of interior faces
$\mathcal{E}_h^\partial$	:	Set of boundary faces

Finite Element

N_h	:	Finite element space dimension	
φ_i	:	FE basis function	$i = 1, \dots, N_h$
$X_h(\Omega)$	:	FE approximation space	$X_h = \text{span}(\{\varphi_i\}_i)$
u_h	:	FE approximation of the field u	

Reduced Basis

N	:	Reduced basis space dimension	
ξ_i	:	RB basis function	$i = 1, \dots, N$
$X_{rb}(\Omega)$	:	RB approximation space	$X_{rb} = \text{span}(\{\xi_i\}_i)$
u_{rb}	:	RB approximation of the field u	
$\boldsymbol{\mu}$	:	RB parameter	
$\mathbb{P}$	:	Parameter space	
Ξ_{rb}	:	RB trainset	
M	:	Empirical Interpolation Method dimension	
q_m	:	EIM basis function	$m = 1, \dots, M$
$\mathbb{P}_{EIM}$	:	EIM parameter space	
Ξ_{EIM}	:	EIM trainset	

Magnetostatic

$\mathbf{E}$	:	Electric field	$N \cdot C^{-1}$
$\mathbf{H}$	:	Magnetic field	$A \cdot m^{-1}$
$\mathbf{D}$	:	Electric flux	$C \cdot m^{-2}$
$\mathbf{B}$	:	Magnetic flux	Tesla (T)
$\mathbf{A}$	:	Magnetic vector potential	$V \cdot s \cdot m^{-1}$
V	:	Magnetic scalar potential	Volt (V)
$\mathbf{j}$	:	Current density	$A \cdot m^{-2}$
ρ	:	Electric charge	$C \cdot m^{-3}$

μ	:	Magnetic permeability	$H \cdot m^{-1}$
ε	:	Electric permitivity	$F \cdot m^{-1}$
σ_0	:	Electric conductivity at reference temperature T_0	$S \cdot m^{-1}$
$\sigma(T)$	:	Electric conductivity	$S \cdot m^{-1}$

Elasticity

E	:	Young modulus	Pascal(Pa)	$= kg \cdot m^{-1} \cdot s^{-2}$
ν	:	Poisson's ratio	-	
α_T	:	Coefficient of thermal expansion	K^{-1}	

Heat Transfer

T	:	Temperature	Kelvin
κ_0	:	Thermal conductivity at reference temperature T_0	$W \cdot m^{-1} \cdot K^{-1}$
$\kappa(T)$	:	Thermal conductivity	$W \cdot m^{-1} \cdot K^{-1}$
Nu	:	Nusselt number	-
C_p	:	Thermal capacity	$J \cdot K^{-1}$
ρ_Q	:	Mass density	$kg \cdot m^{-3}$
$\mathbf{j}_Q$	:	Heat flux density	$W \cdot m^{-2}$
L	:	Lorenz number	$W \cdot \Omega \cdot K^{-2}$
α	:	temperature coefficient of resistivity	K^{-1}
h	:	transfer coefficient	$W \cdot m^{-2} \cdot K^{-1}$

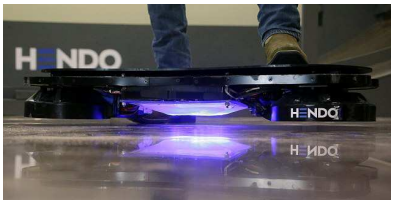
Introduction

Context

The magnetic field is present naturally in our life. For example, the Earth has a magnetic field of about 4.7×10^{-5} Tesla. Even the human brain produces a magnetic field of 10^{-12} T. It is also used in everyday life, doorbells, speakers, motors in a car, or hard drive all use electromagnetism at some point. Other usages include the detection of cancerous cells by the magnetic resonance imaging or MRI, or the cancellation of the gravitation force, allowing to reproduce the conditions of space on Earth in a cheaper way than parabolic flight or the levitation of objects, such as the Maglev train. For this type of application, the magnets are designed to provide a specific profile of a magnetic field. Magnets are also a tool in many research areas, like solid state nuclear magnetic resonance. The more intense the magnetic field is, the higher the precision is.

(a) MRI scanner

(b) Maglev train



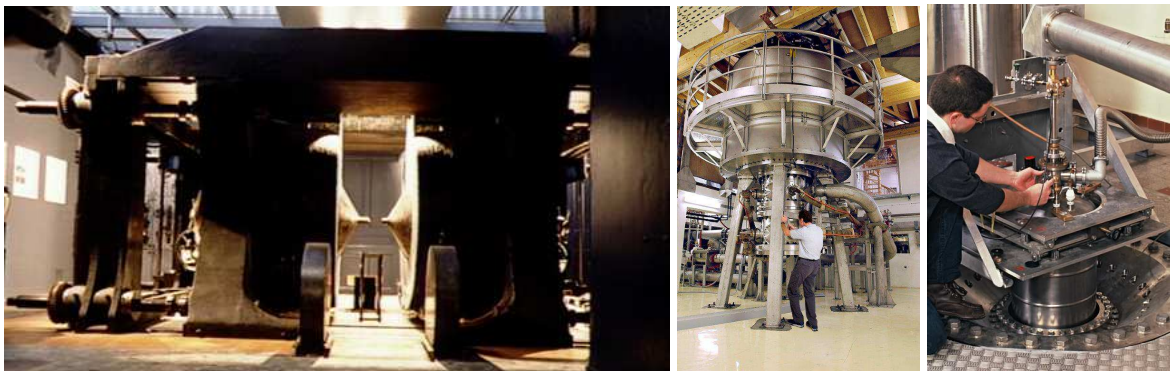
(c) Hendo Hoverboard

Figure 1 – Applications of magnetic field

The development of such magnets for research purpose has started at the beginning of the 20th century. For example, in France, the 120 tons electromagnet of Bellevue, now in a museum, operated from 1920 to 1970, reaching 5 Tesla. More modern techniques have emerged in the 1940s with the works of Francis Bitter and with some improvements on them, see [Montgomery, 1969]. The use of superconducting materials spread in the late 1960s. Superconductors are materials which have zero electrical resistance below a certain temperature. The record intensity using superconducting materials is of 23.5 T, reached at the Ultra-High Field European NMR Center in Lyon, France. Unfortunately, due to the intrinsic limitation of superconductivity, currently above ≈ 24 T we lose superconductivity with standard superconductors materials.

Magnetic fields which intensity is higher than this limit, 24 T, are called *high* magnetic fields. This kind of intensity is usually reached using resistive magnets made of resistive materials, like copper alloy. They require to be water-cooled to keep an

acceptable temperature. This type of magnets are very expensive to build and operate, so only a few laboratories around the world operate them. The magnets are then provided to scientists to perform their experiments. In Europe, they are gathered in the European Magnetic Field Laboratory (EMFL). The French laboratory, the Laboratoire National des Champs Magnétiques Intenses (LNCMI), is split in two sites. One at Toulouse and one at Grenoble. The latter provides a magnetic field of 37 T for several hours.

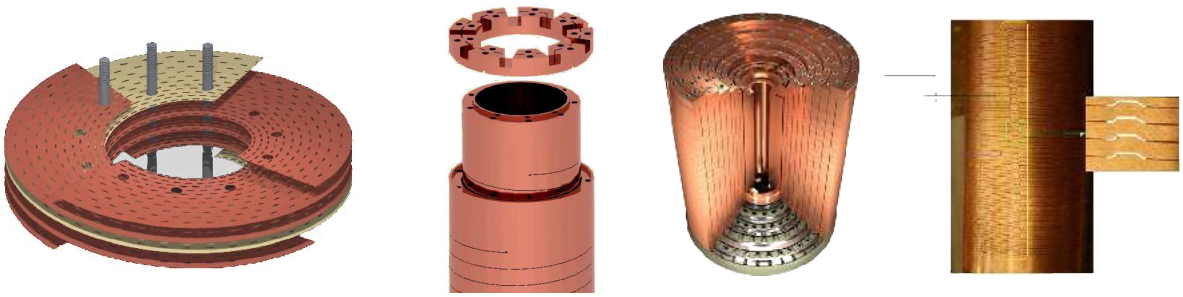


(a) Bellevue magnet (b) Hybrid magnet (c) LNCMI experimental chamber

Figure 2 – The evolution of magnets in time

The design of the magnet plays a great role in the intensity that it can reach. This is why, at the LNCMI, various technologies are tested and combined. The most commonly used type of magnet is the Bitter magnet (see Figure 3a), from the name of its inventor in 1933. It consists of conductive disks, stacked into a solenoid, kept at temperature by passing water into holes made in the disks. The record magnetic field reached using Bitter magnet is 41 T in the United States at the NHMFL [Toth and Bole, 2018]. Another type of magnets is the so-called poly helix magnets (see Figure 3b) described in detail in [Debray et al., 2002] and [Debray et al., 2012]. The magnet is made of several concentric copper alloy tubes, helically cut by spark erosion. They can be cooled either longitudinally, in which case the water circulates only between the tubes, or radially, where the water can also pass between the turns of the tubes. This is possible by the insertion of insulators in the helix cut, a procedure exclusively developed at the LNCMI. Finally, there are hybrid magnets (see Figure 2b) combining an outsert made of superconductor materials and a resistive insert. The current record for these hybrid magnet is 46 T at NHFML. The LNCMI is building a hybrid magnet targeting 42 T.

The international competition to realize magnets capable of reaching the highest possible intensity or the most homogeneous magnetic field to conduct the experiments is tough. Consequently, the LNCMI needs to continually improve the characteristics and the design of its magnets. During the experiments, the electric current can reach 31 kA with a power of 30 MW, necessitating the cooling of the magnet by a water flow of 140 L/s, allowing to evacuating $6kW/m^2$, or approximately $150°C$, and the materials composing the magnet are pushed to 90% of their elasticity limit. All of this enable the



(a) Bitter magnet
(up to 1m in diameter)

(b) Polyhelix magnets
(up to 400mm in diameter)

Figure 3 – Various resistive magnet technologies are used to reach high magnetic fields

LNCMI to reach magnetic fields of up to 43 Tesla, but it also means that extra care is needed to ensure that the magnets are not damaged during the operation, as a resistive magnet can cost up to 300 000 euros and one year to design, provision the material and assemble, whereas a spraconductive magnet can cost up to 3 million euros and up to 10 years to assemble for the Hybrid project developed at Grenoble. So the engineers of the LNCMI have to understand perfectly the process taking place in the magnet, even if the materials used, particular copper alloy to maximize conductivity, come with an uncertainty with regards of their material properties. Also, some scientific experimentations require to have a better homogeneity of the magnetic field than normal in the zone of interest, necessitating specific field profile, and thus specific magnet. To understand the properties or design specific magnets, the numerical simulation plays an essential role as stated in [Trophime et al., 2002]. This is the reason behind the HiFi-Magnet project, which is a collaboration between the LNCMI and the University of Strasbourg. It aims to use the resources of the latter to help the former modeling the physics of a high field magnet.

This thesis, as part of the HiFiMagnet project, continues the work previously done by C. Daversin in [Daversin Catty, 2016] to provide efficient and reliable ways to simulate the 3D multi-physics model on real magnet geometries. Since this model will be used with different types of magnets and/or different experimental parameters, an effort has been made to make it easy to configure. The models and simulations range from mono-physic (e.g. magnetic field computation) to multi-physics (e.g. temperature, electric and magnetic field and displacement computations), for a small part of the magnet and/or the whole magnet. A goal is that engineers at LNCMI might be able to use it through the MSO4SC portal, which provides users a high performance computing (HPC) solution in the cloud and is part of the H2020 European project. During this thesis, we also tried to use the Hybrid Discontinuous Galerkin (HDG) method to better approximate flux variables, such as the electric density or magnetic field. A paper detailing the treatment of the Integral Boundary Condition is in preparation [Guidoboni et al., 2020]. It requires to add another unknown in our equations, which implies a greater cost that we are trying to mitigate by the use of the static condensation technique. The required computational resources may become very important requir-

ing the use of HPC. Lastly, in order to perform optimization problems and sensitivity and uncertainty quantification analysis, we use the Reduced Basis (RB) method, which allows to compute the magnetic field in an efficient way for different parameters.

Feel++

All the methods presented in this thesis have been implemented using the library `Feel++`, for Finite Element Embedded Library in C++. Since the electromagnetism, described by Maxwell, and the other physics involved in a high-field magnet, can be expressed using partial differential equations (PDE), the Finite Element Method (FEM) is well suited to solve this kind of numerical problems.

`Feel++` is an open-source library, which creates a domain specific embedded language (DSEL) aiming to help in the implementation of PDEs with syntax close to the mathematical formulation. It has been described in [Prud'Homme et al., 2012] and can be found at `www.feelpp.org` where various scientific projects are presented.

To efficiently solve the problems, `Feel++` is relying on proven solvers provided by third party libraries like PETSc [Balay et al., 2018], and the meshing of the geometries is outsourced to GMSH [Geuzaine and Remacle, 2009] or Salome [Ribes et al., 2017]. The parallelization is seamless for the user, allowing to run the simulation from a laptop to a cluster without changing the code, a feature needed for the complex geometries of the magnets. `Feel++` already provides a RB framework that we needed to update and extend for our applications. We also had to develop a HDG framework upon the Finite Element implementation of `Feel++` in order to take advantages of this method.

Plan

This manuscript is organised into four parts.

The first part describes the mathematical methods used.
The chapter 1 recalls the theory of the Finite Element Method on which the other developments are based. In this chapter the principle of the FEM is reviewed as well as some families of finite elements convenient for our multi-physics problem, such as Raviart-Thomas and Nedelec.
The chapter 2 aims at detailing the Hybrid Discontinuous Galerkin method. The section 2.1 describes an original contribution, the Integral Boundary Condition, allowing to impose the value of the integral of the normal flux, without knowing neither the value of the potential nor the value of the normal flux. This again is of importance for our thermo-electric problem, since we can modelize the input current setting more closely to the electrical configuration when the magnets are operated.
The chapter 3 details the Reduced Basis method. We focus on the methods dealing with the non-linear and non affinely parametrized problems. We introduce the Empirical Interpolation Method to have an affine decomposition, and the Simultaenous Reduced EIM method to keep a computational cost reasonable.

The second part describes the different physics of our model.
The chapter 4 details the non-linear thermoelectric problem. This problem takes into account the dependence on the temperature of the thermal and electric conductivity to simulate the current flow in the magnet and the increase of temperature resulting from Joule effect.
The chapter 5 enumerates the different resolution strategies for the magnetostatic problem. It shows how to retrieve the magnetic field and potential in the whole domain, including the magnet, using the saddle point or the regularized formulation of the Maxwell equations. Alternatively, we use the Biot & Savart law to compute the magnetic field or potential in a region of interest at the center, but not in the magnet with less computational cost than when using the Maxwell's equations.
Finally the chapter 6 describes the linear elasticity problem. It is used to check if the deformation of the magnet and the stress due to the Lorenz forces and the thermal dilation are not too important.
Each chapter will present the equations related to the physical problem, along with the different formulations of the problem, continuous Galerkin, hybrid discontinuous Galerkin and reduced basis, except for the chapter on Biot & Savart for which there is no HDG formulation.

The third part describes the contribution to the `Feel++` library during this thesis.
The chapter 7 details the HDG implementation of the mixed Poisson and mixed Elasticity problems and the chapter 8 details the implementation of the classes needed for the computation of the magnetic field using the Reduced Method of the Biot & Savart law.

The last part of this manuscript describes the applications of the methods previously presented.
The chapter 9 presents two applications dealing with parameters identification. First, the identification of the water cooling parameters, namely the water temperatures and heat transfer coefficients in each cooling channel of the magnet. Secondly, we reproduce the comissioning of a magnet, identifying two parameters, the field factor and the resistance of the magnet.
Finally, the chapter 10 details geometric optimization and control applications. The first application aims at finding a geometrical configuration to cut the most inner helices to achieve the most homogeneous magnetic field in the region of interest.

Publications

in preparation	*An implementation of HDG methods with Feel++. Application to problems with integral boundary conditions* with C. Prud'homme, L. Sala, S. Bertoluzza, G. Guidoboni, D. Prada, R. Sacco and M. Szopos

2016	ESAIM: Proceedings and Surveys, EDP Sciences, 2016, CEMRACS 2015: Coupling multi-physics models involving fluids, 55, pp.23-40. *Hydromorpho: A coupled model for unsteady Stokes/Exner equations and numerical results with Feel++ library* with C. Prud'homme, N. Aissiouene, T. Amtout, M. Brachet, E. Frénod, A. Rousseau, S. Salmon

Oral Presentations

2019	**Labex IRMIA**, Strasbourg, France *Control and Optimization of high field magnets*
2018	**WCCM**, New-York, USA *High Reynolds Aerothermal Simulations and Reduced Basis*
2018	**CANUM**, Cap d'Agde, France *Optimization and control of magnetic high fields*
2017	**`Feel++` User Days**, Strasbourg, France *HDG Methods in Feel++*
2017	**ANR CHORUS Workshop**, Paris, France *Model Order Reduction for Mutliphysic Problems, Using the Open-Source Framework Feel++*
2016	**CEMRACS**, Marseille, France *Cemracs 2016 : Numerical Challenges in Parallel Scientific Computing*

Poster

2018	**MoRePaS**, Nantes, France *Towards real time computation of 3D magnetic field in parametrized Polyhelix magnets using a reduced basis Biot-Savart model*
2017	**MT25**, Amsterdam, Pays-Bas *Towards real time computation of 3D magnetic field in parametrized Polyhelix magnets using a reduced basis Biot-Savart model*
2016	**Journée Poster de l'école doctorale**, Strasbourg, France *Optimization and control of magnetic high fields*

Part I

Mathematical modeling

Chapter 1

Preliminary Notions

In this chapter, we introduce the basis of the mathematical framework we will work with during this manuscript. We start by recalling some properties of function spaces, in particular the Sobolev spaces. We then introduce the Finite Element Method by exposing its principle and detailing some useful finite element families.

Contents

1.1 Function spaces

1.1.1 Spaces of continuous functions

Let define the operator

$$D^\alpha = \left(\frac{\partial}{\partial x_1}\right)^{\alpha_1} \dots \left(\frac{\partial}{\partial x_n}\right)^{\alpha_n} = \frac{\partial^{|\alpha|}}{\partial x_1 \dots \partial x_n}$$

where α is a multi-index.
Let Ω be an open set in $\mathbb{R}^n$ and let $k \in \mathbb{N}$. We denote by $C^k(\Omega)$ the set of all continuous functions defined on Ω such that $D^\alpha u$ is continuous on Ω for all α with $|\alpha| \leq k$. If Ω is a bounded open set, $C^k(\bar{\Omega})$ can be equipped with the norm

$$\|u\|_{C^k(\bar{\Omega})} := \sum_{\max |\alpha| \leq k} \sup_{x \in \Omega} |D^\alpha u(x)| \quad .$$

The support of a continuous function u defined on an open set $\Omega \subset \mathbb{R}^n$ is defined as the closure of the set $\{x \in \Omega \mid u(x) \neq 0\}$. This is noted supp u.
We denote by $C_0^k(\Omega)$ the set of all u contained in $C^k(\Omega)$ whose support is a bounded subset of Ω. Let

$$C_0^\infty(\Omega) = \cap_{k\geq 0} C_0^k(\Omega) \quad .$$

1.1.2 Spaces of integrable functions

Let p a real number, $p \geq 1$, we denote by $L^p(\Omega)$ the set of all functions defined on an open subset Ω of $\mathbb{R}^n$ such that

$$\int_\Omega |u(x)|^p \mathrm{d}x < \infty \quad .$$

Any two functions which are equal almost everywhere (i.e. equal, except on a set of measure zero) on Ω are identified with each other.
$L^p(\Omega)$ is a Banach space equipped with the norm

$$\|u\|_{L^p(\Omega)} := \left(\int_\Omega |u(x)|^p \mathrm{d}x \right)^{\frac{1}{p}} \quad .$$

An important special case corresponds to $p = 2$, then the space $L^2(\Omega)$ is a Hilbert space with the following norm and inner product

$$\|u\|_{L^2(\Omega)} := \|u\|_{0,\Omega} := \left(\int_\Omega |u(x)|^2 \mathrm{d}x \right)^{\frac{1}{2}} \quad , \quad (u, v) := (u, v)_{0,\Omega)} := \int_\Omega u(x)v(x)\mathrm{d}x \quad .$$

1.1.3 Sobolev spaces

Let us define the concept of weak derivative. Let $u \in C^k(\Omega)$ with Ω an open subset of $\mathbb{R}^n$, and let $v \in C_0^\infty(\Omega)$, then the following integration by parts formula holds:

$$\int_\Omega D^\alpha u(x) \cdot v(x)\mathrm{d}x = (-1)^{|\alpha|} \int_\Omega u(x) \cdot D^\alpha v(x)\mathrm{d}x, \quad |\alpha| \leq k, \quad \forall v \in C_0^\infty(\Omega)$$

Let u be a locally integrable function on Ω (i.e. $u \in L^1(\omega)$ for each bounded open set ω with $\bar{\omega} \subset \Omega$). We call the weak derivative the locally integrable function w_α such that:

$$\int_\Omega w_\alpha(x) \cdot v(x) = (-1)^{|\alpha|} \int_\Omega u(x) \cdot D^\alpha v(x)\mathrm{d}x \quad \forall v \in C_0^\infty(\Omega)$$

The function w_α is called a weak derivative of the function u of order $|\alpha| = \alpha_1 + \cdots + \alpha_n$, and we write $w_\alpha = D^\alpha u$. This function is unique, due to the DuBois Reymond's lemma.

Let $k \geq 0$ an integer and suppose that $p \in [1, \infty]$. We define:

$$W_p^k(\Omega) = \{u \in L^p(\Omega) \mid D^\alpha u \in L^p(\Omega),\ |\alpha| \leq k\}$$

$W_p^k(\Omega)$ is called a Sobolev space of order k.
When $1 < p < \infty$, it can be equipped with the following semi-norm and norm:

$$|u|_{W_p^k(\Omega)} = \left(\sum_{|\alpha|=k} \|D^\alpha u\|_{L^p(\Omega)}^p \right)^{\frac{1}{p}}, \qquad \|u\|_{W_p^k(\Omega)} = \left(\sum_{j=0}^{k} |u|_{W_p^j(\Omega)}^p \right)^{\frac{1}{p}}$$

And when $p = \infty$:

$$|u|_{W^k_\infty(\Omega)} = \sum_{|\alpha|=k} \|D^\alpha u\|_{L_\infty(\Omega)}\,, \qquad \|u\|_{W^k_\infty(\Omega)} = \sum_{j=0}^{k} |u|_{W^j_\infty(\Omega)}$$

An important special case corresponds to $p = 2$. The space $W^k_2(\Omega)$ is a Hilbert space with the inner product:

$$(u, v)_{W^k_2(\Omega)} := \sum_{|\alpha|\le k} (D^\alpha u, D^\alpha v)$$

We then usually write $H^k(\Omega)$ instead of $W^k_2(\Omega)$.
We use frequently $H^1(\Omega)$ and $H^2(\Omega)$.
For p=2, k=1, we consider

$$H^1(\Omega) = \left\{ u \in L^2(\Omega) \mid \frac{\partial u}{\partial x_j} \in L^2(\Omega),\ j = 1, \dots, n \right\} \tag{1.1}$$

$$\|u\|_{H^1(\Omega)} = \|u\|_{1,\Omega} = \left(\|u\|^2_{L^2(\Omega)} + \sum_{j=1}^{n} \left\| \frac{\partial u}{\partial x_j} \right\|^2_{L^2(\Omega)} \right)^{\frac{1}{2}}$$

$$|u|_{H^1(\Omega)} = |u|_{1,\Omega} = \left(\sum_{j=1}^{n} \left\| \frac{\partial u}{\partial x_j} \right\|^2_{L^2(\Omega)} \right)^{\frac{1}{2}}$$

For p=2, k=2, we consider

$$H^2(\Omega) = \left\{ u \in L^2(\Omega) \mid \frac{\partial u}{\partial x_j} \in L^2(\Omega),\ j = 1, \dots, n,\ \frac{\partial^2 u}{\partial x_i \partial x_j} \in L^2(\Omega),\ i, j = 1, \dots, n \right\}$$

$$\|u\|_{H^2(\Omega)} = \left(\|u\|^2_{L^2(\Omega)} + \sum_{j=1}^{n} \left\| \frac{\partial u}{\partial x_j} \right\|^2_{L^2(\Omega)} + \sum_{i,j=1}^{n} \left\| \frac{\partial^2 u}{\partial x_i \partial x_j} \right\|^2_{L^2(\Omega)} \right)^{\frac{1}{2}}$$

$$|u|_{H^2(\Omega)} = \left(\sum_{i,j=1}^{n} \left\| \frac{\partial^2 u}{\partial x_i \partial x_j} \right\|^2_{L^2(\Omega)} \right)^{\frac{1}{2}}$$

We finally define some other useful Sobolev spaces.
Let $H^1_0(\Omega)$ be the closure of $C^\infty_0(\Omega)$ in the norm $\|\cdot\|_{H^1(\Omega)}$. That means $H^1_0(\Omega)$ is the set of all $u \in H^1(\Omega)$ such that u is the limit in $H^1(\Omega)$ of a sequence $\{u_m\}_{m=1}^\infty$ with $u_m \in C^\infty_0(\Omega)$. It can be shown that

$$H^1_0(\Omega) = \{u \in H^1(\Omega) \mid u = 0 \text{ on } \partial\Omega\}$$

This space is a Hilbert space, with the same norm and inner product than $H^1(\Omega)$.

We also define $H^{1/2}(\partial\Omega)$ as the range of the trace operator $tr : H^1(\Omega) \to L^2(\partial\Omega)$:

$$H^{1/2}(\partial\Omega) = \{u \in L^2(\partial\Omega) \mid \exists \tilde{u} \in H^1(\Omega) \text{ such that } u = tr(\tilde{u})\}$$

equipped with the following norm:

$$\|u\|_{H^{1/2}(\partial\Omega)} = \inf_{\substack{\tilde{u} \in H^1(\Omega) \\ tr(\tilde{u}) = u}} \|\tilde{u}\|_{H^1(\Omega)}$$

This Banach space is useful when dealing with boundary conditions.

For the magnetostatic, we will need to integrate terms of divergence or curl of a field. We need to ensure that those terms are square integrable. For such instances, we introduce the two following spaces:

$$H_{\text{div}}(\Omega) = \{\mathbf{u} \in [L^2(\Omega)]^d \mid \nabla \cdot \mathbf{u} \in L^2(\Omega)\} \tag{1.2}$$

$$H_{\text{curl}}(\Omega) = \{\mathbf{u} \in [L^2(\Omega)]^d \mid \nabla \times \mathbf{u} \in [L^2(\Omega)]^v\}, \qquad v = \begin{cases} 1 & \text{if } d = 2 \\ 3 & \text{if } d = 3 \end{cases} \tag{1.3}$$

We shall note that the curl of a 2 dimensional vector is a scalar and the curl of a 3 dimensional vector remains a vector.
Those spaces are Hilbert spaces, with their respective scalar product and norm:

$$(\mathbf{u}, \mathbf{v})_{H_{\text{div}}(\Omega)} = (\mathbf{u}, \mathbf{v})_{L^2(\Omega)} + (\nabla \cdot \mathbf{u}, \nabla \cdot \mathbf{v})_{L^2(\Omega)} \quad , \qquad \|\mathbf{u}\|_{H_{\text{div}}(\Omega)} = \sqrt{(\mathbf{u}, \mathbf{u})_{H_{\text{div}}(\Omega)}} \quad ,$$

$$(\mathbf{u}, \mathbf{v})_{H_{\text{curl}}(\Omega)} = (\mathbf{u}, \mathbf{v})_{L^2(\Omega)} + (\nabla \times \mathbf{u}, \nabla \times \mathbf{v})_{L^2(\Omega)} \quad , \qquad \|\mathbf{u}\|_{H_{\text{curl}}(\Omega)} = \sqrt{(\mathbf{u}, \mathbf{u})_{H_{\text{curl}}(\Omega)}} \quad .$$

1.2 Finite Element Method

Usually, the laws of physics are described in terms of partial differential equations (PDEs). But in general, those equations cannot be solved analytically. We need to use some type of discretizations to approximate the PDEs, and numerically solve the problem. The Finite Element Method allows to compute such approximations

The origin of the method can be traced back to the early 1940s in [Hrennikoff, 1941] and [Courant, 1943] with the mesh discretization and the first apparitions of the elements. The method has grown in popularity later as the use of computers spread, and rigorous basis and further developments were published [Strang and Fix, 1973], [Ciarlet, 1978].

1.2.1 Principles of the method

The first step to apply the finite element method to a partial differential problem is to get a variational formulation of it. It consists in multiplying the PDE by a test function, from a function space to define later, and integrating over the domain. We can view the left-hand side of the variationnal formulation as a bilinear form and the right-hand side as a linear form. The problem is now:

$$\text{Find } u \in V(\Omega) \text{ such that } a(u, v) = l(v) \quad \forall v \in V(\Omega)$$

where $V(\Omega)$ is a Hilbert space of functions defined on Ω equipped with the norm $\|\cdot\|$, a is a bilinear form on $V(\Omega) \times V(\Omega)$ and l a linear form on $V(\Omega)$.

The Lax-Milgram theorem ([Lax and Milgram, 1954]) states that this problem has a unique solution under the following conditions:

- l is continuous : $\exists K$ such that $|l(u)| \leq K \|u\|$, $\forall u \in V(\Omega)$

- a is continuous : $\exists M$ such that $|a(u,v)| \leq M \|u\| \|v\|$, $\forall (u,v) \in V(\Omega) \times V(\Omega)$
- a is coercive : $\exists \alpha > 0$ such that $a(u,u) \geq \alpha \|u\|^2$, $\forall u \in V(\Omega)$

The Babuska-Lax-Milgram ([Babuška, 1971]) extends this results when the test and trial function spaces are different.

To get a numerical approximation of u, we need to look for a solution in $V_h \subset V$ of finite dimension. An approximation u_h of u can be defined by the problem

$$\text{Find } u_h \in V_h \text{ such that } a(u_h, v_h) = l(v_h) \quad \forall v_h \in V_h$$

We can build a sequence of subsets $(V_h)_h$ such that for all element φ of V, there exists a sequence of $\varphi_h \in V_h$ such that $\|\varphi - \varphi_h\| \to 0$ when $h \to 0$. This is called a Galerkin method.

For any $v_h \in V_h \subset V$ we have $a(u, v_h) = l(v_h)$ and $a(u_h, v_h) = l(v_h)$ so $a(u - u_h, v_h) = 0$. From this, we can show that

$$a(u - u_h, u - u_h) = a(u - u_h, u - v_h) \quad \forall v_h \in V_h$$

But since a is coercive and continuous,

$$\alpha \|u - u_h\|^2 \leq a(u - u_h, u - u_h) = a(u - u_h, u - v_h) \leq M \|u - u_h\| \|u - v_h\|$$

or

$$\|u - u_h\| \leq \frac{M}{\alpha} \|u - v_h\| \quad \forall v_h \in V_h \quad ,$$

which implies

$$\|u - u_h\| \leq \frac{M}{\alpha} d(u, V_h) \quad .$$

where $d(u,V) = \min_{v \in V_h} \|u - v\|$. This inequality is called Céa's lemma.

Since V_h is of finite dimension N_h, it admits a basis $(\varphi_1, \ldots, \varphi_{N_h})$. u_h can be written as

$$u_h = \sum_{i=1}^{N_h} U_i \varphi_i \quad .$$

We can then rewrite the variational problem as

$$\text{Find } U = (U_1, \ldots, U_{N_h}) \text{ such that } \sum_{i=1}^{N_h} U_i a(\varphi_i, \varphi_j) = l(\varphi_j), \quad \forall j = 1, \ldots, N_h \tag{1.4}$$

That is equivalent to the linear system:

$$\begin{pmatrix} a(\varphi_1, \varphi_1) & \ldots & a(\varphi_1, \varphi_{N_h}) \\ \vdots & & \vdots \\ a(\varphi_{N_h}, \varphi_1) & \ldots & a(\varphi_{N_h}, \varphi_{N_h}) \end{pmatrix} \begin{pmatrix} U_1 \\ \vdots \\ U_{N_h} \end{pmatrix} = \begin{pmatrix} l(\varphi_1) \\ \vdots \\ l(\varphi_{N_h}) \end{pmatrix} \tag{1.5}$$

or

$$AU = b$$

1.2.2 Finite elements family

The matrix A could be dense, but to be efficient, we want this matrix to be as sparse as possible. For this, we need to define the functions φ_i with a support as small as possible.
The first step to define the finite element used is to partition the domain Ω into a collection of non-empty and disjoint open simplices or hypercubes, $\{K_e\}_{e=1,\dots,N_e}$. In the case of simplices, K_e can be a line, a triangle, or a tetrahedron, and in the case of hypercubes, it can be a line, a quadrangle, or a hexaedrons. The characteristic size of those is h, thus, we call Ω_h the mesh corresponding to Ω. We call $\mathcal{T}_h$ the set of all domains K_e.
The finite element method defines this basis [Ciarlet, 1978], and so the space V_h, from a tuple (K, P_K, Σ_K).

- K is the geometrical domain, simplex or hypercube used to create Ω_h.
- P_K is a polynomial space of finite dimension k defined on K.
- Σ_K is a set of linear functionnals $\{\sigma_i : P_K \to \mathbb{R}\}_{i=1}^k$

The k local basis functions of P_K, $\{\phi_j\}_{j=1}^k$, are defined such that $\sigma_i(\phi_j) = \delta_{ij}$, $1 \leq i, j \leq k$. The global finite element space on Ω_h, V_h, is derived from the local space. Every function of the global space is defined by having its restrictions to each K_e belonging to the local finite element space associated, and by adding the required continuity conditions between each cell.

Each finite element $(K_e, P_{K_e}, \Sigma_{K_e})$ involves its proper basis functions. The right way to handle this is to choose a reference finite element $(\hat{K}, P_{\hat{K}}, \Sigma_{\hat{K}})$. Then each element K_e of the mesh Ω_h is supposed to be the image of $\hat{K}$ from a C^{geo}-diffeomorphism $\phi_{K_e}^{geo}$, where *geo* is the geometrical order of the mesh.
In this way, many computations can be performed once on the reference element, and all we need is the Jacobian of the transformation $J_{K_e}^{geo}$ and its inverse to deal with the integrals.

Depending on the problem, the space V_h needs to have certain properties. Indeed, the solution can belong to H^1, H_{div} or H_{curl}, defined in section 1.1.3. Each space has an interpolation operator defining its discrete version and the finite elements associated.

Those spaces are related to each other by the De Rham complex diagram (1.6) [Boffi et al., 2013], where *the range of each of the operators coincides with the null space of the operator in the sequence, and the last map is a surjection*. We also show the discrete version of the diagram, where the discrete spaces are obtained from the continuous one by an interpolant $\pi_h : X \to X_h$.

$$\begin{array}{ccccccccc} \mathbb{R} & \xrightarrow{id} & H^1(\Omega) & \xrightarrow{\nabla} & H_{curl}(\Omega) & \xrightarrow{\nabla\times} & H_{div}(\Omega) & \xrightarrow{\nabla\cdot} & L^2(\Omega) \\ & & \downarrow \pi_h^U & & \downarrow \pi_h^V & & \downarrow \pi_h^W & & \downarrow \pi_h^Z \\ \mathbb{R} & \xrightarrow{id} & U_h & \xrightarrow{\nabla_h} & V_h & \xrightarrow{\nabla_h\times} & W_h & \xrightarrow{\nabla_h\cdot} & Z_h \end{array} \tag{1.6}$$

We will now define the Lagrange finite elements that are conforming to the space $H^1(\Omega)$, and so define U_h in 1.2.2.1. Then, in 1.2.2.2, we define the Raviart-Thomas finite

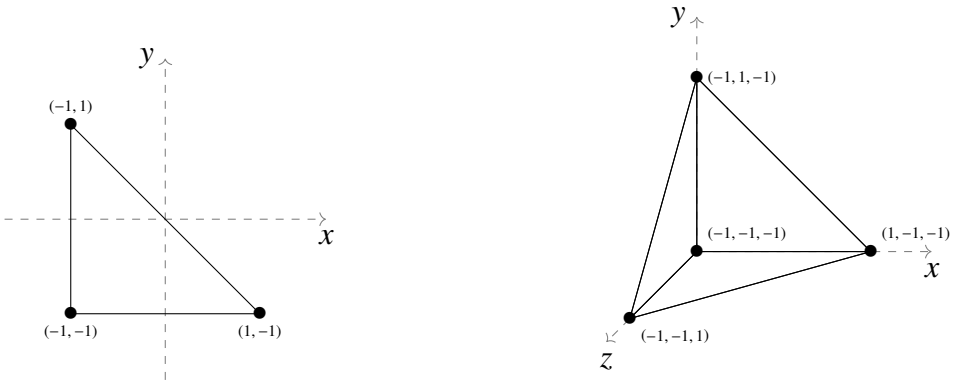
elements, conforming to the space $H_{div}(\Omega)$, and defining the space W_h. And in 1.2.2.3, we define the Nedelec finite elements, conforming to $H_{curl}(\Omega)$ and defining the space V_h.

1.2.2.1 Lagrange finite elements

The Lagrange finite element are one of the most used family of finite element. Indeed, it is suitable to solve problems whose solution resides in H^1 Hilbert space (1.1), and because they allow an easy visualisation of the fields. We recall that the Lagrange finite elements are defined as tuples (K, P_K, Σ_K). The polynomial space P_K of order k is the set of polynomials of degree less than k, $\mathbb{P}^k$. The set of linear functional Σ_K, which are the degrees of freedom, are the evaluation of the polynomials $p \in \mathbb{P}^k$ at the interpolation points of K, denoted $\mathbf{d}_i$:

$$\sigma_i(p) = p(\mathbf{d}_i), \quad \forall p \in \mathbb{P}^k$$

This allow to deduce the Lagrange basis functions.

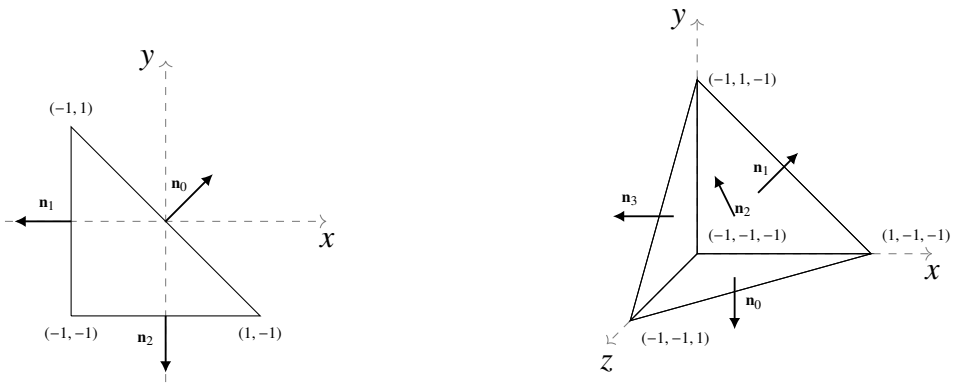


(a) 2D simplex (b) 3D simplex

Figure 1.1 – Reference elements

1.2.2.2 Raviart-Thomas finite elements

For problems involving the integral of divergence terms, we need to ensure that those terms are square integrable. That is the discrete functions are in H_{div}, (1.2). For this to be the case, in particular, the normal component of the solution across the interfaces of the elements of the mesh Ω_h must be continuous. Several divergence conforming finite elements have been proposed using different bases, such as Brezzi-Douglas-Marini (BDM) [Brezzi et al., 1985] or Raviart-Thomas [Raviart and Thomas, 1977], using different polynomial spaces, $[\mathbb{P}^k]^d$ or a subspace of $[\mathbb{P}^{k+1}]^d$. They have been both extended in 3D by Nédélec. We will detail the Raviart-Thomas finite elements.



(a) 2D simplex (b) 3D simplex

Figure 1.2 – Reference elements - normals

Following Ciarlet's formalism, we need to define P_K and Σ_K. The function space P_K is $\mathcal{D}_k$ the vectorial subspace of $[\mathbb{P}^{k+1}]^d$ of order k defined as:

$$\mathcal{D}_k = [\mathbb{P}^k]^d \oplus \mathbf{x}\left(\mathbb{P}^k \ominus \mathbb{P}^{k-1}\right)$$

We can split the set of degrees of freedom Σ_K as two sets of linear fonctionnals. We can use nodal or modal basis functions, in the latter case, at the lowest order zero, we define the faces degrees of freedom $\{\sigma^f\}_{f\in\mathcal{E}_h}$ and from the order one, we also use the inner degrees of freedom $\{\sigma^K\}_{K\in\mathcal{T}_h}$:

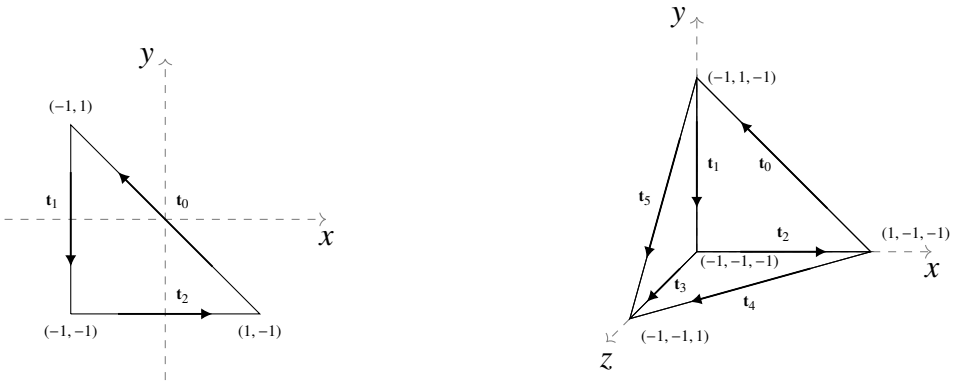
$$\sigma^f(\mathbf{u}) = \int_f \mathbf{u}\cdot\mathbf{n}p \qquad \forall p \in \mathbb{P}^k(f)$$

$$\sigma^K(\mathbf{u}) = \int_K \mathbf{u}\cdot\mathbf{q} \qquad \forall \mathbf{q} \in [\mathbb{P}^{k-1}(K)]^d$$

Remark 1. The orientation of the normal is important for the definition of the face dofs. We need to ensure the unicity of the normal across elements.

1.2.2.3 Nedelec finite elements

In the same way as for the H_{div} conforming elements, we may need to use elements that ensure that the curl of a discrete function is integrable. Similar to the H_{div} conforming elements, H_{curl} conforming elements need to ensure the continuity of the tangential component of the solution. Such elements have been introduced by J.C. Nédélec in the 80s [Nédélec, 1980], [Nédélec, 1986]. They are thus called the Nédélec finite elements, and group two elements types, similar in construction to the Raviart-Thomas elements and the Brezzi-Douglas-Marini elements. We will detail the first kind of the Nédélec finite elements.



(a) 2D simplex (b) 3D simplex

Figure 1.3 – Reference elements - tangents

Following Ciarlet's formalism, we need to define P_K and Σ_K. The function space P_K is $\mathcal{R}^{k,1}$ a vectorial subspace of $[\mathbb{P}^{k+1}]^d$ of order k defined as:

$$\mathcal{R}^{k,1} = [\mathbb{P}^k]^d \oplus \mathcal{S}^k$$

where $\mathcal{S}^k$ is defined as:

$$\mathcal{S}^k = \{\mathbf{u} \in [\mathbb{P}^k]^d \mid \mathbf{u} \cdot \hat{\mathbf{x}} = 0\}$$

where $\hat{\mathbf{x}} \in \hat{K}$ is in the reference element.

As with H_{div} conforming elements, we can use nodal or modal function basis, the latter is detailed here. We split the set of degrees of freedom Σ_K as three sets of linear functionnal. At the lowest order, we have the edge degrees of freedom $\{\sigma^e\}_{e\in\mathcal{D}_h}$ and for highest order we add the set of faces degrees of freedom $\{\sigma^f\}_{f\in\mathcal{E}_h}$ and the set of inner degrees of freedom $\{\sigma^K\}_{K\in\mathcal{T}_h}$.

In 2D, the edges are combined with the faces, thus we have that $\sigma_e \equiv \sigma_f$, and:

$$\begin{aligned}
\sigma^e(\mathbf{u}) &= \int_e (\mathbf{u}\cdot\mathbf{t})p && \forall p \in \mathbb{P}^k(e)\\
\sigma^K(\mathbf{u}) &= \int_K \mathbf{u}\cdot\mathbf{q} && \forall q \in [\mathbb{P}^{k-1}(K)]^2
\end{aligned}$$

In 3D, we have:

$$\begin{aligned}
\sigma^e(\mathbf{u}) &= \int_e (\mathbf{u}\cdot\mathbf{t})p && \forall p \in \mathbb{P}^k(e)\\
\sigma^f(\mathbf{u}) &= \int_f (\mathbf{u}\times\mathbf{n})\cdot\mathbf{q} && \forall \mathbf{q} \in [\mathbb{P}^{k-1}(f)]^3\\
\sigma^K(\mathbf{u}) &= \int_K \mathbf{u}\cdot\mathbf{q} && \forall \mathbf{q} \in [\mathbb{P}^{k-2}(f)]^3
\end{aligned}$$

Remark 2. The orientation of the tangent is important for the definition of the face dofs in 3D. We need to ensure the unicity of the tangent across elements.

1.2.3 Continuous and Discontinuous Galerkin

The Continuous Galerkin (CG) implies that the function v_H is continuous across the edges of the simplicies K:

$$V = \{v_h \in C^0(\Omega_h) \,|\, v_{h|K} \in P_K(K)\}$$

A different class of Finite Element methods uses discontinuous functions, thus it is called Discontinuous Galerkin (DG). These methods allow more flexibility in regards of local changes in each element. For example, using different polynomial order in different elements of the mesh becomes possible, as well as using non-conforming mesh and $h-p$ adaptivity. Since the elements only communicate with their immediate neighbour, the resulting matrices are easily parallelized ([Remacle et al., 2003]). This class of methods has been introduced in [Reed and Hill, 1973] for time independent linear hyperbolic equations. It has then been developed for non-linear time dependent problems in [Cockburn and Shu, 1991, Cockburn and Shu, 1989, Cockburn et al., 1989, Cockburn et al., 1990, Cockburn and Shu, 1998]. More details and recent works can be found in [Cockburn et al., 2011].
A major inconvenience of Discontinuous Galerkin is that due to the discontinuity between elements, for the same vertex in the mesh, we have different degrees of freedom in each cell touching this vertex. This leads to a higher number of degrees of freedom than in the CG methods, and so a much higher computational cost for the same number of elements. We will see in the next chapter a method to reduce this cost.

Chapter 2

Hybrid Discontinuous Galerkin

The discontinuous Galerkin (DG) finite element methods are attractive for many scientists due to inherent advantages. Indeed DG methods can handle any type of mesh, are well suited for *hp*-adaptivity [Berger and Colella, 1989, Bey et al., 1996], allow boundary conditions to be simply imposed and are easy to parallelize [Baggag et al., 1999]. But the considerable increase in number of degrees of freedom needed by these methods makes its computational cost very high in comparison to the continuous Galerkin methods (see Table 2.3 and 2.4 for comparison). The introduction of DG methods capable of taking advantages of the optimization technique called *static condensation* in [Cockburn et al., 2009b] offers a solution to this issue.
In problems involving a primal and a flux variable, such as potential and gradient or curl, HDG methods reduce the number of degrees of freedom by introducing a new variable, which is the numerical trace of the primal variable. Thanks to static condensation, the primal and flux variable can be expressed in terms of this variable. This allows to compute the trace by solving a global problem but only on the face of the elements. Then the primal and flux variables are recovered by solving local problems on each element. Since the trace is defined only on the element boundaries, the computational cost of the global problem is low in comparison to other DG methods.
In addition, for certain types of problems, such as convection-diffusion problems, the HDG methods have optimal convergence for the approximation of the flux. Local post processing can also be performed at the element level to achieve an even better convergence of the solution or obtain a new approximation with conservative properties with the flux in H_{div} [Cockburn et al., 2009c]. Since we use discontinuous approximation, this method is also well suited for mortaring techniques [Barrenechea et al., 2018]. Because the many local problems to solve are defined only by elements, those methods remain highly parallelizable.

Hybrid mixed methods aim at solving a problem by splitting it in many local problems, one for each element of the mesh Ω_h, with transmission conditions between them determined by solving a single global problem. To obtain a hybrid mixed method, we need to write the problem as a system of first order equations in terms of discontinuous approximations for both the flux variable, the primal variable and their trace. Hybrid Discontinuous Galerkin methods have been developed for Stokes and Navier-

Stokes flow [Nguyen et al., 2010, Nguyen et al., 2011], linear and non-linear elasticity [Kabaria et al., 2014, Qiu et al., 2013] and more recently for Maxwell equations [Lu et al., 2015, Chen et al., 2017]. Those equations can be used to describe the physics of the cooling, deformation, and magnetic forces of the magnet. The formulation for second order elliptic problems, developed by [Cockburn et al., 2009b] is suitable for both the electric and thermic problems. It will be described in the following section, with the addition of the *Integral Boundary Condition*. We developed a method to deal with this type of boundary conditions, described in [Guidoboni et al., 2020], that allows us to be as close as possible from the modeling view point of the experimental procedure at the LNCMI, see chapter 4.

The static condensation, a method to reduce the cost of solving such discontinuous problems, will be detailed in section 2.2. Finally, we will present the performances of this method compared to a classic Continuous Galerkin formulation in section 2.3.

Contents

2.1 Integral Boundary Condition

Complex systems, such as the one modeling the thermoelectric problem at the LNCMI, give rise to physical properties that do not fall into standard boundary conditions like Dirichlet, Neumann or Robin. The Integral Boundary Condition (IBC) can help model a problem where we cannot impose directly the value of a field p or of its normal flux $\nabla p \cdot \mathbf{n}$ on a surface. Instead, we want the field p to be constant on the surface with an unknown value and we want the integral over the surface of the normal flux to be equal to a given value. It is especially helpful when it is not possible to experimentally access the pointwise value of p on a surface but the global flux across this surface is a given design target.
An Integral Boundary Condition (IBC) on Γ_I is a condition of the type:

$$\int_{\Gamma_I} \mathbf{u} \cdot \mathbf{n} = g_I \text{ and } p \text{ is an unknown constant on } \Gamma_I \text{ such that } |\Gamma_I| p - \int_{\Gamma_I} p = 0$$

We will describe the method to deal with such condition in a mixed poisson problem.

We want to find p such that:

$$-\nabla \cdot (\kappa \nabla p) = f \tag{2.1}$$

In order to use HDG, we first need to write this equation as a system of first order equations involving the primal variable p and its flux $\mathbf{u}$:
Find $\mathbf{u}$, p, such that:

$$\begin{cases} \mathbf{u} + \kappa \nabla p = 0 \text{ in } \Omega & \text{(2.2a)} \\ \nabla \cdot \mathbf{u} = f \text{ in } \Omega & \text{(2.2b)} \end{cases}$$

With Dirichlet, Neumann and IBC conditions on Γ_D, Γ_N and Γ_I respectively:

$$p = g_D \quad \text{on } \Gamma_D \tag{2.2c}$$

$$\mathbf{u} \cdot \mathbf{n} = g_N \quad \text{on } \Gamma_N \tag{2.2d}$$

$$\int_{\Gamma_I} \mathbf{u} \cdot \mathbf{n} = g_I \quad \text{on } \Gamma_I \tag{2.2e}$$

$$|\Gamma_I| p - \int_{\Gamma_I} p = 0 \quad \text{on } \Gamma_I \tag{2.2f}$$

where $\partial\Omega = \Gamma = \Gamma_D \cup \Gamma_N \cup \Gamma_I$.

We will describe the variational formulation of the problem, but we first need to introduce a fixed function $\bar{\varphi} \in H^{1/2}(\Gamma)$ verifying

$$\bar{\varphi}_{|\Gamma_D} = g_D, \text{ and } \bar{\varphi}_{|\Gamma_I} = 1$$

and the space $H^{1/2}_{00}(\Gamma_N)$ defined as

$$H^{1/2}_{00}(\Gamma_N) = \{\varphi \in H^{1/2}(\Gamma) \mid \varphi = 0 \text{ on } \Gamma_D \cup \Gamma_I\}$$

as well as

$$\mathbb{M} = \text{span} < \bar{\varphi} > \oplus H^{1/2}_{00}(\Gamma_N)$$

Then we look for $\mathbf{u} \in H_{\text{div}}(\Omega)$, $p \in L^2(\Omega)$, and $\hat{p} \in \mathbb{M}$ such that for all $\mathbf{v} \in H_{\text{div}}(\Omega)$, $q \in L^2(\Omega)$ and $\mu \in \mathbb{M}$ we have

$$(\kappa^{-1}\mathbf{u}, \mathbf{v})_\Omega - (p, \nabla \cdot \mathbf{v})_\Omega + \langle \hat{p}, \mathbf{v} \cdot \mathbf{n} \rangle_\Gamma = 0 \tag{2.3a}$$

$$(\nabla \cdot \mathbf{u}, q)_\Omega = (f, q)_\Omega \tag{2.3b}$$

$$\langle \mathbf{u} \cdot \mathbf{n}, \mu \rangle_\Gamma = \langle g_N, \mu \rangle_{\Gamma_N} + g_I |\Gamma_I|^{-1} \langle \mu, 1 \rangle_{\Gamma_I} \tag{2.3c}$$

In order to prove that problem 2.3 admits a unique solution, we start by observing that since the normal trace operator from $H_{\text{div}}(\Omega)$ to $H^{1/2}(\Gamma)$ admits a bounded right inverse, the following inf-sup condition holds:

$$\inf_{\phi \in \mathbb{M}} \sup_{\mathbf{u} \in H_{\text{div}}(\Omega)} \frac{\langle \mathbf{u} \cdot \mathbf{n}, \phi \rangle_\Gamma}{\|\mathbf{u}\|_{H_{\text{div}}(\Omega)} \|\phi\|_{H^{1/2}(\Gamma)}} \gtrsim 1 \tag{2.4}$$

We will first prove the following lemma:

Lemma 1. *Setting*

$$\mathbb{Z} = \{\mathbf{u} \in H_{\text{div}}(\Omega) \mid \langle \mathbf{u} \cdot \mathbf{n}, \phi \rangle_\Gamma = 0 \quad \forall \phi \in \mathbb{M}\}$$

an inf-sup condition of the form

$$\inf_{p \in L^2(\Omega)} \sup_{\mathbf{u} \in \mathbb{Z}} \frac{(p, \nabla \cdot \mathbf{u})_\Omega}{\|p\|_{0,\Omega} \|\mathbf{u}\|_{H_{\text{div}}(\Omega)}} \gtrsim 1 \tag{2.5}$$

holds.

Proof. In order to prove the Lemma, for any given $p \in L^2(\Omega)$ we need to find $\mathbf{u} \in \mathbb{Z}$ such that

$$(p, \nabla \cdot \mathbf{u})_\Omega \gtrsim \|p\|_{0,\Omega} \|\mathbf{u}\|_{H_{\text{div}}(\Omega)}$$

Let then $p \in L^2(\Omega)$ be given and let $\hat{g} \in L^2(\Gamma)$ be some function satisfying $\int_{\Gamma_I} \hat{g} = \int_\Omega p$ and $\hat{g} = 0$ on $\Gamma \setminus \Gamma_I$. Letting $\phi \in H^1(\Omega)$ be the zero mean solution of

$$\nabla \cdot \kappa \nabla \phi = p \text{ in } \Omega, \quad \kappa \nabla \phi \cdot \mathbf{n} = \hat{g} \text{ on } \Gamma,$$

it is not difficult to verify that $\mathbf{u} = -\kappa \nabla \phi$, belongs to $\mathbb{Z}$. Moreover

$$(p, \nabla \cdot \mathbf{u})_\Omega = \int_\Omega |p|^2$$

The inf-sup bound 2.5 follows by observing that

$$\|\mathbf{u}\|_{H_{\text{div}}(\Omega)} \lesssim \|\phi\|_{1,\Omega} \lesssim \|p\|_{H^1(\Omega)'} + \|\hat{g}\|_{H^{1/2}(\Gamma)'} \lesssim \|p\|_{0,\Omega}$$

□

Thanks to 2.4 and 2.5, we can now prove the following theorem

Theorem 1. *For $f \in L^2(\Omega)$, $g_N \in L^2(\Gamma_N)$, the problem 2.3 admits a unique solution $(\mathbf{u}, p, \hat{p})$ satisfying*

$$-\nabla \cdot \kappa \nabla p = f \text{ in } \Omega, , \quad p = 0 \text{ on } \Gamma_D, \quad -\kappa \nabla p \cdot \mathbf{n} = g_N \text{ on } \Gamma_N, \tag{2.6}$$

as well as the integral boundary condition

$$|\Gamma_I| p - \int_{\Gamma_I} p = 0 \text{ on } \Gamma_I \tag{2.7}$$

Moreover, if g_N satisfies suitable compatibility conditions, the non standard boundary condition

$$\int_{\Gamma_I} -\kappa \nabla p \cdot \mathbf{n} = g_I \tag{2.8}$$

is satisfied.

Proof. In view of [Nicolaides, 1982], the two inf-sup conditions (2.4) and (2.5), together with the continuity of bilinear forms on the left-hand side of (2.3) and of the linear operators on the right-hand side of (2.3), imply existence, uniqueness and stability of the solution. By standard arguments it is then not difficult to verify that (2.6) holds and that $p = \hat{p}$ on Γ_I. Condition (2.7) will be encoded directly in the choice of the discretization space. We then only need to prove that (2.8) holds. In order to do so, we need to give a rigorous meaning to the left-hand side of such an equality. We start by observing that p solves the equation (2.6) with additional boundary condition $p = \hat{p}$ on Γ_I. This is a standard mixed Dirichlet-Neumann boundary condition, and we know (see *e.g.* [Grisvard, 1985]) that its solution verifies $p \in W^{1,s}(\Omega)$ for some $s > 2$. This, in turn, implies that $\mathbf{u}\cdot\mathbf{n} = -\kappa\nabla p\cdot\mathbf{n} \in W_{s'}^{-1/2}$ with $s' < 2$. Since the function $\bar{\phi}_{ibc}$ is identically equal to one on Γ_I and to zero on $\Gamma \setminus \Gamma_I$:

$$\bar{\phi}_{ibc} = \begin{cases} 1 & \text{on } \Gamma_I \\ 0 & \text{on } \Gamma \setminus \Gamma_I \end{cases}$$

verifies $\bar{\phi}_{ibc} \in W^{1/2,s'}$, it makes sense to write the integral of the right-hand side of (2.8). Remark that $\bar{\phi}_{ibc} \notin H^{1/2}(\Gamma)$, so that we would not be able to give a sense to such an integral if we only knew that $\mathbf{u} \in H_{\text{div}}(\Omega)$. Let now $\bar{\phi}_0 = \bar{\phi} - \bar{\phi}_{ibc}$. We have

$$\int_{\Gamma_I} \mathbf{u}\cdot\mathbf{n} = \int_{\Gamma} (\mathbf{u}\cdot\mathbf{n})\bar{\phi}_{ibc} = \int_{\Gamma} (\mathbf{u}\cdot\mathbf{n})\bar{\phi} - (\mathbf{u}\cdot\mathbf{n})\bar{\phi}_0 = \int_{\Gamma_N} g_N\bar{\phi} + g_I - \int_{\Gamma} (\mathbf{u}\cdot\mathbf{n})\bar{\phi}_0 = g_I$$

where the last equation comes from the observation that $\text{supp}(\bar{\phi}_0) = \Gamma_N$ and that $\bar{\phi}_0 = \bar{\phi}$ on its support. □

2.1.1 Notations

Let Ω_h be the mesh approximating Ω. It is a set of elements K. We define an interior face as $F = \partial K^+ \cap \partial K^-$ for K^+ and K^- two adjacent elements. A boundary face is a face as $F = \partial K \cap \Gamma$. The set of interior and boundary faces are noted respectively $\mathcal{E}_h^0$ and $\mathcal{E}_h^\partial$. The set of all faces is the union of those two sets, and is noted $\mathcal{E}_h$. We will also need the set $\mathcal{E}_h^I = \mathcal{E}_h^\partial \setminus \Gamma_I$.

We introduce the following spaces:

$$\mathbf{V}_h = \{\mathbf{v} : \Omega_h \to \mathbb{R}^d \,|\, \mathbf{v}_{|K} \in \mathbf{V}^k(K) \quad \forall K \in \Omega_h\}, \tag{2.9a}$$

$$W_h = \{w : \Omega_h \to \mathbb{R} \,|\, w_{|K} \in W^k(K) \quad \forall K \in \Omega_h\}, \tag{2.9b}$$

$$M_h = \{\mu : \mathcal{E}_h^I \to \mathbb{R} \,|\, \mu_{|F} \in M^k(F) \quad \forall F \in \mathcal{E}_h^I\}, \tag{2.9c}$$

$$C_h = \{m \in C^0(\Gamma_I) \,|\, m_{|F} \in P^0(F) \quad \forall F \in \Gamma_I\} \tag{2.9d}$$

Setting the trace $\hat{\mathbf{u}}$ of $\mathbf{u}$ and defining the local spaces $\mathbf{V}^k(K)$, $W^k(K)$ and $M^k(F)$ determines the kind of hybrid method which is used. In the following, we will use:

- For $k \geq 0$:

$$\mathbf{V}^k(K) = [P^k(K)]^d, \quad W^k(K) = P^k(K) \tag{2.10}$$

$$M^k(F) = \{\mu \in L^2(\mathcal{E}_h) \,|\, \mu_{|F} \in P^k(F) \quad \forall F \in \mathcal{E}_h\} \tag{2.11}$$

- For each $K \in \Omega_h$, the numerical normal flux:

$$\hat{\mathbf{u}}_h \cdot \mathbf{n} = \mathbf{u}_h \cdot \mathbf{n} + \tau_K(p_h - \hat{p}_h - \lambda_h) \text{ on } \partial K \tag{2.12}$$

  where $\hat{p}_h \in M_h^k$ is the trace of p_h on all the faces except on Γ_I, where the trace of p_h is represented by $\lambda_h \in C_h$.

Remark 3. The numerical normal flux (2.12) is characteristic of a particular class of HDG methods, the so-called Local Discontinuous Galerkin Hybridizable (LDG-H) methods proposed and investigated in a series of seminal papers [Cockburn et al., 2008, Cockburn et al., 2009b, Cockburn et al., 2009c, Cockburn et al., 2009a] The quantity τ_K is a nonnegative stabilization parameter which can have different values on each face $F \in \partial K$ depending on the mesh element K.

We define also the *jump* of the normal component of a vectorial discontinuous function $\mathbf{v}$ by:

$$[[\mathbf{v}]]_F = \mathbf{v}_{K^+} \cdot \mathbf{n}_{K^+} + \mathbf{v}_{K^-} \cdot \mathbf{n}_{K^-}$$

where $\mathbf{n}_K$ is the unit outward normal of K.

2.1.2 Formulation

In [Arnold and Brezzi, 1985], it was proven that $\hat{p}_h$ can be viewed as a Lagrange multiplier enforcing the continuity condition across each face. In the same way, λ_h is a Lagrange multiplier enforcing the integral boundary condition.
This gives us the following variational formulation:
Find $(\mathbf{u}_h, p_h, \hat{p}_h, \lambda_h) \in \mathbf{V}_h^k \times W_h^k \times M_h^k \times C_h$ such that:

$$(\kappa^{-1}\mathbf{u}_h, \mathbf{v}_h)_\Omega - (p_h, \nabla \cdot \mathbf{v}_h)_\Omega + \langle \hat{p}_h, \mathbf{v}_h \cdot \mathbf{n}\rangle_{\mathcal{E}_h^I} + \langle \lambda_h, \mathbf{v}_h \cdot \mathbf{n}\rangle_{\Gamma_I} = 0 \tag{2.13a}$$

$$(\nabla \cdot \mathbf{u}_h, q_h)_\Omega + \langle \tau_K p_h, q_h\rangle_{\mathcal{E}_h} - \langle \tau_K \hat{p}_h, q_h\rangle_{\mathcal{E}_h^I} - \langle \lambda_h, \tau_K q_h\rangle_{\Gamma_I} = (f, q_h)_\Omega \tag{2.13b}$$

$$\langle \mathbf{u}_h \cdot \mathbf{n}, \mu_h\rangle_{\mathcal{E}_h^0} + \langle \tau_K p_h, \mu_h\rangle_{\mathcal{E}_h^0} - \langle \tau_K \hat{p}_h, \mu_h\rangle_{\mathcal{E}_h^0} = 0 \tag{2.13c}$$

$$\langle \hat{p}_h, \mu_h\rangle_{\Gamma_D} = \langle g_D, \mu_h\rangle_{\Gamma_D} \tag{2.13d}$$

$$\langle \mathbf{u}_h \cdot \mathbf{n}, \mu_h\rangle_{\Gamma_N} + \langle \tau_K p_h, \mu_h\rangle_{\Gamma_N} - \langle \tau_K \hat{p}_h, \mu_h\rangle_{\Gamma_N} = \langle g_N, \mu_h\rangle_{\Gamma_N} \tag{2.13e}$$

$$\langle \mathbf{u}_h \cdot \mathbf{n}, m_h\rangle_{\Gamma_I} + \langle \tau_K p_h, m_h\rangle_{\Gamma_I} - \langle \lambda_h, m_h\rangle_{\Gamma_I} = g_I|\Gamma_I|^{-1}\langle 1, m_h\rangle_{\Gamma_I} \tag{2.13f}$$

for all $(\mathbf{v}_h, q_h, \mu_h, m_h) \in \mathbf{V}_h^k \times W_h^k \times M_h^k \times C_h$.

Remark 1. The boundary condition (2.2c) is enforced directly in the definition of the trial space M_h^k.

Theorem 2 (Existence and uniqueness of the discrete solution)**.** *The discrete problem* (2.12), (2.13) *has a unique solution.*

Proof. Proposition 3.1 of [Cockburn, 2010] holds unchanged. Indeed by testing the discrete equations with $(\mathbf{v}_h, q_h, \mu_h, m_h) = (\mathbf{u}_h, p_h, -\hat{p}_h, -\lambda_h)$ and integrating by parts, also in our case we obtain the discrete energy equation

$$(\kappa^{-1}\mathbf{u}_h, \mathbf{u}_h)_\Omega + \tau_K\langle p_h - \hat{p}_h - \lambda_h, p_h - \hat{p}_h - \lambda_h\rangle_{\mathcal{E}_h} = (f, p_h)_\Omega - \langle g_N, \hat{p}_h\rangle_{\Gamma_N} - g_I|\Gamma_I|^{-1}\langle 1, \lambda_h\rangle_{\Gamma_I}.$$

Now, to prove uniqueness (which, in the discrete setting, implies existence thanks to the finite dimensionality of the discrete spaces) we need to prove that vanishing data yield the null solution. Let then $f = 0$, $g_N = 0$ and $g_I = 0$. The discrete energy equation then imply $\mathbf{u}_h = 0$ and $p_h^K = \hat{p}_h$ on ∂K for all K. Testing equation (2.13a) with $\vec{v}_h^K = \nabla p_h^K \in \mathbf{V}^k(K)$ we obtain that ∇p_h^K is a constant for all K. Then $\hat{p}_h$ is a constant itself, which is necessarily 0, due to the Dirichlet boundary condition. □

2.1.3 Post-Processing

Under the assumptions that the exact solution is smooth enough, and the stabilization function τ_K is of order one, this method gives an order of convergence of $k + 1$ for both the potential and the flux when the polynomial order used is k.

Here, we will show the post processing technique presented in [Cockburn et al., 2012] aiming to obtain a better approximation of the potential p_h^*, which converges one order higher than p_h in L^2. If the potential $p_h \in W_h^k$, we want to find $p_h^* \in W_h^{k+1}$ such that for all $K \in \Omega_h$:

$$(\nabla p_h^*, \nabla q_h)_K = -(\kappa^{-1}\mathbf{u}_h, \nabla q_h)_K \qquad \forall q_h \in W_h^{k+1} \tag{2.14a}$$

$$(p_h^*, 1)_K = (p_h, 1)_K \tag{2.14b}$$

The equation (2.14b) constrains p_h^* to have the same mean value than p_h.

2.2 Static condensation

Problem 2.13 is a monolithic problem, where all variables are solved at once. Since the flux, primal and trace variables are discontinuous the number of dof is very large. The idea of the static condensation is to first solve a global problem on the faces of the mesh corresponding to the transmission conditions between elements and the boundary conditions. This allows to compute $\hat{p}$ on the faces. Then we compute $(\mathbf{u}, p)$ by solving local problems on each cell $K \in \Omega_h$.
The formulation for both these problems can be obtained from the formulation (2.13). We first need to define the following local matrices:

$$A_{00}^K = (\kappa\mathbf{u}, \mathbf{v})_K \qquad A_{01}^K = -(p, \nabla \cdot \mathbf{v})_K \qquad F_0^K = (\mathbf{g}, \mathbf{v})_K$$
$$A_{10}^K = (\nabla \cdot \mathbf{u}, q)_K \qquad A_{11}^K = \langle \tau p, q \rangle_{\partial K} \qquad F_1^K = (f, q)_K$$

- If $K \cap (\Gamma_I \cup \Gamma_D) = \emptyset$:

$$A_{20}^K = \langle \mathbf{u} \cdot \mathbf{n}, \mu \rangle_{\partial K} \qquad A_{21}^K = \langle \tau p, \mu \rangle_{\partial K} \qquad A_{22}^K = -\langle \tau \hat{p}, \mu \rangle_{\partial K}$$

- If $K \cap \Gamma_I = \emptyset$:

$$A_{02}^K = -\langle \hat{p}, \mathbf{n} \cdot \mathbf{n} \rangle_{\partial K}$$
$$A_{12}^K = -\langle \tau \hat{p}, q \rangle_{\partial K}$$

- If $K \cap \Gamma_D \neq \emptyset$:

$$A_{22}^K = \langle \hat{p}, \mu \rangle_{\partial K} \qquad F_2^K = \langle g_D, \mu \rangle_K$$

- If $K \cap \Gamma_N \neq \emptyset$:

$$F_2^K = \langle g_N, \mu \rangle_K$$

- If $K \cap \Gamma_I \neq \emptyset$:

$$A_{03}^K = \langle \lambda, \mathbf{v} \cdot \mathbf{n} \rangle_{\partial K} \qquad A_{13}^K = -\langle \lambda, \tau q \rangle_{\partial K} \qquad F_3^K = \frac{1}{|\Gamma_I|} \langle g_I, m \rangle_{\partial K}$$
$$A_{30}^K = \langle \mathbf{u} \cdot \mathbf{n}, m \rangle_{\partial K} \qquad A_{31}^K = \langle \tau p, m \rangle_{\partial K} \qquad A_{33}^K = -\langle \lambda, m \rangle_{\partial K}$$

If $\hat{p}$ and λ are known, then $(\mathbf{u}, p)$ can be uniquely determined by:

$$\underbrace{\begin{pmatrix} A_{00}^K & A_{01}^K \\ A_{10}^K & A_{11}^K \end{pmatrix}}_{A^K} \underbrace{\begin{pmatrix} \mathbf{u}_K \\ p_K \end{pmatrix}}_{U^K} = \underbrace{\begin{pmatrix} F_0^K \\ F_1^K \end{pmatrix}}_{F^K} - \underbrace{\begin{pmatrix} A_{02}^K & A_{03}^K \\ A_{12}^K & A_{13}^K \end{pmatrix}}_{B^K} \underbrace{\begin{pmatrix} \hat{p}_{\partial K} \\ \lambda_{\partial K} \end{pmatrix}}_{T^K}$$

or:

$$U^K = (A^K)^{-1} F^K - (A^K)^{-1} B^K T^K \tag{2.15}$$

The trace can be written as $\hat{\mathbf{u}} \cdot \mathbf{n} = \mathbf{u} \cdot \mathbf{n} + \tau p - \tau \hat{p} - \tau \lambda$, and the form associated with it:

$$\mu \in M^k(\partial K) \mapsto \begin{cases} \langle \mathbf{u} \cdot \mathbf{n} + \tau p, \mu \rangle_{\partial K} - \langle \tau \hat{p}, \mu \rangle_{\partial K} \text{ if } K \notin \Gamma_I \\ \langle \mathbf{u} \cdot \mathbf{n} + \tau p, \mu \rangle_{\partial K} - \langle \tau \lambda, m \rangle_{\partial K} \text{ if } K \in \Gamma_I \end{cases}$$

whose matrix representation is:

$$\underbrace{\begin{pmatrix} A_{20}^K & A_{21}^K \\ A_{30}^K & A_{31}^K \end{pmatrix}}_{C^K} \underbrace{\begin{pmatrix} \mathbf{u}_K \\ p_K \end{pmatrix}}_{U^K} + \underbrace{\begin{pmatrix} A_{22}^K & 0 \\ 0 & A_{33}^K \end{pmatrix}}_{E^K} \underbrace{\begin{pmatrix} \hat{p}_{\partial K} \\ \lambda_{\partial K} \end{pmatrix}}_{T^K}$$

or using 2.15:

$$\begin{aligned} C^K U^K + E^K T^K &= C^K (A^K)^{-1} F^K - C^K (A^K)^{-1} B^K T^K + E^K T^K \\ &= \underbrace{C^K (A^K)^{-1} F^K}_{D_f^K} - \underbrace{(C^K (A^K)^{-1} B^K + E^K)}_{D^K} T^K \end{aligned} \tag{2.16}$$

Assembling the local matrices D^K, $D_f^K + F_2^K + F_3^K$, T^K into respectively $\mathbb{H}$, $\mathbb{F}$ and $\mathbb{T}$, we have a global problem to solve:

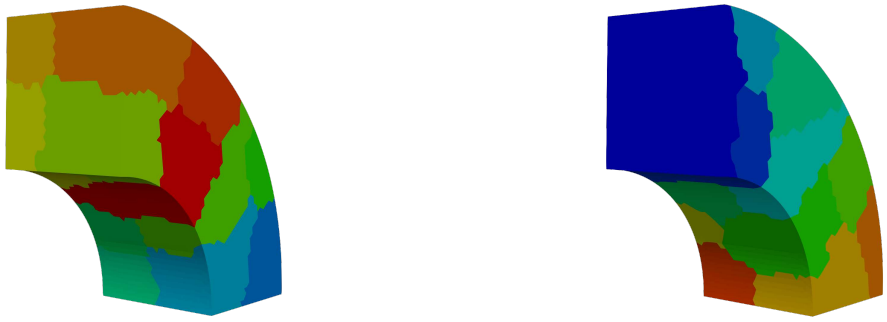
$$\mathbb{H}\mathbb{T} = \mathbb{F} \tag{2.17}$$

Thus, we first resolve the global problem (2.17) to find the traces $(\hat{p}, \lambda)$ and then, using them in the right-hand side of the local problems (2.15), we can retrieve $(\mathbf{u}, p)$.

2.3 Performances

2.3.1 IBC and partitioning

In parallel, a special care has to be taken for the partitioning of the mesh when using integral boundary conditions. Indeed, in absence of integral boundary conditions, each element $K \in \Omega_h$ has degrees of freedom inside it, and on each face $\mathcal{F}_h$, but not on its vertices. Therefore, two elements K_1 and K_2 share their degrees of freedom only if they share a face. In CG method, the nodal continuity at vertices connects the degrees of freedom of an element to an unknown number of elements' degrees of freedom, depending on the size of the mesh.
In the case of integral boundary conditions, the connectivity is different. All the faces $F \in \mathcal{F}_h^{\Gamma_I}$ share a single common degree of freedom through λ. Thus, all the degrees of freedom of all the elements K having at least one face on Γ_I are coupled. In parallel, we need to ensure that all the elements sharing a face with Γ_I are on the same processor, otherwise the communications become the bottleneck of the computation. We impose to the partitioner that each IBC face is a neighbor of all the other IBC faces. The standard partitioning, where we do not ensure that all the elements are on the same processor and the special partitioning for 16 processors are presented in figure 2.1. The difference of computational time between the standard and the special partitioning is detailed in table 2.1.



(a) Standard (b) Special

Figure 2.1 – Standard and special partitioning on 16 procs with IBC condition in front

Partitioning	$np = 2$	$np = 4$	$np = 8$	$np = 16$
Standard	1132.78	412.666	17147.3	21778.5
Special	1145.71	424.482	354.697	31.272
Speed-up	0.99	0.97	48.44	674.84

Table 2.1 – Time to solve a problem with $30e^6$ dofs using standard and special partitionning for different number of processors

We can see that without the special partitioning, the advantages of parallel computing is totally lost, more processors leads to worst performances. Whereas with special

partitioning, for 2 or 4 processors we have the same performances than with the standard partitioning, this is expected since the IBC surface is not partitioned for so few processors. But with 8 or 16 processors, the speed-up is important, the special partitioning is 674 times faster than the standard one with 16 processors.
It is absolutely necessary to use this partitioning when dealing with integral boundary condition in parallel.

2.3.2 Conservation of the current density

An important feature of the HDG method is that we impose the conservativity of the traces between elements [Arnold et al., 2002]. In the case of an electric problem, this leads to have much better conservation of the current density $\mathbf{j}$, and so we really comply with the condition $\nabla \cdot \mathbf{j} = 0$. To verify this property, we computed the difference of input current and output current in a magnet, using a Newton method for the nonlinearity in CG on a fine mesh (22M elements, 4.3×10^6 dofs), a Picard method in CG on a coarse mesh (3.3M elements, 10^6 dofs) and a Picard method in HDG on the coarse mesh ($(39 + 13 + 22) \times 10^6$ dofs). If the HDG method takes more time to solve (5252s on 32 procs) than with the CG method (944s for the fine mesh, 284s for the coarse mesh, on 32 procs), we can see in table 2.2 that the difference between the input current and the output current is practically 0 with the HDG method, whereas with the CG method, both with the fine and coarse mesh, the difference is much larger (≈ 30).

Method	Mesh	Input $\mathbf{j}$	Output $\mathbf{j}$	Difference
CG Newton	fine	30982.94	-31013.84	30.9
CG Picard	coarse	30991.47	-31021.05	29.58
HDG Picard	coarse	30898.33	-30898.35	0.02

Table 2.2 – Difference on the current between inlet and outlet

This property is very important both from a physical point of view, for the engineers of the LNCMI, and from a computational point of view, when using the current density as right-hand side in the magnetostatic problem, having $\nabla \cdot \mathbf{j} = 0$ is crucial for the convergence of the methods.

2.3.3 Computational cost

Even if the total number of degrees of freedom is greater than in the CG formulation, the use of the static condensation allows to reduce drastically the cost of the HDG method. A comparative study between CG and HDG methods can be found in [Kirby et al., 2012].
In table 2.3 is presented reference error for the potential (e_p) or the flux (e_u), number of dof (# Dof) and time of resolution (for the creation of the matrix, the vector, the resolution and the total) for a 2D Continuous Galerkin problem of polynomial order 1. In table 2.4 is presented the corresponding results in HDG to match each reference error (including the post processed error for the potential e_{p^*}), number of dof or time of resolution of table 2.3 for the same problem and the same polynomial order.

All the computations done in this part have been done on one of the five nodes of ATLAS, the computing resource cluster at Cemosis and IRMA laboratory. The node

used has 24 cores on 2 sockets (Intel Xeon E5-2680 v3 2.50GHz) hyperthreaded with 256 GB of RAM. Everything is interconnected with both 10Gb Ethernet cards and 40Gb Infini-band cards. The workload manager is slurm. For both CG and HDG, the Geometric Algebraic MultiGrid (GAMG) preconditioner [Amestoy et al., 2000] is used.

h	# Dof	e_p	e_u	N	matrix	vector	solve	total
$5e^{-3}$	$1.1e^5$	$6.2e^{-7}$	$1.4e^{-3}$	1 12	0.51s	2.17s	1.72s	4.4s

Table 2.3 – Reference for a 2D CG problem with polynomial order 1

h	# Dof	e_p	e_{p^*}	e_u	N	matrix	vector	solve	post	total
$2.5e^{-2}$	$\mathbf{1.1e^5}$	$1.0e^{-5}$	$7.8e^{-8}$	$4.2e^{-5}$	1 12	0.04s	0.97s	0.92s	0.3s	2.23s
$6e^{-3}$	$1.8e^6$	$\mathbf{5.8e^{-7}}$	$1.0e^{-9}$	$2.4e^{-6}$	1 12	0.75s	17.13s	27.5s	6.8s	52.2s
$5e^{-2}$	$2.7e^4$	$3.9e^{-5}$	$\mathbf{4.9e^{-7}}$	$1.7e^{-4}$	1 12	0.02s	0.34s	0.22s	0.09s	0.67s
$1.5e^{-1}$	$3.2e^3$	$3.4e^{-4}$	$1.2e^{-5}$	$\mathbf{1.4e^{-3}}$	1 12	0.02s	0.17s	0.04s	0.03s	0.26s

Table 2.4 – Corresponding values to CG problem in bold for a 2D HDG problem with polynomial order 1

We can see in the previous tables that with the same number of dofs, the HDG method is slower than the CG method. Also for the same error for the potential, HDG is much slower than CG, but the error on the flux is much lower. But if we post process the potential, then for the same error as the potential in the CG method, HDG becomes faster than CG, and the error on the flux is lower. If we are only interested in the error on the flux, then we can see that for the same error, HDG is also faster than CG although the error on the potential is greater in this case.

2.4 Conclusion

In this chapter, we have described the use of the Integral Boundary Condition (IBC) in the case of a mixed Poisson problem and proved that the resulting problem is well-posed. We have also described a way to use static condensation when dealing with this type of condition to efficiently solve the problem. Finally, we detailed three points showing the performances of the method, first in parallel when using IBC with the special partitioning, then with respect to the conservation of the flux, and we compared the computational cost of the HDG method with the CG method at comparable errors. We found that for similar cost, we are able to reach better errors and physical properties with HDG than with CG. Even if the HDG method is competitive compared to CG we

want to achieve real-time computations and for this we need another method to solve our problems.

Chapter 3

Model Order Reduction

The finite element method presented in the previous chapter is widely used in both industrial and academic context for its robustness and precision and the fact that it builds on a powerful mathematical framework. But a small change of a parameter in the equations requests to do all the resolution again, leading to a high computational cost when we need to solve the problem for numerous parameters. This can be the case when trying to do uncertainty quantification, optimize part of the problem, a component or an experimental setup for example, control the state of an experiment, or simply explore the solutions set. This is called many-query methods, and even with the ever growing power of super computers, the cost can be too high to use the high fidelity resolution.

To lower the computational complexity of such problems, the methods of Model Order Reduction (MOR) are used. One possible way of reducing the order of a model is to simplify the physics involved in the problem, but the accuracy of our solution decreases accordingly. Other methods can be used where the model equations are projected on a space with a reduced dimensionality than the full order space. Such methods are for example the proper generalized decomposition [Chinesta et al., 2011], the proper orthogonal decomposition [Kerschen et al., 2005] or the reduced basis method. We will focus on the latter.

The work on the reduced basis method started for non-linear problems with [Noor, 1981]. It has then been extended to other types of problems such as incompressible flows or multi-parameter problems [Balmès, 1996], [Fink and Rheinboldt, 1983], [Peterson, 1989], [Rheinboldt, 1992]. The method was improved by the use of efficient errors estimators, leading to the Certified Reduced Basis Method, see [Prud'Homme et al., 2001], [Veroy et al., 2003], [Prud'homme and Patera, 2004], [Quarteroni et al., 2011], [Rozza et al., 2007] or [Patera and Rozza, 2007]. The estimated error between the full-order model and the reduced one is computed on a quantity of interest, the output of the model, with an offline/online strategy. This allows to explore the parametric domain and optimally build the reduced basis.

The first section of this chapter will describe the notations and the greedy algorithm to select the basis vectors. In the second section, we will detail the operator approxi-

mations used, the Empirical Interpolation Method (EIM) [Barrault et al., 2004], [Grepl, Martin A. et al., 2007], [Maday et al., 2008], its discrete version, [Chaturantabut and Sorensen, 2010], an algorithm to accelerate the method, [Daversin, C. et al., 2013], used in case of non-linear problems and the Empirical Quadrature Method (EQM).

Contents

3.1 Reduced Basis Method

We seek to solve a parametric problem:

Problem 1. Find $u(\boldsymbol{\mu}) \in V$ such that:

$$a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f(v; \boldsymbol{\mu}) \quad \forall v \in V$$

and evaluate

$$s(\boldsymbol{\mu}) = l(u(\boldsymbol{\mu}); \boldsymbol{\mu})$$

$u(\boldsymbol{\mu})$ is known as the exact solution. It belongs to the solution manifold:

$$\mathcal{M} = \{u(\boldsymbol{\mu}) \,|\, \boldsymbol{\mu} \in \mathbb{P} \text{ and } u(\boldsymbol{\mu}) \text{ solution of Problem 1}\} \subset V$$

$\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_p)$ belongs to the space $\mathbb{P} \subset \mathbb{R}^p$.

We want to approximate this solution numerically, thus we use a space V_h of dimension N_h and of basis $\{\varphi_i\}_{i=1}^{N_h}$.
This approximation of high accuracy is obtained by solving the "truth" problem:

Problem 2. Find $u_h(\boldsymbol{\mu}) \in V_h$ such that:

$$a(u_h(\boldsymbol{\mu}), v_h; \boldsymbol{\mu}) = f(v_h; \boldsymbol{\mu}) \quad \forall v_h \in V_h$$

The solutions $u_h(\boldsymbol{\mu})$ belong to the discrete version of the solution manifold:

$$\mathcal{M}_h = \{u_h(\boldsymbol{\mu}) \,|\, \boldsymbol{\mu} \in \mathbb{P} \text{ and } u_h(\boldsymbol{\mu}) \text{ solution of Problem 2}\} \subset V_h$$

But the cost of solving Problem 2 depends on N_h and thus can be very large. The idea of the reduced basis method is that for many problems we can approximate $\mathcal{M}_h$ with a few basis functions with a small error. This basis $\{\xi_i\}_{i=1}^N$ is the reduced basis which spans the subspace V_{rb} of V_h, of dimension $N \ll N_h$.
The reduced problem reads as:

Problem 3. Find $u_{rb}(\boldsymbol{\mu}) \in V_{rb}$ such that:

$$a(u_{rb}(\boldsymbol{\mu}), v_{rb}; \boldsymbol{\mu}) = f(v_{rb}; \boldsymbol{\mu}) \quad \forall v_{rb} \in V_{rb}$$

The quality of the space V_{rb} of dimension N, is determined by

$$d(\mathcal{M}_h, V_{rb}) = \sup_{u_h \in \mathcal{M}_h} \inf_{v_{rb} \in V_{rb}} \|u_h - v_{rb}\|_V$$

and the best possible error of approximation within the spaces of dimension N is the Kolmogorov N-width [Kolmogoroff, 1936]:

$$\inf_V d(\mathcal{M}_h, V). \tag{3.1}$$

The idea is to build V_{rb} such that $d(\mathcal{M}_h, V_{rb})$ decreases rapidly when N increases or at least such that $d(\mathcal{M}_h, V_{rb})$ is close to the Kolmogorov N-width.

3.1.1 Reduced basis space generation

Let introduce the discrete set of parameters $\mathbb{P}_h \subset \mathbb{P}$ and the associated manifold of cardinality $M = |\mathbb{P}_h|$:

$$\mathcal{M}_h(\mathbb{P}_h) = \{u_h(\boldsymbol{\mu}) \,|\, \boldsymbol{\mu} \in \mathbb{P}_h \text{ and } u_h(\boldsymbol{\mu}) \text{ solution of Problem 2}\} \subset V_h$$

We consider the case where a is continuous coercive and symmetric. We introduce a scalar product and the norm associated with the bilinear form:

$$(u, v)_{\boldsymbol{\mu}} = a(u, v; \boldsymbol{\mu}) \qquad\qquad \|u\|_{\boldsymbol{\mu}} = \sqrt{(u, u)_{\boldsymbol{\mu}}}$$

In the following, we will use a reference parameter $\bar{\boldsymbol{\mu}}$ for the scalar product $(\cdot, \cdot)_{\bar{\mu}}$ and the norm $\|\cdot\|_{\bar{\mu}}$.

In the next sections, we will present the two most common method to generate the reduced basis.

3.1.1.1 Proper Orthogonal Decomposition

The Proper Orthogonal Decomposition (POD) is probably the best known method for model order reduction, it has been introduced by Noor in [Noor, 1981], and then studied by other researchers [Fink and Rheinboldt, 1983, Peterson, 1989, Porsching and Lee, 1987]. We will show briefly how it works.
We denote $\psi_m = u_h(\boldsymbol{\mu}_m)$ for $m = 1, \ldots, M$ and create the correlation matrix $C \in \mathbb{R}^{M \times M}$:

$$C_{mq} = \frac{1}{M}(\psi_m, \psi_q)_V, \quad 1 \leq m, q \leq M$$

Then, we solve the following eigen problem for the N eigenmodes (λ_n, v_n) with the largest eigenvalues and $\|v_n\|_{l^2(\mathbb{R}^M)} = 1$

$$Cv_n = \lambda_n v_n, \quad 1 \leq n \leq M$$

We can now define the reduced basis $\{\xi_n\}$. With $\lambda_1 \geq \cdots \geq \lambda_N$, we have:

$$\xi_n(x) = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} (v_n)_m \psi_m(x) \quad 1 \leq n \leq N$$

The space V_{POD} spanned by $\{\xi_n\}$ is the N dimensional space that minimizes the quantity:

$$\sqrt{\frac{1}{M} \sum_{\boldsymbol{\mu} \in \mathbb{P}_h} \inf_{v_{rb} \in V_{rb}} \|u_h(\boldsymbol{\mu}) - v_{rb}\|_V^2}$$

over all N-dimensional subspaces V_{rb}.
If P_N is the projection from V onto V_{POD}, then we have the following error estimate:

$$\sqrt{\frac{1}{M} \sum_{m=1}^{M} \|\psi_m - P_N(\psi_m)\|_V^2} = \sqrt{\sum_{m=N+1}^{M} \lambda_m}$$

But in practice, determining $\mathbb{P}_h$ to have a good approximation is difficult and thus, we often need to have $M \gg N$, which makes the construction of the matrix C and the resolution of the eigen problem very expensive (it scales like $O(NN_h^2)$).

3.1.1.2 Greedy algorithm

Here, we will explain the greedy algorithm, see [DeVore et al., 2012] for more details.
We introduce an error bound $\eta(\boldsymbol{\mu})$ such that:

$$\|u_h(\boldsymbol{\mu}) - u_{rb}(\boldsymbol{\mu})\|_{\bar{\mu}} \leq \eta(\boldsymbol{\mu}), \quad \forall \boldsymbol{\mu} \in \mathbb{P}$$

We will develop how to define such estimator in section 3.1.3.

The greedy algorithm is an iterative procedure where at each step, we expand the reduced basis with $u_h(\boldsymbol{\mu})$ corresponding to the parameter $\boldsymbol{\mu}$ that maximizes the error $\eta(\boldsymbol{\mu})$.
We first set $n = 1$, choose $\boldsymbol{\mu}_1$ randomly in $\mathbb{P}_h$, and set a tolerance tol, such that when $\eta(\boldsymbol{\mu}) \leq tol$, the process is over.

1. We compute $u_h(\boldsymbol{\mu}_n)$ and set $V_{rb} = \text{span}\{u_h(\boldsymbol{\mu}_1), \ldots, u_h(\boldsymbol{\mu}_n)\}$
2. For each $\boldsymbol{\mu} \in \mathbb{P}_h$
   (a) We compute $u_{rb}(\boldsymbol{\mu}) \in V_{rb}$
   (b) We evaluate $\eta(\boldsymbol{\mu})$
3. We choose $\boldsymbol{\mu}_{n+1} = \arg\max_{\boldsymbol{\mu} \in \mathbb{P}_h} \eta(\boldsymbol{\mu})$

4. If $\eta(\boldsymbol{\mu}_{n+1}) > tol$ set $n = n + 1$ and go back to 1.
   Else terminate.

The advantages over the POD are that for a N-dimensional space, we only need N truth approximations, and the evaluations of $u_{rb}(\boldsymbol{\mu})$ and $\eta(\boldsymbol{\mu})$ are inexpensive. Additionally, since the basis at each step are hierarchical, if one want to improve the quality of the space V_{rb}, one needs only to do n additional steps.

If we assume that the bilinear form a is coercive and continuous for every $\boldsymbol{\mu} \in \mathbb{P}$, that is there exists a positive constant $\alpha(\boldsymbol{\mu}) \geq \alpha \geq 0$ and a finite constant $\gamma(\boldsymbol{\mu}) \leq \gamma \leq 0$ such that

$$a(v, v; \boldsymbol{\mu}) \geq \alpha(\boldsymbol{\mu})\|v\|_V^2 \quad \text{and} \quad a(w, v; \boldsymbol{\mu}) \leq \gamma(\boldsymbol{\mu})\|w\|_V\|v\|_V$$

and that the solution manifold $\mathcal{M}$ has en exponentially small Kolmogorov N-width $d_N(\mathcal{M}) \leq ce^{-aN}$, $a > \log(1 + \sqrt{\frac{\gamma}{\alpha}})$, then it has been proven [Buffa, Annalisa et al., 2012], that the reduced basis approximation converges exponentially fast in the sense that there exists $\beta > 0$ such that

$$\forall \boldsymbol{\mu} \in \mathbb{P}, \quad \|u_h(\boldsymbol{\mu}) - u_{rb}(\boldsymbol{\mu})\|_V \leq Ce^{-\beta N}$$

Remark 4. We shall note that the vectors $\{u_h(\boldsymbol{\mu}_1), \ldots, u_h(\boldsymbol{\mu}_N)\}$ may be linearly dependent, so we need to orthogonalize it using for example the Gram-Schmidt algorithm to obtain the basis $\{\xi_1, \ldots, \xi_N\}$. It is necessary to have a reasonable condition number, bounded by $\frac{\gamma(\boldsymbol{\mu})}{\alpha(\boldsymbol{\mu})}$.

3.1.2 Affine decomposition

Let's have a look at the different issues from an algebraic point of view. We recall that the basis of V_h is $\{\varphi_1, \ldots, \varphi_{N_h}\}$ and the reduced basis is $\{\xi_1, \ldots, \xi_N\}$. Then we can write $u_h^{\boldsymbol{\mu}} = \sum_{i=1}^{N_h} u_i^{\boldsymbol{\mu}} \varphi_i$, $\xi_n = \sum_{i=1}^{N_h} \xi_{in}\varphi_i$, and $u_{rb}^{\boldsymbol{\mu}} = \sum_{n=1}^{N} u_{rb,n}^{\boldsymbol{\mu}} \xi_n$.
The truth problem is then

$$A_h^{\boldsymbol{\mu}} U_h^{\boldsymbol{\mu}} = F_h^{\boldsymbol{\mu}}$$

with $A_h^{\boldsymbol{\mu}} \in \mathbb{R}^{N_h \times N_h}$, $U_h^{\boldsymbol{\mu}}$ and $F_h^{\boldsymbol{\mu}}$ belonging to $\mathbb{R}^{N_h}$, such that

$$(A_h^{\boldsymbol{\mu}})_{ij} = a(\varphi_i, \varphi_j; \boldsymbol{\mu}), \quad (U_h^{\boldsymbol{\mu}})_i = u_i^{\boldsymbol{\mu}}, \quad (F_h^{\boldsymbol{\mu}})_i = f(\varphi_i; \boldsymbol{\mu})$$

We introduce the matrices $B \in \mathbb{R}^{N_h \times N}$, $A_{rb}^{\boldsymbol{\mu}} \in \mathbb{R}^{N \times N}$ and the vector $F_{rb}^{\boldsymbol{\mu}} \in \mathbb{R}^N$ such that

$$(B)_{in} = \xi_{in}, \quad (A_{rb}^{\boldsymbol{\mu}})_{mn} = a(\xi_m, \xi_n; \boldsymbol{\mu}), \quad (F_{rb}^{\boldsymbol{\mu}})_n = f(\xi_n; \boldsymbol{\mu})$$

The matrix $A_{rb}^{\boldsymbol{\mu}}$ can be computed as $B^T A_h^{\boldsymbol{\mu}} B$, and the vector $F_{rb}^{\boldsymbol{\mu}}$ as $B^T F_h^{\boldsymbol{\mu}}$.
Then the reduced problem could be solved by:

$$A_{rb}^{\boldsymbol{\mu}} U_{rb}^{\boldsymbol{\mu}} = F_{rb}^{\boldsymbol{\mu}}$$

where $(U_{rb}^{\boldsymbol{\mu}})_n = u_{rb,n}^{\boldsymbol{\mu}}$.
The problem is that to compute $A_{rb}^{\boldsymbol{\mu}}$ we need to assemble the truth matrix $A_h^{\boldsymbol{\mu}}$. But this matrix depends on N_h and thus, the resolution of the reduced problem would also

depend on it, diminishing its performances. The idea is then to split the computations using an offline/online strategy, where all the heavy computations are done during the offline phase, and during the online phase, only the assembly of the matrix and vector and the resolution are done.

Now, let's assume that we can decompose the forms $a(\cdot,\cdot;\boldsymbol{\mu})$, $f(\cdot;\boldsymbol{\mu})$, $l(\cdot;\boldsymbol{\mu})$ such that:

$$a(w,v;\boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}) a_q(w,v), \qquad a_q : V_h \times V_h \to \mathbb{R}, \theta_a^q : \mathbb{P} \to \mathbb{R}, \qquad \forall q = 1,\dots,Q_a \tag{3.2}$$

$$f(v;\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}) f_q(v), \qquad f_q : V_h \to \mathbb{R}, \theta_f^q : \mathbb{P} \to \mathbb{R}, \qquad \forall q = 1,\dots,Q_f \tag{3.3}$$

$$l(v;\boldsymbol{\mu}) = \sum_{q=1}^{Q_l} \theta_l^q(\boldsymbol{\mu}) l_q(v) \qquad l_q : V_h \to \mathbb{R}, \theta_l^q : \mathbb{P} \to \mathbb{R}, \qquad \forall q = 1,\dots,Q_l \tag{3.4}$$

We can introduce the matrices $A_{rb}^q \in \mathbb{R}^{N\times N}$, for $1 \le q \le Q_a$, and the vectors $F_{rb}^q \in \mathbb{R}^N$ for $1 \le q \le Q_f$ such that:

$$(A_{rb}^q)_{mn} = a_q(\xi_m, \xi_n) \quad (F_{rb}^q)_n = f_q(\xi_n)$$

or

$$A_{rb}^q = B^T A_h^q B \quad F_{rb}^q = B^T F_h^q$$

Now, we have that:

$$A_{rb}^{\boldsymbol{\mu}} = \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}) A_{rb}^q, \quad F_{rb}^{\boldsymbol{\mu}} = \sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}) F_{rb}^q \tag{3.5}$$

where all the matrices A_{rb}^q and the vectors F_{rb}^q can be pre-computed during the offline phase, where the complexity depends on the dimension of the finite element space. In doing so, the assembly of the matrix $A_{rb}^{\boldsymbol{\mu}}$ does not depend on N_h, and so can be done during the online phase, such that the complexity depends solely on N, Q_a and Q_f.

3.1.3 Certified Error Bound

In this section we will detail the definition of $\eta(\boldsymbol{\mu})$ needed in the greedy algorithm for the reduced bases generation (see section 3.1.1.2). The efficient computation of such a bound for coercive problems is detailed in [Ngoc Cuong et al., 2005] [Prud'Homme et al., 2001]. An error bound should be accurate enough to be valid for all N and for all parameter values in $\mathbb{P}$ and to allow the minimal number of basis to achieve the precision wanted. Since we need to compute $\eta(\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in \mathbb{P}_h$, and we may have to compute it during the online phase, it needs to be fast to compute.

3.1.3.1 Estimators and efficiency

Key ingredients of the certified error bound are the error estimators and their efficiency. We will show how to compute them, interested readers can found more details

in [Hesthaven et al., 2015]. We first need to define the discrete coercivity and continuity constants of the bilinear form a:

$$\alpha_h(\boldsymbol{\mu}) = \inf_{v_h \in V_h} \frac{a(v_h, v_h; \boldsymbol{\mu})}{\|v_h\|_V^2}, \quad \text{and} \quad \gamma_h(\boldsymbol{\mu}) = \sup_{w_h \in V_h} \sup_{v_h \in V_h} \frac{a(w_h, v_h; \boldsymbol{\mu})}{\|w_h\|_V \|v_h\|_V} \tag{3.6}$$

Since V_h is included in V, we have $\alpha(\boldsymbol{\mu}) \leq \alpha_h(\boldsymbol{\mu})$ and $\gamma_h(\boldsymbol{\mu}) \leq \gamma(\boldsymbol{\mu})$.

We will use $s_h(\boldsymbol{\mu}) = l(u_h(\boldsymbol{\mu}); \boldsymbol{\mu})$, $s_{rb}(\boldsymbol{\mu}) = l(u_{rb}(\boldsymbol{\mu}); \boldsymbol{\mu})$ and the error $e(\boldsymbol{\mu}) = u_h(\boldsymbol{\mu}) - u_{rb}(\boldsymbol{\mu})$ and the equation linking it to the residual:

$$a(e(\boldsymbol{\mu}), v_h; \boldsymbol{\mu}) = r(v_h; \boldsymbol{\mu}) \quad \forall v_h \in V_h \tag{3.7}$$

where $r(\cdot; \boldsymbol{\mu})$ is the residual living in the dual space of V_h. Since $r(\cdot; \boldsymbol{\mu}) \in V_h'$, we introduce its Riesz representation $\hat{r}_h(\boldsymbol{\mu}) \in V_h$ such that:

$$(\hat{r}_h(\boldsymbol{\mu}), v_h)_V = r(v_h; \boldsymbol{\mu}), \quad \forall v_h \in V_h \tag{3.8}$$

We recall that

$$\|\hat{r}_h(\boldsymbol{\mu})\|_V = \|r(\cdot; \boldsymbol{\mu})\|_{V_h'} = \sup_{v_h \in V_h} \frac{r(v_h; \boldsymbol{\mu})}{\|v_h\|_V} \tag{3.9}$$

We will assume that we have a lower bound for $\alpha_h(\boldsymbol{\mu})$ that we will note $\alpha_{LB}(\boldsymbol{\mu})$. This lower bound will be detailed in 3.1.3.3.

Then we can define the error estimators for the energy norm, the output and the relative output:

$$\eta_{en}(\boldsymbol{\mu}) = \frac{\|\hat{r}_h(\boldsymbol{\mu})\|_V}{\sqrt{\alpha_{LB}(\boldsymbol{\mu})}}, \tag{3.10}$$

$$\eta_s(\boldsymbol{\mu}) = \frac{\|\hat{r}_h(\boldsymbol{\mu})\|_V^2}{\alpha_{LB}(\boldsymbol{\mu})} = \eta_{en}(\boldsymbol{\mu})^2, \tag{3.11}$$

$$\eta_{s,rel}(\boldsymbol{\mu}) = \frac{\|\hat{r}_h(\boldsymbol{\mu})\|_V^2}{\alpha_{LB}(\boldsymbol{\mu}) s_{rb}(\boldsymbol{\mu})} = \frac{\eta_s(\boldsymbol{\mu})}{s_{rb}(\boldsymbol{\mu})} \tag{3.12}$$

By remarking that

$$\alpha_{LB}(\boldsymbol{\mu}) \|e(\boldsymbol{\mu})\|_V^2 \leq a(e(\boldsymbol{\mu}), e(\boldsymbol{\mu}); \boldsymbol{\mu}) = \|e(\boldsymbol{\mu})\|_\mu^2 \leq \|\hat{r}_h(\boldsymbol{\mu})\|_V \|e(\boldsymbol{\mu})\|_V$$

we have that those estimators are rigorous upper bounds for the following errors

$$\|u_h(\boldsymbol{\mu}) - u_{rb}(\boldsymbol{\mu})\|_\mu \leq \eta_{en}(\boldsymbol{\mu}), \tag{3.13}$$

$$s_h(\boldsymbol{\mu}) - s_{rb}(\boldsymbol{\mu}) \leq \eta_s(\boldsymbol{\mu}), \tag{3.14}$$

$$\frac{s_h(\boldsymbol{\mu}) - s_{rb}(\boldsymbol{\mu})}{s_h(\boldsymbol{\mu})} \leq \eta_{s,rel}(\boldsymbol{\mu}) \tag{3.15}$$

To control the quality of the estimators, we introduce the effictivity index associated:

$$\texttt{eff}_{en}(\boldsymbol{\mu}) = \frac{\eta_{en}(\boldsymbol{\mu})}{\|u_h(\boldsymbol{\mu}) - u_{rb}(\boldsymbol{\mu})\|_\mu}, \tag{3.16}$$

$$\texttt{eff}_s(\boldsymbol{\mu}) = \frac{\eta_s(\boldsymbol{\mu})}{s_h(\boldsymbol{\mu}) - s_{rb}(\boldsymbol{\mu})}, \tag{3.17}$$

$$\texttt{eff}_{s,rel}(\boldsymbol{\mu}) = \frac{\eta_{s,rel}(\boldsymbol{\mu}) s_h(\boldsymbol{\mu})}{s_h(\boldsymbol{\mu}) - s_{rb}(\boldsymbol{\mu})} \tag{3.18}$$

The efficiency of the estimators have to be ≥ 1 for the error bound to be correct, but to ensure a dimension of the basis as small as possible and avoid unnecessary basis vector, we want it to be as close to 1 as possible. If we assume that the problem is coercive and compliant, that is $l = f$ and a is symmetric for all parameters, we have:

$$\texttt{eff}_{en}(\boldsymbol{\mu}) \leq \sqrt{\frac{\gamma_h(\boldsymbol{\mu})}{\alpha_{LB}(\boldsymbol{\mu})}}, \tag{3.19}$$

$$\texttt{eff}_{s}(\boldsymbol{\mu}) \leq \frac{\gamma_h(\boldsymbol{\mu})}{\alpha_{LB}(\boldsymbol{\mu})}, \tag{3.20}$$

$$\texttt{eff}_{s,rel}(\boldsymbol{\mu}) \leq (1 + \eta_{s,rel})\frac{\gamma_h(\boldsymbol{\mu})}{\alpha_{LB}(\boldsymbol{\mu})} \tag{3.21}$$

Some other estimators that could be useful to define are the ones using the V-norm:

$$\eta_V(\boldsymbol{\mu}) = \frac{\|\hat{r}_h(\boldsymbol{\mu})\|_V}{\alpha_{LB}(\boldsymbol{\mu})}, \qquad \|u_h(\boldsymbol{\mu}) - u_{rb}(\boldsymbol{\mu})\|_V \leq \eta_V(\boldsymbol{\mu}), \tag{3.22}$$

$$\eta_{V,rel}(\boldsymbol{\mu}) = \frac{2\|\hat{r}_h(\boldsymbol{\mu})\|_V}{\alpha_{LB}(\boldsymbol{\mu})\|u_{rb}(\boldsymbol{\mu})\|_V}, \qquad \frac{\|u_h(\boldsymbol{\mu}) - u_{rb}(\boldsymbol{\mu})\|_V}{\|u_h(\boldsymbol{\mu})\|_V} \leq \eta_{V,rel}(\boldsymbol{\mu}) \tag{3.23}$$

The associated effectivity indices are

$$\texttt{eff}_V(\boldsymbol{\mu}) = \frac{\eta_V(\boldsymbol{\mu})}{\|u_h(\boldsymbol{\mu}) - u_{rb}(\boldsymbol{\mu})\|_V}, \qquad \texttt{eff}_V(\boldsymbol{\mu}) \leq \frac{\gamma_h(\boldsymbol{\mu})}{\alpha_{LB}(\boldsymbol{\mu})}, \tag{3.24}$$

$$\texttt{eff}_{V,rel}(\boldsymbol{\mu}) = \frac{\eta_{V,rel}(\boldsymbol{\mu})\|u_h(\boldsymbol{\mu})\|_V}{\|u_h(\boldsymbol{\mu}) - u_{rb}(\boldsymbol{\mu})\|_V}, \qquad \texttt{eff}_{V,rel}(\boldsymbol{\mu}) \leq 3\frac{\gamma_h(\boldsymbol{\mu})}{\alpha_{LB}(\boldsymbol{\mu})} \tag{3.25}$$

As stated in section 3.1.3, we need to compute those estimators in the offline phase for all $\boldsymbol{\mu} \in \mathbb{P}_h$, and also in the online phase. Thus, the computation need to be efficient. All estimators rely on $\|\hat{r}_h(\boldsymbol{\mu})\|_V$ and $\alpha_{LB}(\boldsymbol{\mu})$. We will use the reduced basis to compute those two quantities efficiently.

3.1.3.2 Riesz representation

In this section, we will explain how to compute $\|\hat{r}_h(\boldsymbol{\mu})\|_V$ effectively, that is, using the reduced basis, more details can found in [Hesthaven et al., 2015].

First, we need to decompose $r(v_h; \boldsymbol{\mu})$ by using the affine decomposition:

$$r(v_h; \boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}) f_q(v_h) - \sum_{q=1}^{Q_a}\sum_{n=1}^{N} \theta_a^q(\boldsymbol{\mu}) u_{rb,n}^{\boldsymbol{\mu}} a_q(\xi_n, v_h)$$

We introduce the vector $r(\boldsymbol{\mu}) \in \mathbb{R}^{Q_r}$ with $Q_r = Q_f + NQ_a$:

$$r(\boldsymbol{\mu}) = \left(\theta_f^1(\boldsymbol{\mu}), \ldots, \theta_f^{Q_f}(\boldsymbol{\mu}), -(u_{rb}^{\boldsymbol{\mu}})^T\theta_a^1(\boldsymbol{\mu}), \ldots, -(u_{rb}^{\boldsymbol{\mu}})^T\theta_a^{Q_a}(\boldsymbol{\mu})\right)^T \tag{3.26}$$

and the vector of linear forms $R \in (V_h')^{Q_r}$:

$$R = \left(f_1, \ldots, f_{Q_f}, a_1(\xi_1, \cdot), \ldots, a_1(\xi_N, \cdot), \ldots, a_{Q_a}(\xi_1, \cdot), \ldots, a_{Q_a}(\xi_N, \cdot)\right)^T \tag{3.27}$$

Combining (3.26) and (3.27), we have:

$$(\hat{r}_h(\boldsymbol{\mu}), v_h)_V = r(v_h; \boldsymbol{\mu}) = \sum_{q=1}^{Q_r} r_q(\boldsymbol{\mu}) R_q(v_h) \quad \forall v_h \in V_h$$

By noting $\hat{r}_h^q$ the Riesz representation of $R_q \in V_h'$ for $1 \leq q \leq Q_r$, we have:

$$\hat{r}_h(\boldsymbol{\mu}) = \sum_{q=1}^{Q_r} r_q(\boldsymbol{\mu}) \hat{r}_h^q, \qquad \|\hat{r}_h(\boldsymbol{\mu})\|_V^2 = \sum_{p,q=1}^{Q_r} r_p(\boldsymbol{\mu}) r_q(\boldsymbol{\mu}) (\hat{r}_h^p, \hat{r}_h^q)_V \tag{3.28}$$

By reusing the notation from the section 3.1.2, with the additional matrices $M_h, A_h^q \in \mathbb{R}^{N_h \times N_h}$ for $1 \leq q \leq Q_a$, and the vectors $F_h^q \in \mathbb{R}^{N_h}$ for $1 \leq q \leq Q_f$, such that:

$$(M_h)_{ij} = (\varphi_i, \varphi_j)_V, \quad (A_h^q)_{ij} = a_q(\varphi_i, \varphi_j), \quad (F_h^q)_i = f_q(\varphi_i)$$

we can rewrite (3.28) as

$$H = (F_h^1, \ldots, F_h^{Q_f}, A_h^1 B, \ldots, A_h^{Q_a} B)^T \tag{3.29}$$
$$G = H^T M_h^{-1} H \in \mathbb{R}^{Q_r \times Q_r}$$
$$\|\hat{r}_h(\boldsymbol{\mu})\|_V = \sqrt{r(\boldsymbol{\mu})^T G r(\boldsymbol{\mu})} \tag{3.30}$$

3.1.3.3 Stability constant

There exists several ways to compute efficiently the stability constant $\alpha_{LB}(\boldsymbol{\mu})$. The two first, Min-$\theta$ and multi parameter Min-θ [Machiels et al., 2000], [Veroy, Karen et al., 2002], [Patera and Rozza, 2007] are restricted to a parametrically coercive problems. The last, the successive constraint method (SCM) [Huynh et al., 2007], [Vallaghé et al., 2011], is more general and more precise but more complex.

First we need to recall that computing the coercivity constant

$$\alpha_h(\boldsymbol{\mu}) = \inf_{v_h \in V_h} \frac{a(v_h, v_h; \boldsymbol{\mu})}{\|v_h\|_V^2}$$

is the same than finding the smallest eigenvalue of the following problem:

Problem 4. Find $(\lambda, w_h) \in \mathbb{R}^+ \times V_h$ such that

$$a(w_h, v_h; \boldsymbol{\mu}) = \lambda (w_h, v_h)_V \quad \forall v_h \in V_h$$

Min-θ approach

The Min-θ and the multi parameter Min-θ methods are restricted to the problems that are parametrically coercive:

- $\theta_a^q(\boldsymbol{\mu}) > 0, \forall \boldsymbol{\mu} \in \mathbb{P},\ q = 1, \ldots, Q_a$
- $a_q(\cdot, \cdot) : V_h \times V_h \to \mathbb{R}$ is semi-positive definite for all $q = 1, \ldots, Q_a$.

To use the Min-θ approach, we assume to already have computed one coercivity constant, $\alpha_h(\boldsymbol{\mu}')$ for a parameter $\boldsymbol{\mu}'$, using Problem 4.
Then we can compute the lower bounds for each $\boldsymbol{\mu}$:

$$\alpha_{LB}(\boldsymbol{\mu}) = \alpha_h(\boldsymbol{\mu}') \min_{q=1,\dots,Q_a} \frac{\theta_a^q(\boldsymbol{\mu})}{\theta_a^q(\boldsymbol{\mu}')} \tag{3.31}$$

During the offline phase we solve the eigenvalue problem, which can be computationally intensive, and in the online phase, we use the equation (3.31) to compute the bound efficiently for $\boldsymbol{\mu}$.

Multi-parameter Min-θ approach

The multi-parameter Min-θ approach refines the simple Min-θ approach by using M parameter $\boldsymbol{\mu}^1, \dots, \boldsymbol{\mu}^M$ for which the coercivity constant $\alpha_h(\boldsymbol{\mu}^m)$ is computed during the offline phase by using Problem 4.
Then we can compute the lower bounds fast in the online phase for each $\boldsymbol{\mu}$:

$$\alpha_{LB}(\boldsymbol{\mu}) = \max_{m=1,\dots,M} \left(\alpha_h(\boldsymbol{\mu}^m) \min_{q=1,\dots,Q_a} \frac{\theta_a^q(\boldsymbol{\mu})}{\theta_a^q(\boldsymbol{\mu}^m)} \right) \tag{3.32}$$

This method is more accurate than the simple Min-θ, but it also requires more eigenvalue problems to be solved. Furthermore, both methods are restrained to the parametrically coercive problems, for other problems, we need to use the Successive Constraint Method.

Successive Constraint Method (SCM)

By using the affine decomposition, we can write

$$\alpha_h(\boldsymbol{\mu}) = \inf_{v_h \in V_h} \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}) \frac{a_q(v_h, v_h)}{\|v_h\|_V^2}.$$

The right-hand side can be written as a minimization problem:

Problem 5. Find $\alpha_h(\boldsymbol{\mu})$ such that;

$$\alpha_h(\boldsymbol{\mu}) = \min_{y \in \mathcal{Y}} S(\boldsymbol{\mu}, y)$$

where the functional S is defined as:

$$S : \mathbb{P} \times \mathbb{R}^{Q_a} \longrightarrow \mathbb{R}$$

$$(\boldsymbol{\mu}, y) \longmapsto S(\boldsymbol{\mu}, y) = \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}) y_q$$

and the set $\mathcal{Y}$ by

$$\mathcal{Y} = \left\{ y = (y_1, \dots, y_{Q_a}) \in \mathbb{R}^{Q_a} \mid \exists v_h \in V_h \text{ s.t. } y_q = \frac{a_q(v_h, v_h)}{\|v_h\|_V^2},\ 1 \le q \le Q_a \right\}$$

To have a lower and upper bounds of $\alpha_h(\boldsymbol{\mu})$, we enlarge and respectively restrict the set of admissible solutions, $\mathcal{Y}_{UB} \subset \mathcal{Y} \subset \mathcal{Y}_{LB}$.

To define the sets $\mathcal{Y}_{UB}$ and $\mathcal{Y}_{LB}$, we use an iterative process. We arbitrarily choose a parameter $\boldsymbol{\mu}^1$ in a subset Ξ_a of $\mathbb{P}_h$. We set a tolerance `tol` to stop the process.
And we also need to introduce the function providing close parameter values:

$$P_M(\boldsymbol{\mu}; E) = \begin{cases} M \text{ closest points to } \boldsymbol{\mu} \in E & \text{if } |E| > M, \\ E & \text{if } |E| \leq M \end{cases}$$

The iterative process is the following:
We first need to do some initialization:

- Solve for the smallest σ_q^- and the largest σ_q^+ eigenvalue of the problem:

$$a_q(w_h, v_h) = \sigma_q(w_h, v_h)_V \quad \forall v_h \in V_h \quad \forall q = 1, \ldots, Q_a$$

- Set the box $\mathcal{B}$ such that: $\mathcal{B} = \prod_{q=1}^{Q_a} [\sigma_q^-, \sigma_q^+]$
- Set $C_0 = \emptyset$ and $\alpha_{LB}^0(\boldsymbol{\mu}) = 0, \ \forall \boldsymbol{\mu} \in \Xi_a$, and $n = 1$,

Then do while $\eta_n(\boldsymbol{\mu}_n) >$ `tol`:

1. Set $C_n = C_{n-1} \cup \{\boldsymbol{\mu}_n\}$
2. Solve the eigen problem

$$a(w_h^n, v_h; \boldsymbol{\mu}^n) = \alpha_h(\boldsymbol{\mu}^n)(w_h^n, v_h)_V \quad \forall v_h \in V_h$$

3. Set $(y^n)_q = \dfrac{a_q(w_h^n, w_h^n)}{\|w_h^n\|_V^2}$
4. Set $\mathcal{Y}_{UB}^n = \{y^j, \ 1 \leq j \leq n\}$
5. For each $\boldsymbol{\mu} \in \Xi_a$:
   (a) Compute $\alpha_{UB}^n(\boldsymbol{\mu}) = \min_{y \in \mathcal{Y}_{UB}^n} S(\boldsymbol{\mu}, y)$
   (b) Set

$$\mathcal{Y}_{LB}^n = \left\{ \begin{array}{r} y \in \mathcal{B} \,|\, S(\boldsymbol{\mu}', y) \geq \alpha_h(\boldsymbol{\mu}') \ \forall \boldsymbol{\mu}' \in P_{M_\alpha}(\boldsymbol{\mu}, C_n), \\ S(\boldsymbol{\mu}', y) \geq \alpha_{LB}^{n-1}(\boldsymbol{\mu}') \ \forall \boldsymbol{\mu}' \in P_{M_+}(\boldsymbol{\mu}, \Xi_a \setminus C_n) \end{array} \right\} \tag{3.33}$$

   (c) Compute $\alpha_{LB}^n(\boldsymbol{\mu}) = \min_{y \in \mathcal{Y}_{LB}^n} S(\boldsymbol{\mu}, y)$
   (d) Set $\eta_n(\boldsymbol{\mu}) = 1 - \dfrac{\alpha_{LB}^n(\boldsymbol{\mu})}{\alpha_{UB}^n(\boldsymbol{\mu})}$
6. Set $\boldsymbol{\mu}_{n+1} = \arg\max_{\boldsymbol{\mu} \in \Xi_a} \eta_n(\boldsymbol{\mu})$ and $n = n + 1$

To find $\alpha_{LB}(\boldsymbol{\mu})$ for any $\boldsymbol{\mu} \in \mathbb{P}$, we have to solve

$$\alpha_{LB}(\boldsymbol{\mu}) = \min_{y \in \mathcal{Y}_{LB}(\boldsymbol{\mu})} S(\boldsymbol{\mu}, y) \tag{3.34}$$

This algorithm depends on P_{M_α} and P_{M_+}. An improvement of this algorithm, simplifying the choice of those values and accelerating it, is proposed in [Vallaghé et al., 2011] and explained hereafter.
By noting that (3.33) is a linear program with Q_a design variables, we can decrease the number of constraints. First, the constraints on $\boldsymbol{\mu}' \in P_{M_+}(\boldsymbol{\mu}, \Xi_a \setminus C_n)$ only take into account a lower bound of the coercivity constant, and thus are of poorer quality than the first set of constraints. So we remove them completely, along with the need of M_+. In a linear program with Q_a variables, there is maximum Q_a active constraints when the minimum is reached. In addition to this, we can see that if at the step n a constraint is not active, then it will not be active for the next steps. We need to keep track of the active constraints at each step, and so we do not need M_α anymore. The space $\mathcal{Y}_{LB}$ now reads:

$$\mathcal{Y}_{LB}^n = \begin{Bmatrix} y \in \mathcal{B} \,|\, S(\boldsymbol{\mu}', y) \geq \alpha_h(\boldsymbol{\mu}') \; \forall \boldsymbol{\mu}' \in A(\boldsymbol{\mu}, C_{n-1}), \\ S(\boldsymbol{\mu}_n, y) \geq \alpha_h(\boldsymbol{\mu}_n) \end{Bmatrix}$$

where $A(\boldsymbol{\mu}, C_{n-1})$ is the set of active constraints at minimum for the computation of $\alpha_{LB}(\boldsymbol{\mu}, C_{n-1})$. After $\alpha_{LB}(\boldsymbol{\mu}, C_n)$ is computed, we check which constraints are active among $A(\boldsymbol{\mu}, C_{n-1}) \cup \boldsymbol{\mu}_n$ and store the active ones in $A(\boldsymbol{\mu}, C_n)$.

During the online phase, the set of active constraints $A(\boldsymbol{\mu}, C_n)$ has not been computed, so the idea is to use all $\boldsymbol{\mu}_n \in C_n$. Even if it seems a brute force approach, the authors note that the computational cost is still reasonable, being able to compute 10^5 lower bounds in 1 minute for $K = 1500$. The advantage of this method is to reduce considerably the offline cost of the SCM, with an example reaching a factor 60 between the original and the proposed method.

3.2 Operator approximation

We may need to approximate operators such as non-linear expressions, matrices, vectors or integrals. The following sections detail methods for such approximations.

3.2.1 Empirical Interpolation Method

Very often, the hypothesis made in section 3.1.2 that we can write

$$a(u, v; \boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}) a_q(u, v)$$

does not hold. In this case, we want to approximate the non-affine terms in a way that would allow us to establish an affine decomposition for a. Such method is described in [Barrault et al., 2004], [Grepl, Martin A. et al., 2007], [Maday et al., 2008] or [Daversin, C. et al., 2013] and is called the Empirical Interpolation Method.

The idea of the Empirical Interpolation Method (EIM), is to approximate a general parametrized function by a sum of affine terms:

$$g(x, \boldsymbol{\mu}) \approx g_M(x, \boldsymbol{\mu}) = \sum_{m=1}^{M} \theta_{g,M}^m(\boldsymbol{\mu}) q_m(x)$$

To do this, we choose a subset $\Xi_{EIM} \subset \mathbb{P}_{EIM}$, and a tolerance. In the following, we will note $g_{\boldsymbol{\mu}} = g(\cdot, \boldsymbol{\mu}) : \Omega \to \mathbb{R}$ and $g_{M,\boldsymbol{\mu}} = g_M(\cdot, \boldsymbol{\mu})$.

1. Choose $\boldsymbol{\mu}_1 \in \Xi_{EIM}$ s.t. $g_{\boldsymbol{\mu}_1} \neq 0$, $t_1 = \arg\max_{x \in \Omega} |g_{\boldsymbol{\mu}_1}(x)|$, $q_1(x) = \frac{g_{\boldsymbol{\mu}_1}(x)}{g_{\boldsymbol{\mu}_1}(t_1)}$
2. Set $e_1 = \|g_{\boldsymbol{\mu}_1} - g_{1,\boldsymbol{\mu}_1}\|_{L^2(\Omega)}$
3. Set $m = 2$ and do until $e_{m-1} < tol$
   (a) Choose $\boldsymbol{\mu}_m = \arg\max_{\boldsymbol{\mu} \in \Xi_{EIM}} \|g_{\boldsymbol{\mu}} - g_{m-1,\boldsymbol{\mu}}\|_{L^\infty(\Omega)}$
   (b) Set $(T^{m-1})_{ij} = q_j(t_i)$, $(g_{\boldsymbol{\mu}_m}^{m-1})_i = g_{\boldsymbol{\mu}_m}(t_i)$, $1 \leq i, j \leq m-1$,
   (c) Solve $T^{m-1}\theta_{g,m-1} = g_{\boldsymbol{\mu}_m}^{m-1}$
   (d) Set $r_m = g_{\boldsymbol{\mu}_m} - g_{m,\boldsymbol{\mu}_m}$, $t_m = \arg\sup_{x \in \Omega} |r_m(x)|$, $q_m(x) = \frac{r_m(x)}{r_m(t_m)}$.
   (e) Set $e_m = \|r_m\|_{L^2(\Omega)}$, $m = m + 1$

This procedure output a basis $q_1, \ldots, q_M$ of linearly independent functions and interpolation points $t_1, \ldots, t_M$ such that the matrix $T_{ij} = (T^M)_{ij} = q_j(t_i)$ is lower triangular with unity diagonal of size $M \times M$.

Then to compute $g_{M,\boldsymbol{\mu}}$ for all $\boldsymbol{\mu} \in \mathbb{P}_{EIM}$, we need to solve the linear problem:

$$T\theta_{g,M} = g_{\boldsymbol{\mu}}^M \tag{3.35}$$

For example, say we have

$$a(u, v : \boldsymbol{\mu}) = \int_\Omega g(x, \boldsymbol{\mu}) b(u, v; x) dx$$

Then using the EIM decomposition of $g(x, \boldsymbol{\mu}) \approx \sum_{m=1}^M \theta_{g,M}^m q_m(x)$ we have the following approximation for a:

$$a(u, v; \boldsymbol{\mu}) \approx \sum_{m=1}^{M} \theta_{g,M}^m(\boldsymbol{\mu}) \int_\Omega q_m(x) b(u, v; x) dx = \sum_{m=1}^{M} \theta_{g,M}^m(\boldsymbol{\mu}) a_q(u, v)$$

3.2.2 EIM for Discrete Operators

In some cases, for complex operators such as stabilization terms or geometric transformations, it is not easy to have an analytical expression of the non-affine component. For this, we want to approximate the discrete operator directly [Chaturantabut and

Sorensen, 2010]. The main elements of the method and algorithm stay the same, but instead to act on space/parameter functions, we act on vectors or matrices. We want to have:

$$\mathbf{T}(x, \boldsymbol{\mu}) \approx \mathbf{T}_M(x, \boldsymbol{\mu}) = \sum_{m=0}^{M} \Theta_m(\boldsymbol{\mu}) \boldsymbol{\Phi}^m(x)$$

The main difference is that in steps 1 and 3d, instead of setting the interpolation point t_m, we set an interpolation index i_m such that:

$$i_m = \arg\max_{j \in \mathbb{I}} \left| \mathbf{R}_m(x, \boldsymbol{\mu})_j \right|$$

An element of the set of indices $\mathbb{I}$ can be an integer defining the index of a vector or a pair of integers in a matrix. More generally, for a tensor of order r, $\mathbb{I}$ contains tuples of integers.

An index of a tensor corresponds to a degree of freedom in a finite element space. To be able to compute efficiently the right-hand side of (3.35) during the online phase, we need to build a finite element space containing only the degrees of freedom linked to the indices of interpolations.
In order to do this, we need to extract a sub-mesh containing only the elements associated with these degrees of freedom. This step is very important if we want to be independent of the finite element dimension during the online phase.

3.2.3 Simultaneous EIM and Reduced basis

We now consider non-linear problems where we cannot apply the previous method readily.

Problem 6. Find $u_h(\boldsymbol{\mu}) \in V_h$ such that:

$$a(u_h, v_h; \boldsymbol{\mu}) = \theta_a^1(\boldsymbol{\mu}) a_1(u, v) + (g(u(\boldsymbol{\mu})), v) = f(v_h; \boldsymbol{\mu}) \quad \forall v \in V_h$$

where g is a non linear function.

We could try to solve the following system:

$$\theta_a^1(\boldsymbol{\mu}) A_h^1 u_h^{\boldsymbol{\mu}} + g_h(u_h^{\boldsymbol{\mu}}) = F_h$$

where $(g_h(u_h^{\boldsymbol{\mu}}))_i = (g(u_h(\boldsymbol{\mu})), \varphi_i)$.
This gives the following reduced problem:

$$\theta_a^1(\boldsymbol{\mu}) A_{rb}^1 u_{rb}^{\boldsymbol{\mu}} + g_{rb}(u_{rb}^{\boldsymbol{\mu}}) = F_{rb}$$

where $g_{rb}(u_{rb}^{\boldsymbol{\mu}}) = B^T g_h(B u_{rb}^{\boldsymbol{\mu}})$.
But the matrix $B \in \mathbb{R}^{N_h \times N}$ so this would not be efficient to compute such a term.

We could use the EIM approximation of g to write:

$$(g_{rb}(u_{rb}^{\boldsymbol{\mu}}), v_{rb}) \approx (g_M(u_{rb}^{\boldsymbol{\mu}}), v_{rb}) = \sum_{m=1}^{M} \theta_{g,M}^m(\boldsymbol{\mu}) b_m(v_{rb})$$

with

$$b_m(v_{rb}) = (q_m, v_{rb}) \tag{3.36}$$

$$\theta^m_{g,M}(\boldsymbol{\mu}) = \sum_{k=1}^{M} (T^{-1})_{mk}\ g(u^{\boldsymbol{\mu}}_{rb}(t_k)) \tag{3.37}$$

$$= \sum_{k=1}^{M} (T^{-1})_{mk}\ g\left(\sum_{n=1}^{N} (u^{\boldsymbol{\mu}}_{rb})_n \xi_n(t_k)\right) \tag{3.38}$$

$$\tag{3.39}$$

Then the problem 6 can be written as:

$$\theta^1_a(\boldsymbol{\mu}) A^1_{rb} u^{\boldsymbol{\mu}}_{rb} + \sum_{m=1}^{M} \theta^m_{g,M}(\boldsymbol{\mu}) b^q_{rb} = F_{rb} \tag{3.40}$$

with $(b^q_{rb})_n = b_q(\xi_n)$.
And it can be solved using Newton or Picard algorithm.

The issue in such case is that the step 3a implies that you use the "truth" solver to compute the solution of problem 6 for each parameter $\boldsymbol{\mu} \in \Xi_{EIM}$. This can be quite expensive to do, especially for a non-linear problem.
An algorithm has been presented in [Daversin and Prud'Homme, 2015] to reduce the cost of this computation, the Simultaneous EIM and Reduced basis (SER). Instead of using the "truth" solver, we would use the reduced basis approximation to speed up the construction of the interpolation.

Of course the EIM is necessary to have an efficient reduced basis approximation. We then need to build simultaneously the EIM basis and reduced basis:

1. Choose $\boldsymbol{\mu}_1$ and compute the first EIM basis q_1 and first reduced basis ξ_1.
2. Set $n = 1$ and $m = 1$.
3. Do until reaching tolerance or maximum number of basis:
   (a) Use u^n_{rb} to choose $\boldsymbol{\mu}_{m+1}$ in step 3a and build q_{m+1}.
   (b) Use g_m to compute ξ_{n+1}.
   (c) Set $n = n + 1$ and $m = m + 1$.

This algorithm allows to reduce considerably the number of finite element resolution when using EIM. Instead of doing $\#\Xi_{EIM}$ resolutions to compute the approximation errors, and N resolutions for the reduced basis vectors, we only do $N + 1$ resolutions, one for each basis vector of the reduced basis and one to start the EIM basis.
Other variants have been studied in [Daversin Catty, 2016], the most effective being the multi-level SER, which consists in doing multiple times the loop of the SER algorithm, using the reduced basis approximation of the previous loop, thus improving the approximation.

Validation

To validate our implementation of the EIM and SER methods, we propose to test it on a 2D non-linear and non-affinely parametrized benchmark, proposed in [Grepl, Martin A. et al., 2007].

The domain $\Omega = [0,1]^2$ is the unit square. The parameter space is $\mathcal{D} = [0.01, 10]^2$. The problem is described by the following elliptic and non linear equation, $\forall \boldsymbol{\mu} = (\mu_1, \mu_2) \in \mathcal{D}$

$$-\Delta u(\boldsymbol{\mu}) + \mu_1 \frac{e^{\mu_2 u(\boldsymbol{\mu})} - 1}{\mu_2} = 100 \sin(2\pi x) \sin(2\pi y), \tag{3.41}$$

and $u(\boldsymbol{\mu}) = 0$ on $\Gamma = \partial\Omega$.

The average of u on Ω is the output of interest.

$$s(\boldsymbol{\mu}) = \int_\Omega u(\boldsymbol{\mu})\, d\Omega\,. \tag{3.42}$$

Introducing the space $X_h = \{v_h \in C^0(\Omega_h) \,|\, v_{h|K} \in P_K(K)\}$, we can write the weak formulation as find $u_h(\boldsymbol{\mu}) \in X_h$ such that

$$\int_\Omega \nabla u_h(\boldsymbol{\mu}) \cdot \nabla v_h \, d\Omega \; + \int_\Omega \mu_1 \frac{e^{\mu_2 u_h(\boldsymbol{\mu})} - 1}{\mu_2} v_h \, d\Omega \; = \int_\Omega 100 \sin(2\pi x) \sin(2\pi y) v_h \, d\Omega\,, \quad \forall v_h \in X_h \tag{3.43}$$

To compute the RB approximation of $u_N(\boldsymbol{\mu})$, we will use an EIM approximation of the term $\mu_1 \frac{e^{\mu_2 u_h(\mu)} - 1}{\mu_2}$:

$$\mu_1 \frac{e^{\mu_2 u_h(\boldsymbol{\mu})} - 1}{\mu_2} \approx \sum_{m=1}^{M} \beta_m(\boldsymbol{\mu}) g^m. \tag{3.44}$$

The mesh is composed of 2880 elements and the train set for the EIM greedy algorithm is a uniform grid 40×40 of $\mathcal{D}$, and the dual norm of the residual is used as an error indicator. A fixed point algorithm is used to solve the problem both during the offline and the online phases. We present a profile of the solution for an arbitrary chosen $\boldsymbol{\mu}$, on figure 3.1.

We are interested in the relative error between the truth approximation $u_h(\boldsymbol{\mu})$ and the RB approximation $u_N(\boldsymbol{\mu})$, and also the relative error between the truth output and its RB approximation:

$$e_N^u(\boldsymbol{\mu}) = \frac{\|u_h(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})\|_0}{\|u_h(\boldsymbol{\mu})\|_0},$$
$$e_N^s(\boldsymbol{\mu}) = \frac{|s_h(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})|}{|s_h(\boldsymbol{\mu})|}.$$

We compare the classic EIM algorithm and the SER algorithm on 50 parameters chosen randomly with respect to the number of reduced basis used, up to 20 basis.

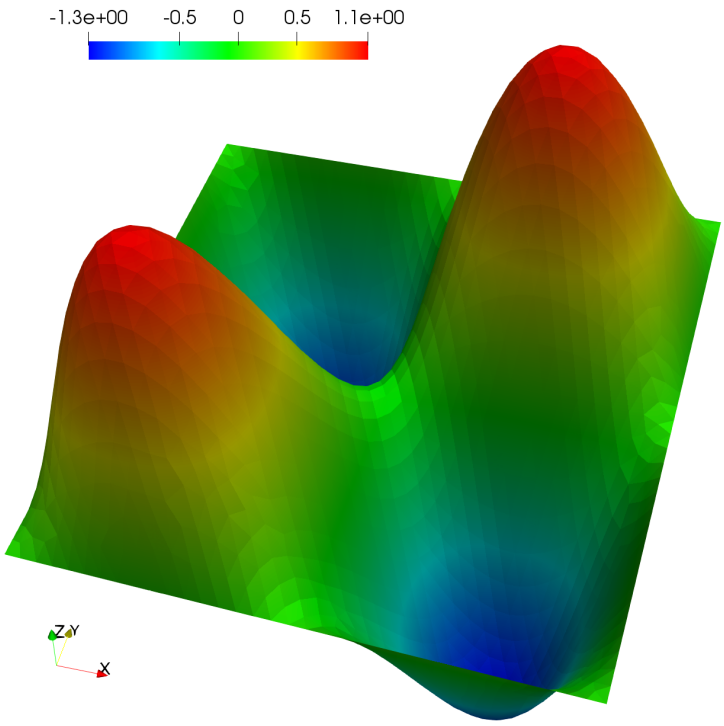
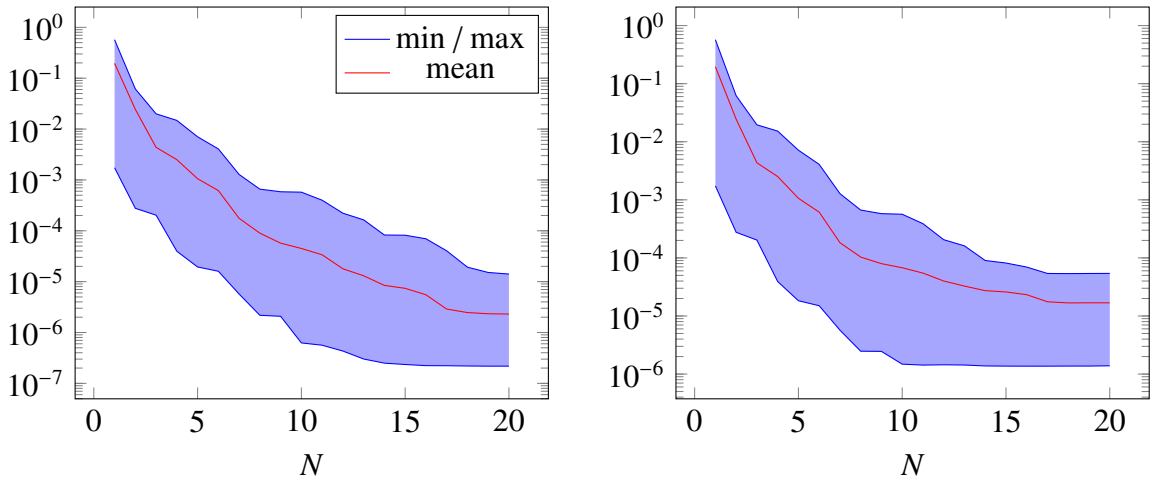


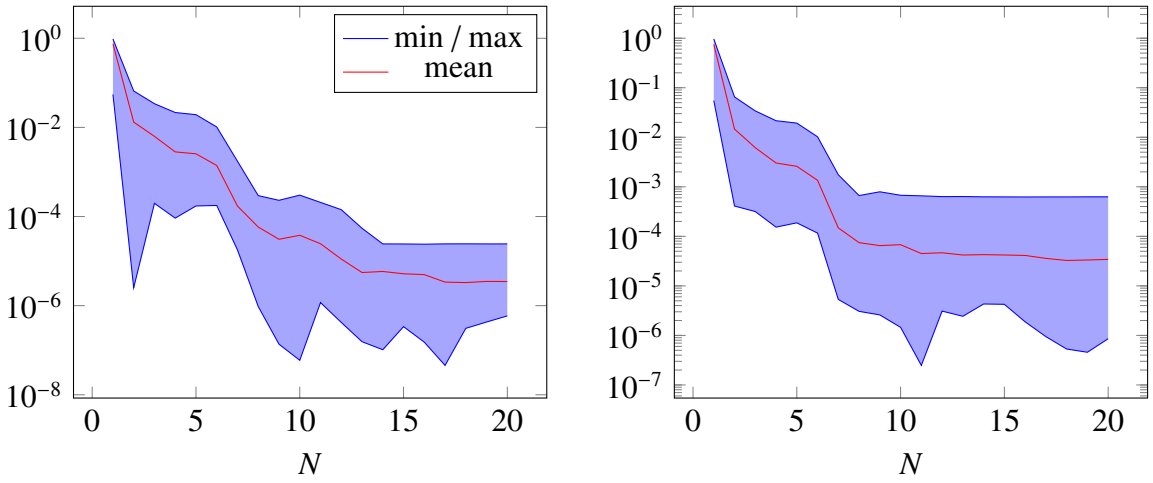
Figure 3.1 – Example of the solution for problem (3.41) with $\boldsymbol{\mu} = (1.91, 2.77)$



(a) Classic EIM

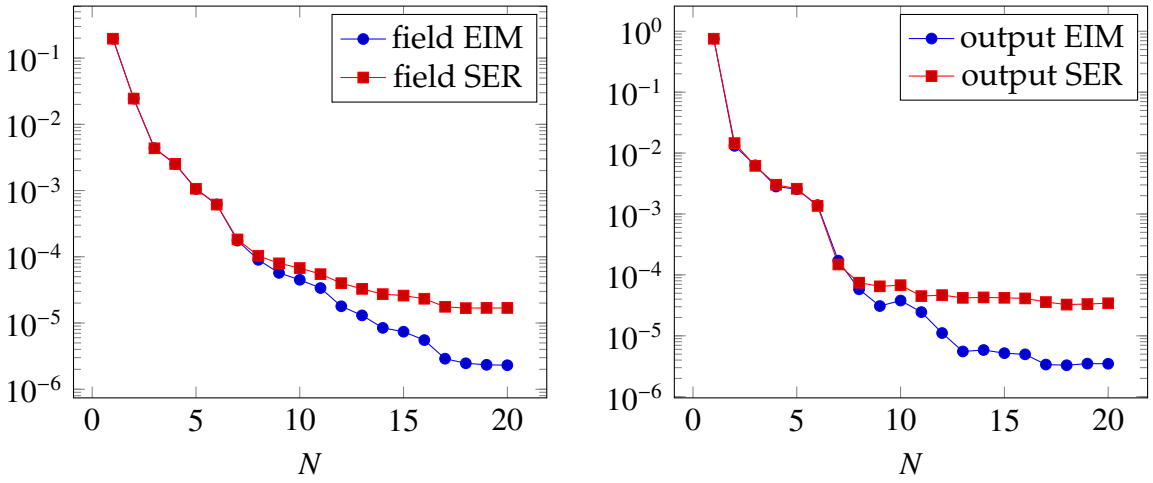
(b) SER

Figure 3.2 – Min, max and mean errors on the field, e_N^u, on 50 parameters



(a) Classic EIM (b) SER

Figure 3.3 – Min, max and mean errors on the output, e_N^s, on 50 parameters



(a) Error e_N^u on the field u (b) Error e_N^s ont the output s

Figure 3.4 – Comparison of the mean errors on u and the output between SER and EIM

The errors presented in figures 3.2 and 3.3 have the same behavior regarding the repartition between the min and the max. Whereas the figure 3.4 shows that even if the SER algorithm converges more slowly than the classical EIM, we reach an error that is reasonable in comparison. Especially, if we take into account that the convergence study took 4000s with the classical algorithm whereas with SER, it took only 500s.

3.2.4 Empirical Quadrature Method

In [Yano and Patera, 2019], a method is presented to reduce the computational cost of integrating parametrized functions. They developed an offline/online procedure to obtain an empirical quadrature rule optimized for the parametrized function considered. This allows not to depend anymore on the finite element dimension for computing an integral.

When computing an integral with the finite element method, we use quadrature points set to discretize the integral, see [Canuto et al., 2006]. A quadrature rule is a set of points and weights that allows to approximate the integral. In `Feel++` the quadrature points set $(\xi_i^{truth}, w_i^{truth})_{i=1,\dots,\mathcal{N}_h}$ used is hard-coded and formulas are provided in [Solin et al., 2003]. This quadrature rule represents the "truth" quadrature, meaning that it approximates the integral to a certain tolerance:

$$\left| \int_\Omega g(x)\mathrm{d}x - \sum_{i=1}^{\mathcal{N}_h} w_i^{truth} g(\xi_i^{truth}) \right| \leq \varepsilon/2 \tag{3.45}$$

The Empiric Quadrature method consists in optimizing the quadrature rule used for a set of parametrized expressions $\{g_m\}$ of size M. Since the number $\mathcal{N}_h$ is typically very large, it depends on the number of elements in the domain to integrate, we want to find the quadrature points and weights that allow to best approximate the integrals of our expressions.

If we note

$$I_m(\boldsymbol{\mu}) = \int_\Omega g_m(x, \boldsymbol{\mu})\mathrm{d}x \qquad \text{and} \qquad I_m^{truth}(\boldsymbol{\mu}) = \sum_{i=1}^{\mathcal{N}_h} w_i^{truth} g_m(\xi_i^{truth}, \boldsymbol{\mu})$$

then, we want

$$I_m^\nu(\boldsymbol{\mu}) = \sum_{i=1}^{K^\nu} w_i^\nu g_m(\xi_i^\nu, \boldsymbol{\mu}) \tag{3.46}$$

such that

$$\left| I_m^{truth}(\boldsymbol{\mu}) - I_m^\nu(\boldsymbol{\mu}) \right| \leq \varepsilon/2 \quad \forall m, \forall \boldsymbol{\mu} \tag{3.47}$$

and where $K^\nu << \mathcal{N}_h$. This leads to $\left| I_m(\boldsymbol{\mu}) - I_m^\nu(\boldsymbol{\mu}) \right| \leq \varepsilon$, $\forall m$, $\forall \boldsymbol{\mu}$.

During the offline phase, we will train our algorithm on a set of parameters Ξ_J^{train} of size J. We will note $g_{m,j}^i = g_m(\xi_i^{truth}, \boldsymbol{\mu}_j^{train})$. To this end, we need to solve a linear program LP_{quad}^ν where the structural variables are the weights ρ_i, and we seek to minimize their sum:

$$\sum_{i=1}^{\mathcal{N}_h} \rho_i \tag{3.48a}$$

under the constraints that the weights are positive:

$$\rho_i \geq 0, \quad 1 \leq i \leq \mathcal{N}_h \tag{3.48b}$$

and that we are accurate enough on our train set:

$$\left| \sum_{i=1}^{\mathcal{N}_h} w_i^{truth} g_m(\xi_i^{truth}, \boldsymbol{\mu}_j^{train}) - \sum_{i=1}^{\mathcal{N}_h} \rho_i g_m(\xi_i^{truth}, \boldsymbol{\mu}_j^{train}) \right| \leq \varepsilon/2 \qquad \forall 1 \leq j \leq J,\ \forall 1 \leq m \leq M$$

which translate as constraints on the auxiliary variables x_k:

$$\sum_{i=1}^{\mathcal{N}_h} w_i^{truth} g_{m,j}^i - \varepsilon/2 \leq x_k \leq \sum_{i=1}^{\mathcal{N}_h} w_i^{truth} g_{m,j}^i + \varepsilon/2 \qquad \forall 1 \leq j \leq J,\ \forall 1 \leq m \leq M \tag{3.48c}$$

where the MJ auxiliary variables x_k are:

$$x_k = \sum_{i=1}^{\mathcal{N}_h} \rho_i g_{m,j}^i \qquad \forall 1 \leq j \leq J,\ \forall 1 \leq m \leq M \tag{3.48d}$$

The four equations (3.48) define the linear program LP_{quad}^{ν}.

We can see that we need to compute $g_{m,j}^i = g_m(\xi_i^{truth}, \boldsymbol{\mu}_j^{train})$ for all expressions g_m, for all points ξ_i^{truth} and for all parameters $\boldsymbol{\mu}_j^{train}$. This is very time consuming due to the number of evaluations to do. If the expressions are complex, the order of the truth quadrature rule rises and with it, the number of points. As an example, in `Feel++`, the order is set automatically, for a complex expression of order 10, it gives 126 points by elements, and it can rise up to 1001 points by elements for very complex expressions. Multiplied by the number of elements in the domain of integration, the number of evaluations can be well beyond hundreds of millions.

To test our implementations of the method, we will compute the error between the truth quadrature and the empiric quadrature for the following set of expressions:

$$g_0 = x\cos(\alpha) + y\sin(\alpha) \qquad\qquad g_1 = -x\sin(\alpha) + y\cos(\alpha)$$

with $\alpha = \mu_0 x^2 + \mu_1 y^2 + \mu_2 z^2$ and where $\boldsymbol{\mu} = [0.1, 10]^3$ and $\Omega = [0, 1]^3$.
We use a tolerance $\varepsilon = 2e^{-4}$, on a mesh with 84 elements. The order of quadrature determined by `Feel++` is 5, which gives 14 points by elements, so 1176 evaluations and as many constraints for our linear programming problem. The reference time to compute an integral over all the elements is $6e^{-3}$s.
The number of quadrature points optimized K^{ν}, the maximum errors for each expression, e_1 and e_2 over 50 parameters chosen randomly, and the time to compute the integral with respect to the size J of the train set is summarized in table 3.1.

J	10	50	100	250	500	1000
K^{ν}	20	94	144	177	191	199
t	$3e^{-5}$	$1.4e^{-4}$	$2e^{-4}$	$2.4e^{-4}$	$2.6e^{-4}$	$2.8e^{-4}$
e_1	0.2588	0.0463	$2.41e^{-3}$	$5.97e^{-4}$	$3.98e^{-4}$	$1.479e^{-4}$
e_2	0.3152	0.0168	$2.48e^{-3}$	$2.9e^{-4}$	$4.47e^{-4}$	$3.419e^{-4}$

Table 3.1 – Error of the Empirical Quadrature method for different sizes of train set

As expected, the more parameters the quadrature have been trained on, the more precision we get. With a reasonable size for the train set, we are close to the tolerance set. We also see that the time of computation rises with the number of quadrature points used, but stays at least an order of magnitude less than the reference time.

3.3 Conclusion

To conclude, we have a method that allows to compute the solution efficiently or an output of interests for a parametrized problem. This problem can be non-affine or even non-linear thanks to the EIM algorithm, and complex operators can be dealt with its version for discrete operators. With the SER algorithm, we have a way to use the EIM algorithm without the extreme cost associated with non-linear problems. And the EQM algorithm allows us to reduce the cost of computing the integral of a parametrized expression over a domain by limiting the number of quadrature points. Finally, all those algorithms are split between an offline phase where all the heavy computation is done, and an online phase, where the results can be obtained in real time.

Part II

Three dimensional non-linear multi-physics model

A high field magnet involves several different coupled physics. First, the electric current that runs through the magnet will produce heat because of the Joule losses. The temperature will change the thermal and electric conductivity of the copper alloys constituting the magnet. To control the temperature, the magnet will be cooled by a forced water flow. The current density will produce a magnetic field. The thermal dilation and the Lorentz forces will deform the magnet. Finally, the deformation of the magnet may alter the cooling and hence the electric current.

Since modeling directly such a complex problem is very difficult, we chose to make some simplification. First, we place ourselves in a stationary problem, thus eliminating the difficulty of time-dependent problems. Then, the cooling of the magnet involves a fluid mechanic problem that is very expensive to solve. We chose to simplify this problem by introducing heat exchange coefficients derived from standard correlation in thermohydraulics. Finally, we suppose that the deformations are very small and thus we work solely in the reference geometry, rather than the deformed one. If this hypothesis does not hold, we should couple the models via the ALE map to work on the deformed geometry. The simplified physics involved in a high field magnets are presented in the figure 3.5.

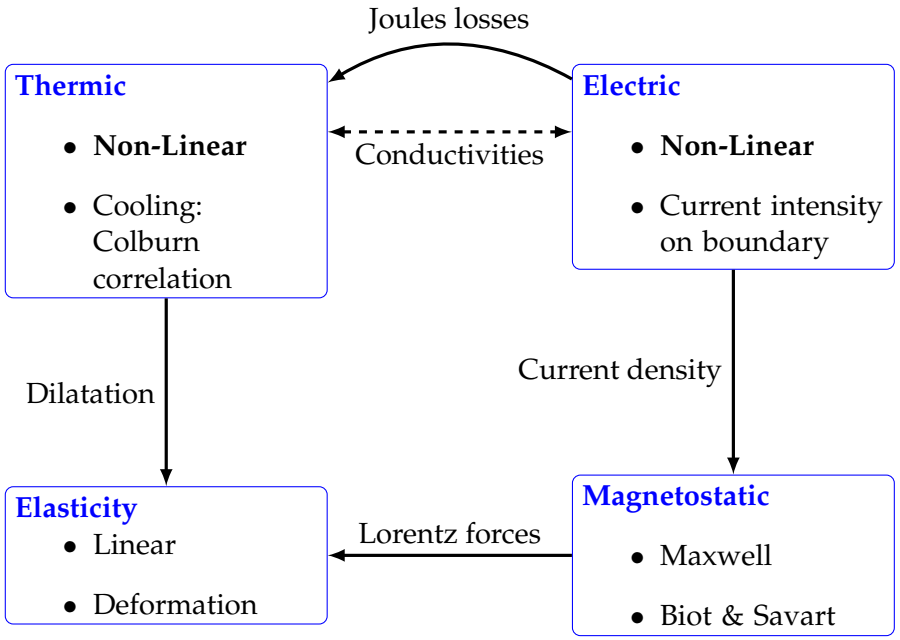


Figure 3.5 – Coupled physics in a high field magnet

In the following part of this thesis, the problems for each of those physics will be detailed. We begin by the thermoelectric problem and its coupling in chapter 4. The CG and HDG formulation are presented as well as some results for the reduced basis. In chapter 5 two methods to compute the magnetic field are detailed. The Maxwell equations are used to solve a problem or a formula given by Biot & Savart can be used, which allows to use the reduced basis method to improve the computation cost. Lastly, the linear elasticity problem is presented in chapter 6. Both the CG and HDG formulations are introduced.

Chapter 4

Electro-Thermic model

Even if electromagnetism is the main physic of interest, other physics are involved in the study of high field magnets. In order to produce the magnetic field, a current density, typically few hundred A/mm^2, is provided to the electromagnet. Joule losses induced by the current create an important increase of heat within the magnet. The temperature can possibly degrade its mechanical properties and lead to thermal dilatation which may damage the electromagnet. Thus, beside an accurate estimation of the current density, an accurate estimate of the temperature is also needed, especially since material properties such as electric or thermic conductivity depend on it.
First we will describe the coupled electro-thermic model used, and in the following sections, present the Continuous Galerkin formulation with its validation, followed by the Hybrid Discontinuous Galerkin formulation of the same problem. Finally, the Reduced Basis method is used to reduce the computational cost in order to use this problem in a many-query situation.

Contents

The Maxwell-Faraday equation states that an electric field, **E**, is proportional to the variation of the magnetic field, **B**, over time. In steady case, due to the properties of

the curl operator, this implies that there exists a scalar potential V such that $\mathbf{E} = -\nabla V$. Furthermore, the Ohm's law tells us that the current density $\mathbf{j}$ is proportional to the electric field $\mathbf{E}$:

$$\mathbf{j} = \sigma \mathbf{E}$$

where σ is the electrical conductivity of the material.
Finally, the charge conservation principle

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{j} = 0$$

where ρ is the charge density, gives, in the steady case, the equation:

$$\nabla \cdot (-\sigma \nabla V) = 0 \tag{4.1}$$

The heat variation can be obtained in two ways. Firstly, by using the first thermodynamic principle, where the work done by the system is neglected because we consider only small deformations.

$$\delta Q = C_p \rho_Q \frac{\partial T}{\partial t} \tag{4.2}$$

where C_p and ρ_Q are respectively the specific heat capacity and the mass density of the material.
Secondly, the heat variation can also be read from the heat flux density $\mathbf{j}_Q$ and the eventual internal sources of heat P

$$\delta Q = \nabla \cdot \mathbf{j}_Q + P \tag{4.3}$$

The Fourier's law defines $\mathbf{j}_Q$ as $\kappa \nabla T$ where κ is the thermal conductivity of the material. The internal source of heat is the Joule effect, we then have $P = \mathbf{j} \cdot \mathbf{E} = \sigma \nabla V \cdot \nabla V$.
In the steady case, (4.2-4.3) give us:

$$-\nabla \cdot (\kappa \nabla T) = \sigma \nabla V \cdot \nabla V \tag{4.4}$$

Both electric and thermic equations (4.1-4.4), involve the respective conductivity, σ and κ. Those conductivities depend on the material but also on the temperature for metal and alloys.

The electrical conductivity describes the capacity of a material to transport electric charges. It is the reciprocal of the resistivity, which measures the opposition to the flow of the electric current. In the case of metals and alloys, the material's resistivity ρ can be expressed in terms of the resistivity at reference temperature T_0 and of the temperature coefficient α. It is linearly dependent on the temperature:

$$\rho(T) = \rho_0(1 + \alpha(T - T_0))$$

Since the conductivity is the inverse of the resistivity, $\sigma_0 = 1/\rho_0$ is the conductivity at reference temperature and

$$\sigma(T) = \frac{\sigma_0}{1 + \alpha(T - T_0)} \tag{4.5}$$

The Wiedemann–Franz law states that the ratio between the thermal and the electric conductivities for metals and alloys is proportional to the temperature, by a constant L known as the Lorenz number. Thus, we can write the thermal conductivity as:

$$\kappa(T) = \sigma(T)LT \tag{4.6}$$

This leads to a non-linearity in the coupled electro-thermal model.

To discuss the boundary conditions associated with the system, we will use the following notations: $\partial\Omega = \Gamma_{in} \cup \Gamma_{out} \cup \Gamma_e = \Gamma_c \cup \Gamma_t$ where each surface has its own boundary condition type, which definition follows.

We consider the cooling water surrounding the magnet as electrically insulating. Thus, we add a homogeneous Neumann condition on these surfaces, noted Γ_e:

- $-\sigma(T)\nabla V \cdot \mathbf{n} = 0$ on Γ_e

In the magnet, to impose the current circulation, a difference of potential is created between current input and output. This difference can be modeled in three different ways:

1. as a Dirichlet condition:
   - $V = 0$ on Γ_{in}
   - $V = V_D$ on Γ_{out}
2. or to be closer to the experiment as a Neumann condition:
   - $V = 0$ on Γ_{in}
   - $-\sigma(T)\nabla V \cdot \mathbf{n} = \dfrac{I}{|\Gamma_{out}|}$ on Γ_{out}
     where I is the input current and $|\Gamma_{out}|$ is the area of the output surface.

An issue with the first type of condition is that to ensure that the current created by the difference of potential has the right intensity, we may need to control its value using a control feedback loop to adjust the difference of potential V_D. This can be done using a Proportional Integral Derivative (PID) controller, but the tuning of the coefficients can be difficult and is problem dependant. The main issue is that it requires also many evaluations of the system. The second type of condition gives the correct intensity directly, but we need to ensure the constantness of the potential on the surface. A solution to those issues is the Integral Boundary Condition presented in section 2.1 of chapter 2 so that the difference of potential is modeled

3. as an IBC:
   - $V = 0$ on Γ_{in}
   - $\displaystyle\int_{\Gamma_{out}} -\sigma(T)\nabla V \cdot \mathbf{n} = I$ and V is constant on Γ_{out}

The impact of the choice of boundary condition will be presented in section 4.2.2.

To model the magnet cooling by a forced water flow, we assume that the thermal flux is proportional to the difference of temperature between the cooling water and the magnet. The factor is called the heat transfer coefficient, h. Hence the cooling of the magnet can be modeled by a Robin condition:

- $-\kappa(T)\nabla T \cdot \mathbf{n} = h(T - T_w)$ on Γ_c

As the exchange originates from a convection phenomenon, h can be deduced from the thermal conductivity, the hydraulic diameter D_h and the Nusselt number Nu by the following relation:

$$h = \frac{\kappa(T_w)Nu}{D_h}$$

In normal operation, the water flow is turbulent and parallel to the surface Γ_c. In this condition, Nu is classically expressed as:

$$Nu = \alpha\, Re^n, Pr^m$$

with Re the Reynolds and Pr the Prandtl number. Values of α, n and m depends from the correlation used. For instance, the Colburn correlation [Colburn, 1933] leads to $\alpha = 0.023, n = 0.8, m = 0.3$.

In practice, magnet designer also often use a simpler correlation derived by Montgomery [Montgomery, 1969]:

$$h = 1426\,(1 + 0.015T_w)\,\frac{u^{0.8}}{/}D_h^{0.2}$$

where u stands for the water flow velocity.

On other surfaces, no heat exchanges are considered. Thus, we use a homogeneous Neumann condition:

- $-\kappa(T)\nabla T \cdot \mathbf{n} = 0$ on Γ_t

Then, the problem is to find V, T such that:

$$\begin{cases} \nabla \cdot (-\sigma(T)\nabla V) = 0 & \text{in } \Omega & (4.7a)\\ \nabla \cdot (\kappa(T)\nabla T) = \sigma(T)\nabla V \cdot \nabla V & \text{in } \Omega & (4.7b)\\ V = 0 & \text{on } \Gamma_{in} & (4.7c)\\ v = V_D & \text{on } \Gamma_{out} & (4.7d)\\ -\sigma(T)\nabla V \cdot \mathbf{n} = 0 & \text{on } \Gamma_e & (4.7e)\\ -\kappa(T)\nabla T \cdot \mathbf{n} = h(T - T_w) & \text{on } \Gamma_c & (4.7f)\\ -\kappa(T)\nabla T \cdot \mathbf{n} = 0 & \text{on } \Gamma_t & (4.7g) \end{cases}$$

4.1 CG formulation

4.1.1 Variational formulation

The variational formulation of (4.7a) reads:

$$\int_\Omega \nabla \cdot (-\sigma(T)\nabla V)\varphi_V = 0 \quad \forall \varphi_V \in X_V \tag{4.8}$$

which gives:

$$\int_\Omega \sigma(T)\nabla V \cdot \nabla \varphi_V - \int_{\partial\Omega} \varphi_V \sigma(T)\nabla V \cdot \mathbf{n} = 0 \quad \forall \varphi_V \in X_V \tag{4.9}$$

Using strong Dirichlet conditions for (4.7c-4.7d) and the definition of (4.7e), we need to find $V \in \{v \in H^1(\Omega) \mid v = 0 \text{ on } \Gamma_{in} \text{ and } v = V_D \text{ on } \Gamma_{out}\}$ such that $\forall \varphi_V \in \{v \in H^1(\Omega) \mid v = 0 \text{ on } \Gamma_{in} \text{ and } v = 0 \text{ on } \Gamma_{out}\}$:

$$\int_\Omega \sigma(T)\nabla V \cdot \nabla \varphi_V - \int_{\Gamma_{in}} \underbrace{\varphi_V}_{=0} \sigma(T)\nabla V \cdot \mathbf{n} - \int_{\Gamma_{out}} \underbrace{\varphi_V}_{=0} \sigma(T)\nabla V \cdot \mathbf{n} - \int_{\Gamma_e} \varphi_V \underbrace{\sigma(T)\nabla V \cdot \mathbf{n}}_{=0} = 0$$

that is find $V \in \{v \in H^1(\Omega) \mid v = 0 \text{ on } \Gamma_{in} \text{ and } v = V_D \text{ on } \Gamma_{out}\}$ such that:

$$\int_\Omega \sigma(T)\nabla V \cdot \nabla \varphi_V = 0 \quad \forall \varphi_V \in \{v \in H^1(\Omega) \mid v = 0 \text{ on } \Gamma_{in} \text{ and } v = 0 \text{ on } \Gamma_{out}\} \tag{4.10}$$

Alternatively, we can impose the Dirichlet conditions weakly by using the Nitsche method [Nitsche, 1971, Freund and Stenberg, 1995]. In this case, instead of using the function space to enforce them, we add two terms containing the weak form of the normal derivative of the solution and the test function, causing the method to be symmetric and consistent. And we also add a term with a positive constant γ such that the term dominates the two others on the boundary to ensure well-posedness, in order to penalize the difference between the solution and the Dirichlet condition. It also depends on the discretization of the domain via the size of the element h_K in order to scale the coefficients. This term causes the method to be stable, and it can be proven that if γ is sufficiently large, the discrete solution converges to the exact one with optimal order.
We need to find $V \in H^1(\Omega)$ such that $\forall \varphi_V \in H^1(\Omega)$:

$$\begin{aligned}\int_\Omega \sigma(T)\nabla V \cdot \nabla \varphi_V - \int_{\Gamma_D} \sigma(T)\left(\nabla V \cdot \mathbf{n}\right)\varphi_V \\ - \int_{\Gamma_D} \sigma(T)\left(\nabla \varphi_V \cdot \mathbf{n}\right) V + \int_{\Gamma_D} \frac{\sigma(T)\gamma}{h_K} V\varphi_V = - \int_{\Gamma_D} \left(\nabla \varphi_V \cdot \mathbf{n}\right) g_D + \int_{\Gamma_D} \frac{\sigma(T)\gamma}{h_K} g_D \varphi_V\end{aligned}$$

Similarly, the variational formulation for (4.7b) reads:

$$\int_\Omega \kappa(T)\nabla T \cdot \nabla \varphi_T - \int_{\Gamma_c} \varphi_T \underbrace{\kappa(T)\nabla T \cdot \mathbf{n}}_{=h(T-T_w)} - \int_{\Gamma_t} \varphi_T \underbrace{\kappa(T)\nabla T \cdot \mathbf{n}}_{=0} = \int_\Omega \sigma(T)\nabla V \cdot \nabla V \varphi_T \quad \forall \varphi_T \in X_T$$

Using the definitions of the boundary conditions (4.7f-4.7g), we need to find $T \in H^1(\Omega)$ such that $\forall \varphi_T \in H^1(\Omega)$:

$$\int_\Omega \kappa(T)\nabla T \cdot \nabla \varphi_T + \int_{\Gamma_c} hT\varphi_T = \int_\Omega \sigma(T)\nabla V \cdot \nabla V \varphi_T + \int_{\Gamma_c} hT_w\varphi_T \tag{4.11}$$

4.1.2 Verification

In this verification case, we will estimate the rise in temperature due to Joules losses in a stranded conductor cooled by a force flow. Electrical potential V_0 and V_1 are respectively applied to the entry and exit of the conductor.
The geometry of the conductor is chosen as to have an analytical expression for the temperature.

4.1.2.1 Problem

The conductor consists of a rectangular cross section torus which is somehow "cut" to allow for applying electrical potential. The conductor is cooled with a force flow along its cylindrical faces.
In 2D, the geometry is as follows:

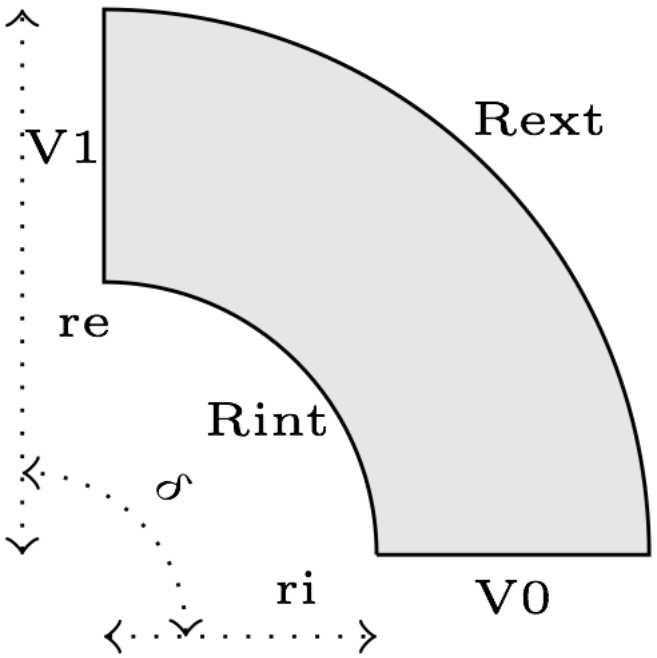


Figure 4.1 – 2D Geometry of the thermoelectric validation

In 3D, this is the same geometry, but extruded along the z axis.

The input parameters are given in table 4.1, and the material properties in table 4.2.

Name	Description	Value	Unit
r_i	internal radius	1	mm
r_e	external radius	2	mm
δ	angle	$\pi/2$	rad
Z	height	1	mm
I	current intensity	-800	A
V_D	electrical potential	0.03125	V
h_i	internal transfer coefficient	0.08	$W \cdot mm^{-2} \cdot K^{-1}$
T_{wi}	internal water temperature	293	K
h_e	external transfer coefficient	0.08	$W \cdot mm^{-2} \cdot K^{-1}$
T_{we}	external water temperature	293	K

Table 4.1 – Input parameters

Name	Description	Marker	Value	Unit
σ	electric conductivity	omega	$58e3$	$S.mm^{-1}$
κ	thermic conductivity	omega	0.38	$W/(mm.K)$

Table 4.2 – Materials

The boundary conditions for the electrical problem are introduced as simple Dirichlet boundary conditions for the electric potential on the entry/exit of the conductor. For the remaining faces, as no current is flowing through these faces, we add Homogeneous Neumann conditions.

Marker	Type	Value
V0	Dirichlet	0
V1	Dirichlet	V_D
Rint, Rext, top*, bottom*	Neumann	0

Table 4.3 – Electric boundary conditions

As for the heat equation, the forced water cooling is modeled by Robin boundary condition with Tw the temperature of the coolant and h a heat exchange coefficient.

Marker	Type	Value
Rint	Robin	$h_i(T - T_{wi})$
Rext	Robin	$h_e(T - T_{we})$
V0, V1, top*, bottom*	Neumann	0

Table 4.4 – Thermal boundary conditions

*: only in 3D

The main fields of concern are the electric potential V, the temperature T and the current density $\mathbf{j}$ or the electric field $\mathbf{E}$.
The analytical solutions in the linear case are given by:

$$V = \frac{V_D}{\delta}\theta = \frac{V_D}{\delta}\,\text{atan2}(y, x)$$
$$\mathbf{E} = \left(-\frac{V_D}{\delta}\frac{y}{x^2+y^2}, \frac{V_D}{\delta}\frac{x}{x^2+y^2}\right)$$
$$T = -A\log\left(\frac{r}{r_0}\right)^2 + T_m$$

with:

$$A = -\frac{\sigma}{2k}\left(\frac{V_D}{\delta}\right)^2$$
$$B = \frac{k}{h_i r_i} + \frac{k}{h_e r_e} + \log\left(\frac{r_e}{r_i}\right)$$
$$C = \log\left(\frac{r_e}{r_i}\right)\log(r_e r_i) + 2\kappa\left(\frac{\log(r_i)}{h_i r_i} + \frac{\log(r_e)}{h_e r_e}\right)$$
$$r_0 = \exp\left(\frac{T_{wi} - T_{we} + AC}{2AB}\right)$$
$$T_m = \frac{2A\kappa}{h_i r_i + h_e r_e}\log\left(\frac{r_e}{r_i}\right) + \frac{h_i r_i T_{wi} + h_e r_e T_{we}}{h_i r_i + h_e r_e} + \frac{A}{h_i r_i + h_e r_e}\left(h_i r_i \log\left(\frac{r_i}{r_0}\right)^2 + h_e r_e \log\left(\frac{r_e}{r_0}\right)^2\right)$$

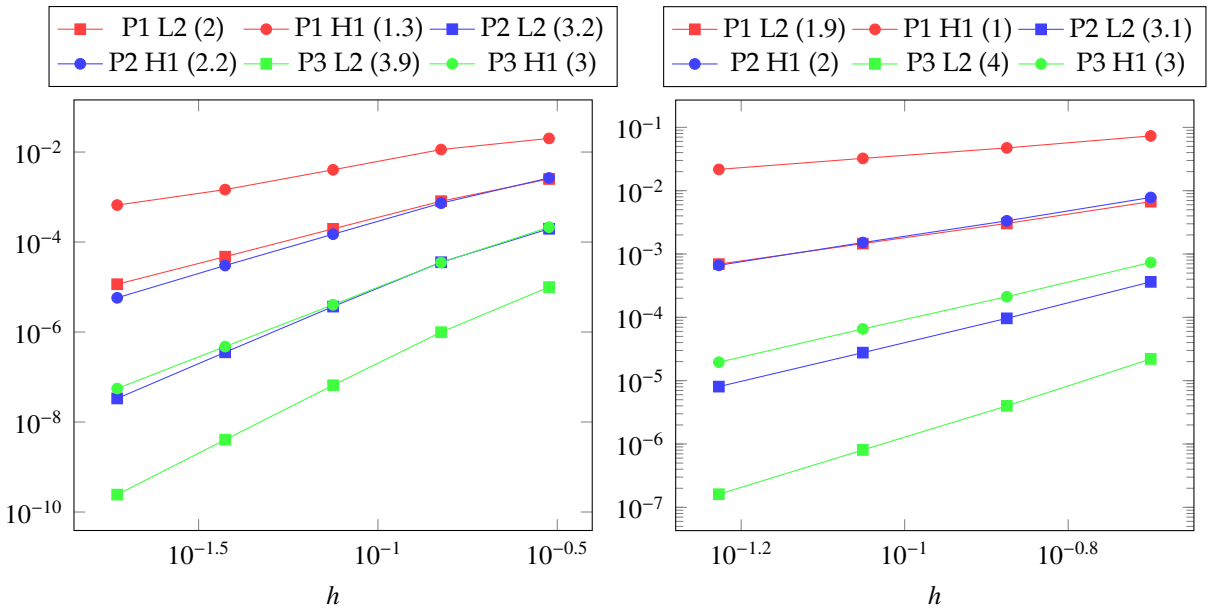
4.1.2.2 Results

We will check if the approximations converge at the appropriate rate: k+1 for the L_2 norm for V and T, k for the H_1 norm for V and T, k for the L_2 norm for $\mathbf{E}$ and $\mathbf{j}$, and k-1 for the H_1 norm for $\mathbf{E}$ and $\mathbf{j}$.

The convergence rates for the temperature, the electric potential and the current density are presented respectively in figure 4.2, 4.3 and 4.4 for 2D and 3D.
We can see that the errors converge at the expected rates, except in 3D for the current density where we do not reach the rate exactly but are close to it.

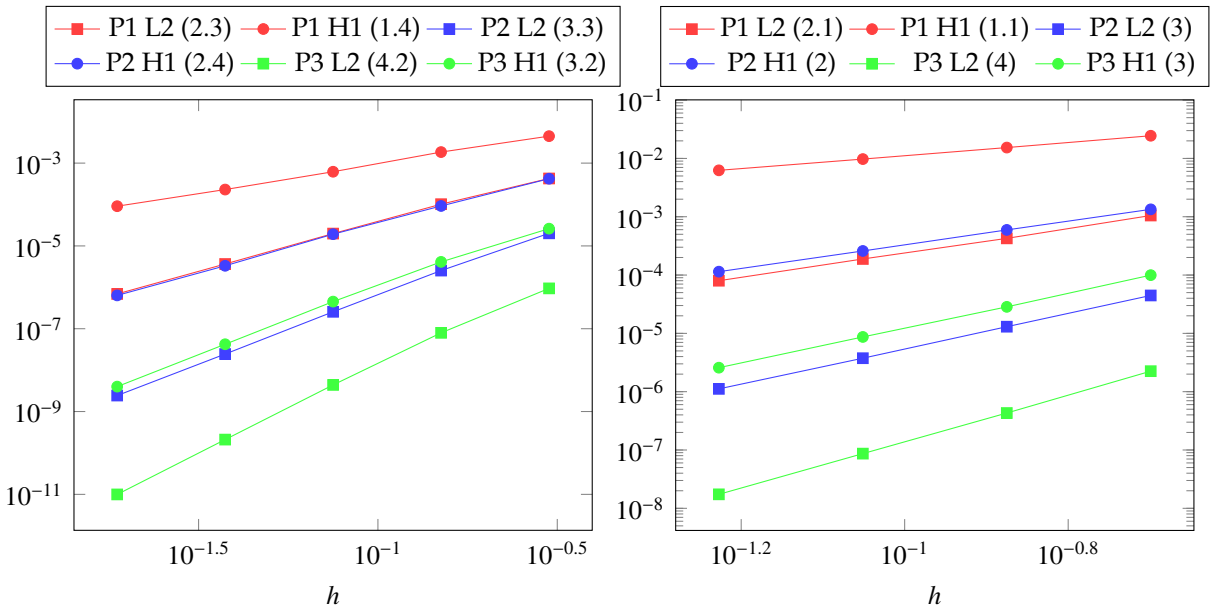
4.2 HDG formulation

Each problem, electric and thermal, can be described by a mixed Poisson equation as seen in chapter 2, where the flux and the potential are the current density and the electric potential for the electric problem and the temperature flux and the temperature for the thermal problem. The non-linearity is handled by a fixed point algorithm, in which we first solve the electric problem to know the Joule's losses and then we can solve the thermal problem. At each step of the algorithm, we then update the conductivities with the previous temperature and solve again the two problems until convergence, that is the increment of the electric potential and the temperature between two steps is less than a certain tolerance.



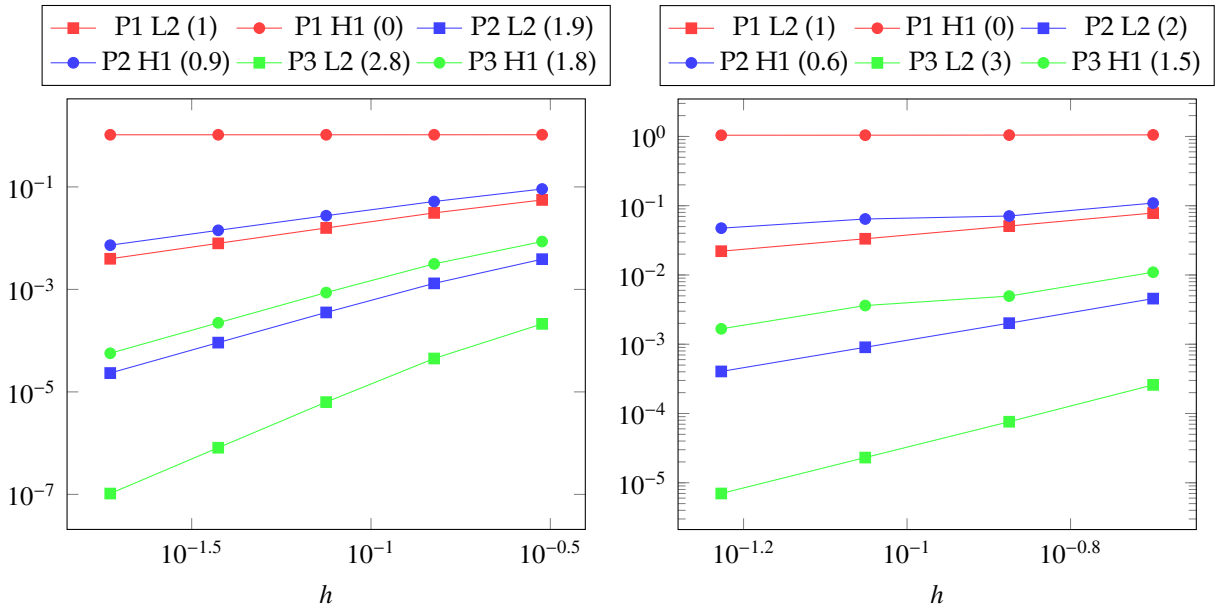
(a) 2D (b) 3D

Figure 4.2 – Temperature convergence for L2 and H1 norm



(a) 2D (b) 3D

Figure 4.3 – Electric potential convergence for L2 and H1 norm



(a) 2D (b) 3D

Figure 4.4 – Current density convergence for L2 and H1 norm

4.2.1 Verification

In this section we will verify our implementation both in the linear case in section 4.2.1.1, and in non-linear case in section 4.2.1.2.

4.2.1.1 Linear case

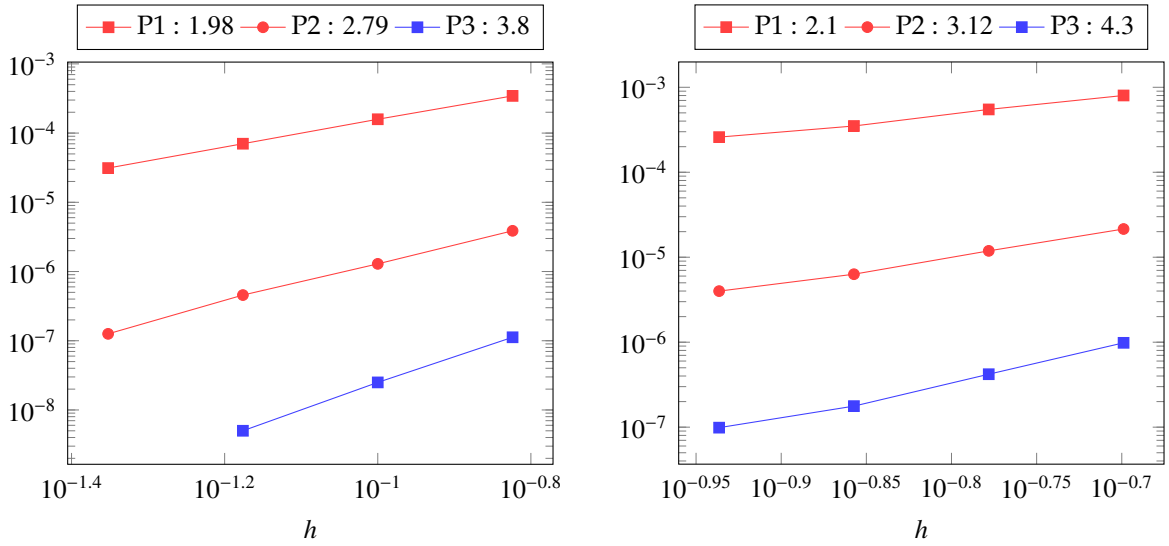
We will use the same problem as in section 4.1.2.1 to verify our implementation. The IBC condition will be tested by replacing the boundary conditions of the electric problem by:

Marker	Type	Value
V0	Dirichlet	0
V1	IBC	I
Rint, Rext, top*, bottom*	Neumann	0

Table 4.5 – Electric boundary conditions

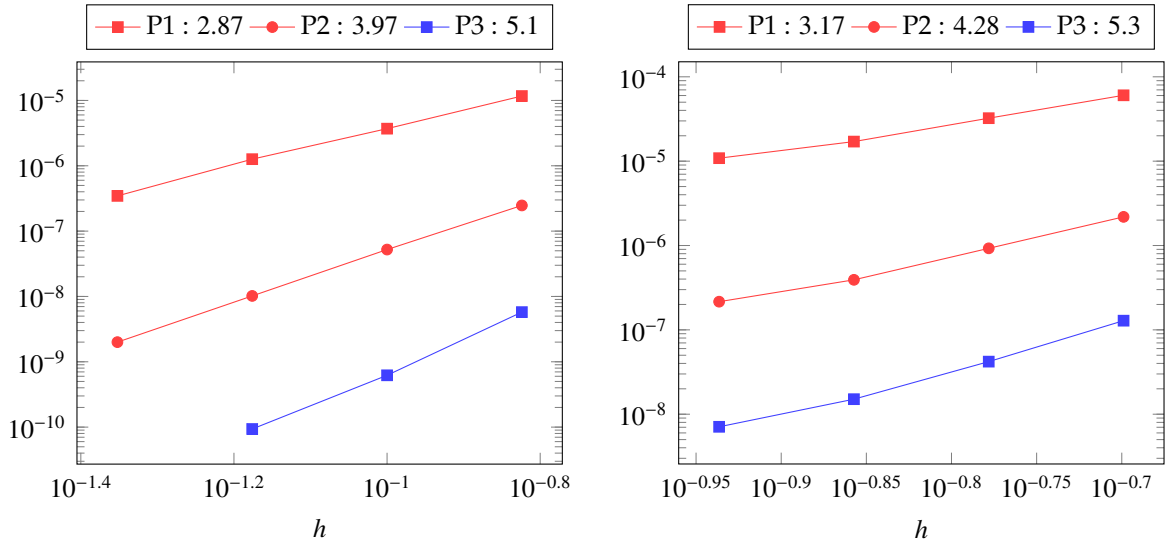
The convergence of the errors in the L_2 norm for the electric potential, post-processed electric potential, current density and temperature are presented respectively in figures 4.5, 4.6, 4.7, 4.8.

We can see that the computed orders of convergence are in agreement with the theoretical predictions from chapter 2.



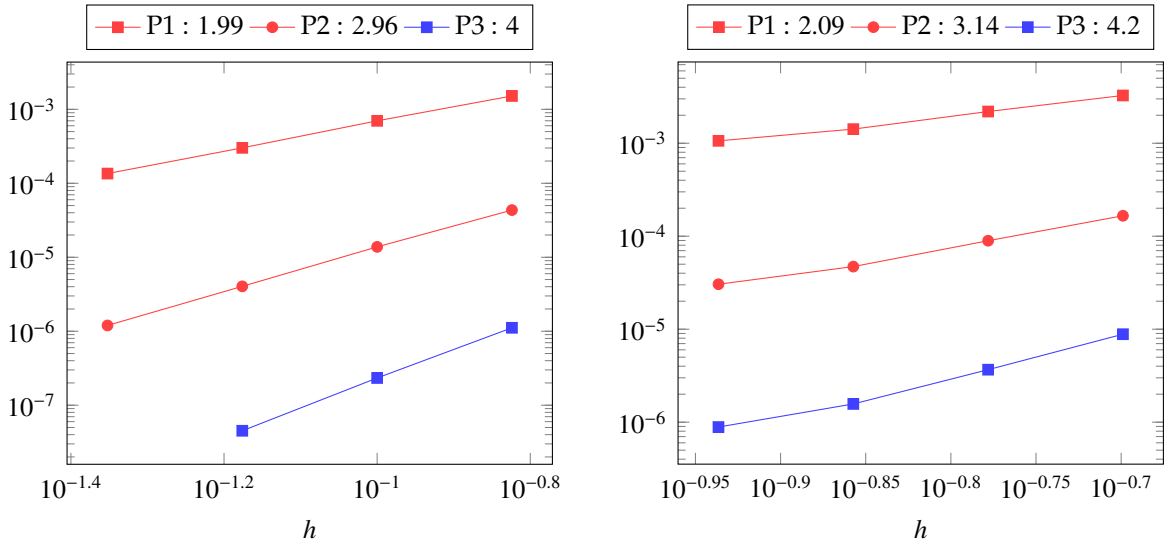
(a) 2D (b) 3D

Figure 4.5 – Electric potential convergence for L2 norm



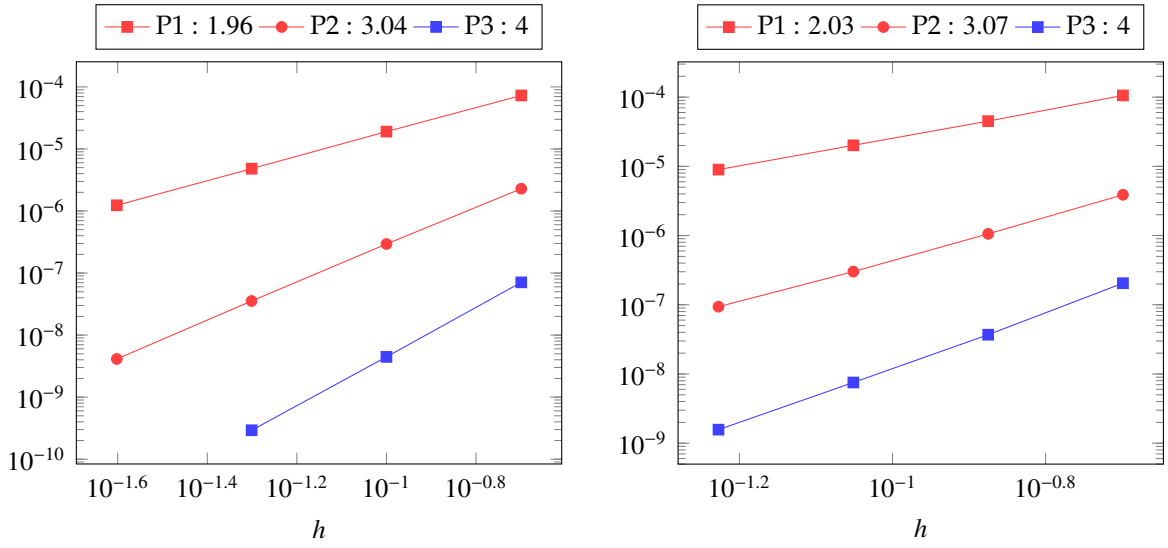
(a) 2D (b) 3D

Figure 4.6 – Post-processed electric potential convergence for L2 norm



(a) 2D (b) 3D

Figure 4.7 – Current density convergence for L2 norm



(a) 2D (b) 3D

Figure 4.8 – Temperature convergence for L2 norm

4.2.1.2 Non-linear case

Here we will test if a non-linear problem is solved correctly using a Picard algorithm. For this we will solve a non-linear mixed poisson problem for which we know a solution. The domain is $[0,1] \times [0,1]$ and the exact solutions are:

$$p = (7x+1)^{\frac{1}{3}} - 1$$
$$u = \left(-\frac{7}{3}, 0\right)$$

for the non-linear conductivity $k = (1+p)^2$. The boundary conditions are summarized in table 4.6.

Type	Domain	Value
Dirichlet	$\{0\} \times [0,1]$	0
Neumann	$[0,1] \times \{0,1\}$	0
IBC	$\{1\} \times [0,1]$	-7/3

Table 4.6 – Boundary conditions for the non-linear test case

The Picard algorithm converged in between 16 and 22 iterations for a tolerance of $1e^{-12}$, whereas the GAMG preconditioner used in the linear solver converged in between 18 and 25 iterations, also for a tolerance of $1e^{-12}$. The least number of iterations is for the polynomial order 1 and the coarsest mesh and the greater number of iterations for the polynomial order 3 and the finer mesh.

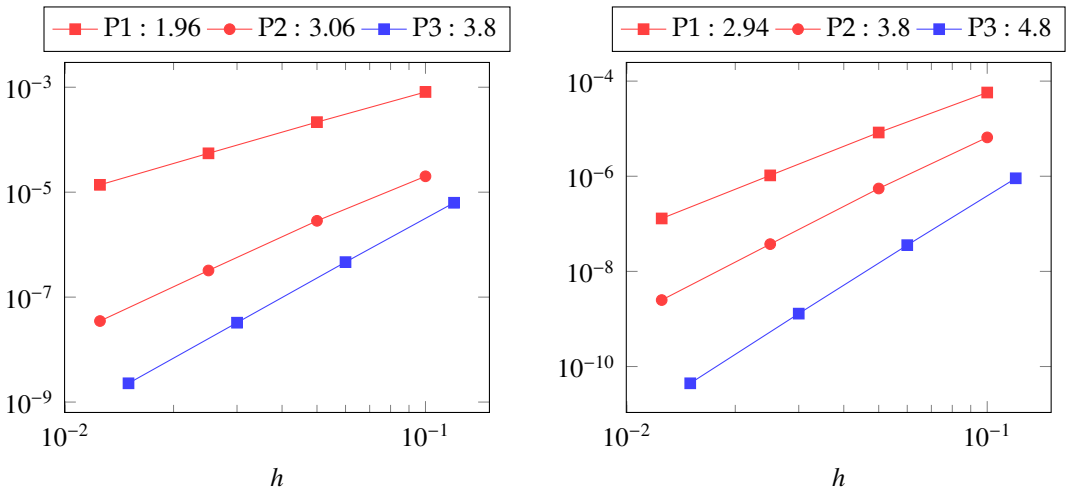
The errors are presented in figure 4.9. We can see that the computed orders of convergence are in agreement with the theoretical predictions from chapter 2, except for the potential and post-processed potential at order 3.

4.2.2 Validation

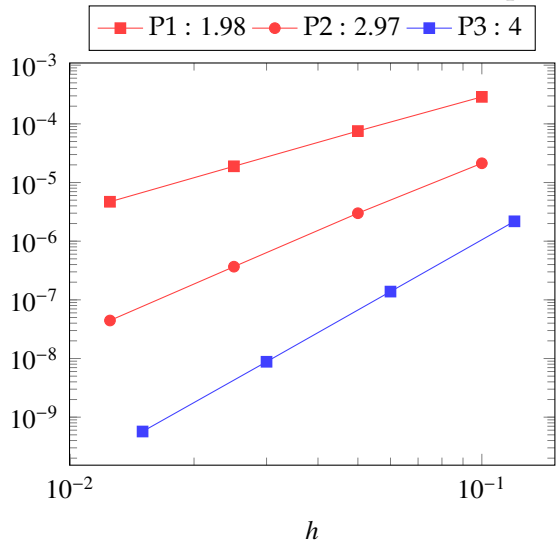
In this section, we want to validate our HDG formulation of the non-linear thermoelectric problem to see if the results are in agreement with the CG formulation. For this, we will look at quantities of interests in our problem on two real geometries, a section of a bitter (see figure 4.10a) which parameters are given in table 4.7, and a helix of a magnet (see figure 4.10b), to see if they agree with the expected values from a physical point of view.

Parameter	Symbol	Units	Value
length of the bitter	l	mm	96
angle between Γ_{in} and Γ_{out}	α	radian	$\pi/18$
diameter of $\Gamma_{cool,1}$	r_1	mm	10
width of $\Gamma_{cool,2}$	w_2	mm	1.1
length of $\Gamma_{cool,2}$	l_2	mm	5.9
height	-	mm	1

Table 4.7 – Geometrical parameters for the bitter.

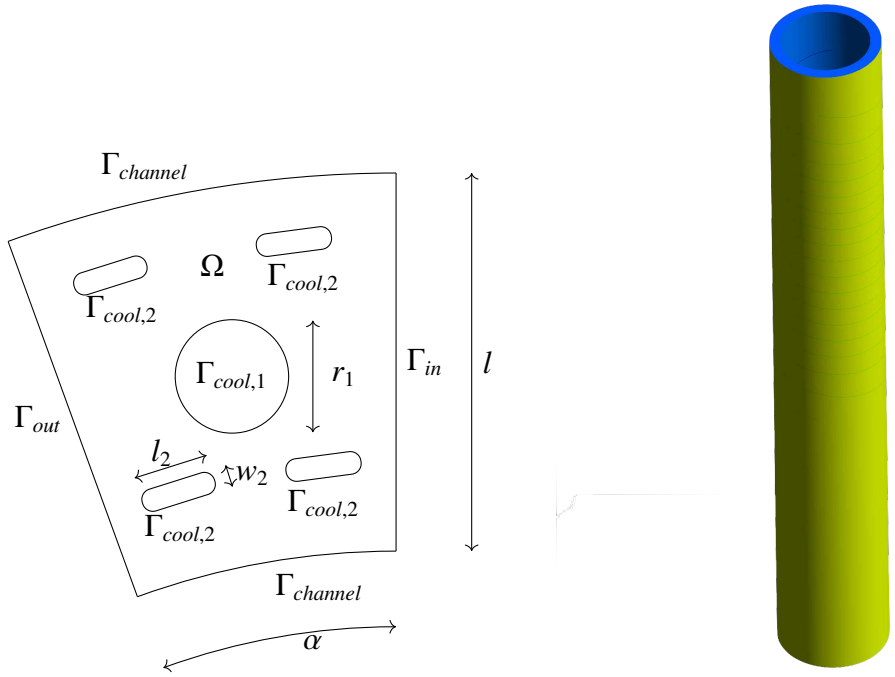


(a) Potential (b) Post-processed potential



(c) Flux

Figure 4.9 – Non linear convergence for L2 norm



(a) A schematic view of a portion of a bitter (b) Helix of a magnet

Figure 4.10 – Geometries of a magnet

The quantities that we are interested in are the minimal, mean and maximal temperature and the current intensity on the input and output surfaces I_0 and I_1 (where the electric potential and the intensity are imposed), the power in the domain P and also the standard deviation (SD) of the electric potential on the surface where the intensity is imposed.
In CG, the intensity is imposed using a Neumann condition, whereas in HDG the intensity is imposed using an Integral Boundary Condition.

		$T_{\max}$	T_{mean}	$T_{\min}$	I_0	I_1	P	SD
Bitter	CG	308.15	302.89	296.12	−3329.42	3329.5	133.49	$1.5e^{-2}$
	HDG	307.18	302.81	296.12	−3333.0	3333.0	133.45	$6.6e^{-7}$
HL-31_H1	CG	408.12	331.1	290.12	−30004.2	30002.16	277823.2	$1.4e^{-4}$
	HDG	408.95	333.03	290.14	−30000.0	30000.0	284183.0	$4.2e^{-5}$

Table 4.8 – Comparison on the quarter turn

In table 4.8, the quantities of interest are the one expected from a physical point of view. Furthermore, we can see that the temperatures are comparable, the difference does not exceed 0.6%. For the intensity, in HDG, we have the exact value imposed, whereas in CG, we have an error of approximately 0.1%. For the bitter, the powers computed in CG and HDG are close (0.03%), but in the case of the helix, the powers do not agree completely but are still relatively close (0.2%). We can also see that the standard deviation of the potential is always greater in CG than when using the IBC in HDG.

Thus, we can validate the HDG formulation for real non-linear thermoelectric problems.

4.3 CRB formulation

In this section, we will describe the affine decomposition of the thermoelectric problem, as well as a way to compute the value of a field at a certain point during the online phase.

4.3.1 Affine decomposition

We want to be able to modify the following parameters in the context of the reduced basis method: the electric conductivity at reference temperature, the temperature coefficient, the Lorenz number, the difference of potential, the heat transfer coefficient and the temperature of the water. Thus, the parameter will be of the form $\boldsymbol{\mu} = (\sigma_0, \alpha, L, V_D, h, T_w) \in \mathbb{P} \subset \mathbb{R}^6$.
We now also need to approximate the conductivities and the right-hand side of (4.7b) with EIM:

$$\sigma(T) \approx \sum_{m_\sigma=1}^{M_\sigma} \beta^\sigma_{m_\sigma}(\boldsymbol{\mu}) q^\sigma_{m_\sigma}, \quad \kappa(T) \approx \sum_{m_k=1}^{M_k} \beta^k_{m_k}(\boldsymbol{\mu}) q^k_{m_k}, \quad \sigma(T)\nabla V \cdot \nabla V \approx \sum_{m_f=1}^{M_f} \beta^f_{m_f}(\boldsymbol{\mu}) q^f_{m_f} \tag{4.12}$$

This leads to the following affine decomposition of the problem:

$$\begin{aligned}
&\sum_{m_\sigma=1}^{M_\sigma} \beta^\sigma_{m_\sigma}(\boldsymbol{\mu}) \int_{\Omega_{cond}} q^\sigma_{m_\sigma} \nabla V \cdot \nabla \phi_V - \sum_{m_\sigma=1}^{M_\sigma} \beta^\sigma_{m_\sigma}(\boldsymbol{\mu}) \int_{\Omega_{cond}} q^\sigma_{m_\sigma}((\nabla V \cdot \mathbf{n})\phi_V - (\nabla \phi_V \cdot \mathbf{n})V + \frac{\gamma}{h} V \phi_V) \\
&\qquad + \sum_{m_k=1}^{M_k} \beta^k_{m_k}(\boldsymbol{\mu}) \int_{\Omega_{cond}} q^k_{m_k} \nabla T \cdot \nabla \phi_T + h \int_{\Omega_{cond}} T \phi_T \\
&= \sum_{m_f=1}^{M_f} \beta^f_{m_f}(\boldsymbol{\mu}) \int_{\Omega_{cond}} q^f_{m_f} \phi_T - \sum_{m_\sigma=1}^{M_\sigma} \beta^\sigma_{m_\sigma}(\boldsymbol{\mu}) V_D \int_{\Omega_{cond}} q^\sigma_{m_\sigma}(\nabla \phi_V \cdot \mathbf{n} - \frac{\gamma}{h}\phi_V) + hT_w \int_{\Omega_{cond}} \phi_V
\end{aligned}$$

It allows us to retrieve the electric potential and the temperature as:

$$V = \sum_{n=1}^{N} V_n(\boldsymbol{\mu}) \xi^V_n \qquad\qquad T = \sum_{n=1}^{N} T_n(\boldsymbol{\mu}) \xi^T_n \tag{4.13}$$

4.3.2 Value at a point

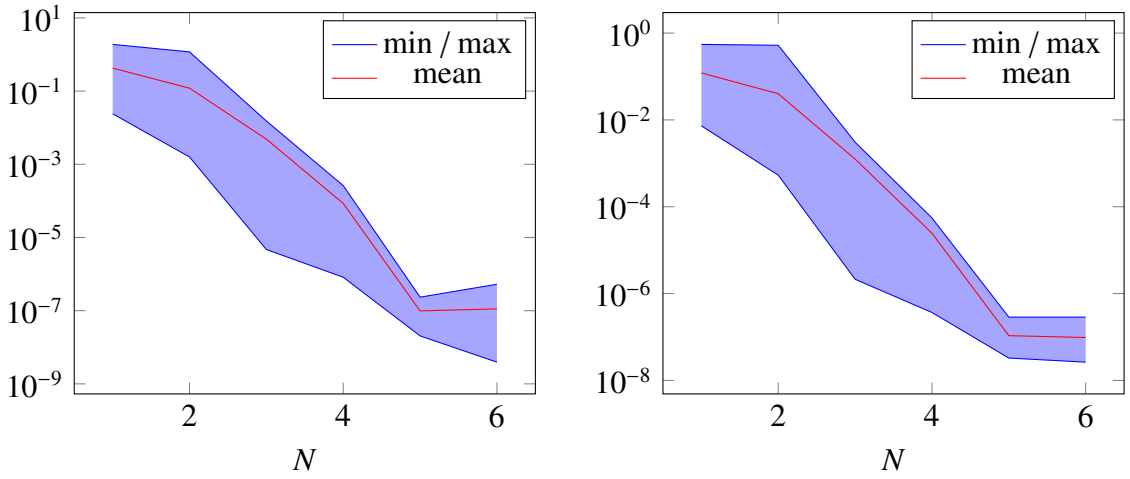
In chapter 9 we will need to compute the value of the electric potential and of the temperature at different points, each corresponding to a sensor. With Galerkin methods, we need to interpolate the value by using the dofs surrounding the point. But this approach would not be feasible during the online phase without computing the high fidelity field first.

So in order to compute those values in real time, we need to compute the value of each reduced basis at those points during the offline phase, what we call the reduced basis context. And during the online phase, we only need to multiply this matrix by the vector of coefficients of our solution for the parameter to have a vector of values for the points.

4.3.3 Results

Here are presented the results on the same problem as in 4.1.2.1, with the parameters described in 4.3.1. We used SER to compute the three EIM, for σ, k and the Joule losses, with a train set of size 1000, and 10 reduced basis chosen randomly.

In figure 4.11a and 4.11b are presented the min, max and average over 50 parameters of the error for V and T respectively, between the reduced solution and the high fidelity one.



(a) Relative error on V

(b) Relative error on T

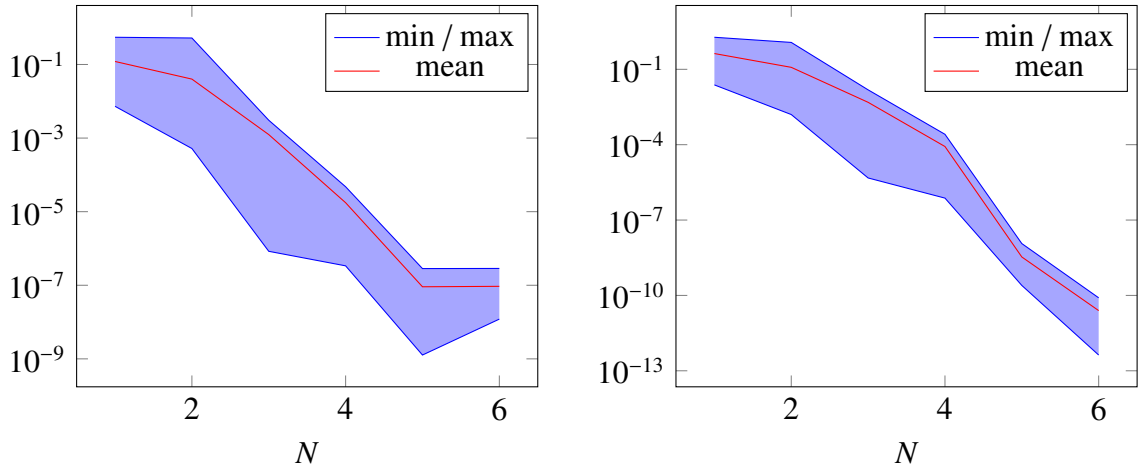
Figure 4.11 – Min, max and mean error over 50 random parameters

The error for the output of the average temperature in the domain is shown in figure 4.12a, whereas in figure 4.12b, we present the error for the value of the temperature on a point.

As we can see, the error rapidly decreases to be around 10^{-8}, both for the two fields and the outputs.

4.4 Conclusion

In this chapter we have described the thermoelectric problem and the boundary conditions associated with it. Both the Continuous Galerkin and the Hybrid Discontinuous Galerkin discretizations are presented, compared and validated using a test case problem corresponding to what we will need to solve to modelize high field magnets. We can see that HDG can reach equivalent errors to CG, or even better errors for flux fields, with a mesh much coarser, but the computational cost is still greater in 3D case.



(a) Relative error on the output

(b) Relative error on the value at a point

Figure 4.12 – Min, max and mean error over 50 random parameters

For the CRB formulation, we have explained the affine decomposition and the choices for the EIM approximations needed. We then presented the results on the same test case. Those results show that our implementation can be used on real problems for high field magnets to more interesting applications.

Chapter 5

Magnetostatic

The electromagnetism is, of course, very important in the modelling of a high field magnet. The magnetic field is directly linked to the current density **j** from the previous chapter. The physic is governed by the Maxwell's equations and has been detailed in [Feynman et al., 2011] for example. To compute the magnetic field in a region carrying no current, we can also use the Biot-Savart law described in [Kratz and Wyder, 2002]. We will first recall the Maxwell's equations and compare two CG formulations to solve them. Then, the Biot-Savart law will be detailed along with methods allowing to take advantage of the reduced basis method previously described.

Contents

5.1 Maxwell

The Maxwell's equations describe the generation of electric and magnetic fields by currents, establishing relations between the electric and magnetic fields (resp. **E** and **H**), the electric and magnetic flux (resp. **D** and **B**), the current density **j** and the electric

charge ρ. They are essential in the study of high field magnets.

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \quad \text{(Faraday)} \tag{5.1a}$$

$$\nabla \times \mathbf{H} = \mathbf{j} + \frac{\partial \mathbf{D}}{\partial t} \quad \text{(Maxwell-Ampère)} \tag{5.1b}$$

$$\nabla \cdot \mathbf{B} = 0 \quad \text{(Gauss magnetic law)} \tag{5.1c}$$

$$\nabla \cdot \mathbf{D} = \rho \quad \text{(Gauss electric law)} \tag{5.1d}$$

In static case, the time derivatives are obviously not considered.

These equations have to be completed with the following constitutive laws, linking **B** to **H** and **D** to **E** from material properties.

$$\mathbf{B} = \mu \mathbf{H}, \quad \mathbf{D} = \varepsilon \mathbf{E} \tag{5.2}$$

where μ is the magnetic permeability, ε the electric permittivity.

The Gauss's magnetic law in (5.1c) leads to the existence of a vectorial magnetic potential **A**

$$\nabla \cdot \mathbf{B} = 0 \Rightarrow \exists \mathbf{A} \text{ such that } \mathbf{B} = \nabla \times \mathbf{A} \tag{5.3}$$

The magnetic potential **A** is not unique since the curl of a gradient is always zero. Thus, if **A** is a potential, then $\mathbf{A} + \nabla\phi$ is also a potential for the same magnetic field **B**.

Combining Maxwell Ampère's equation (5.1b) with the previously introduced constitutive laws (5.2), we have

$$\nabla \times \mathbf{H} = \mathbf{j} \overset{\mathbf{B}=\mu\mathbf{H}}{\Longrightarrow} \nabla \times \left(\frac{1}{\mu}\mathbf{B}\right) = \mathbf{j} \overset{\mathbf{B}=\nabla\times\mathbf{A}}{\Longrightarrow} \nabla \times \left(\frac{1}{\mu}\nabla \times \mathbf{A}\right) = \mathbf{j} \tag{5.4}$$

For magnetostatic problems, the independence of time removes the temporal partial derivatives of (5.1a-5.1b). Then, combining the Maxwell-Ampère equation with (5.2) and (5.3) gives the potential based magnetostatic problem

$$\nabla \times \left(\frac{1}{\mu}\nabla \times \mathbf{A}\right) = \mathbf{j} \tag{5.5}$$

We consider here the classical boundary condition for magnetostatic, which impose the tangential component of magnetic potential on the boundary that reads

$$\mathbf{A} \times \mathbf{n} = \mathbf{A}_D \text{ on } \partial\Omega \tag{5.6}$$

where **n** is the outward unit normal on $\partial\Omega$. Classically $\mathbf{A}_D$ is set to zero on $\partial\Omega$ to mimic the behavior of A at infinity.

The problem (5.5) does not have a unique solution, due to the non-unicity of **A**. However we shall remark that the gradient field $\nabla\phi$ doesn't affect the magnetic flux **B** which is the classical quantity of interest.
Nevertheless, the unicity of the solution is essential for the numerical solving.

The first way to guarantee the solution unicity consists in adding a condition on the divergence, using a gauge. The use of the Coulomb gauge $\nabla \cdot \mathbf{A} = 0$ adds a divergence-free condition, and the unique solution is obtained by solving a saddle point problem. A second way consists in considering the ungauged magnetostatic problem (5.5) as a special case of the potential based full maxwell problem in frequency domain.

5.1.1 Saddle-point

Enforcing the Coulomb gauge (divergence-free condition) to the potential based magnetostatic problem (5.5) can be managed by a scalar Lagrange multiplier p, see [Dumitru, 2013] for more details, giving the following problem to solve

$$\begin{cases} \nabla \times \left(\frac{1}{\mu}\nabla \times \mathbf{A}\right) + \nabla p = \mathbf{j} \quad \text{on } \Omega & \text{(5.7a)} \\ \nabla \cdot A = 0 \quad \text{on } \Omega & \text{(5.7b)} \\ \mathbf{A} \times \mathbf{n} = \mathbf{A}_D \text{ on } \partial\Omega & \text{(5.7c)} \\ p = 0 \quad \text{on } \partial\Omega & \text{(5.7d)} \end{cases}$$

5.1.1.1 Variational formulation

The variational formulation consists in finding $(\mathbf{A}, p) \in (X \subset H_{\text{curl}}(\Omega) \times H_0^1(\Omega))$ such that

$$\int_\Omega \frac{1}{\mu}(\nabla \times \mathbf{A}) \cdot (\nabla \times \mathbf{v}) + \int_{\partial\Omega} \frac{1}{\mu}(\nabla \times \mathbf{A}) \cdot (\mathbf{v} \times \mathbf{n}) + \int_\Omega \mathbf{v} \cdot \nabla p = \int_\Omega \mathbf{j} \cdot \mathbf{v} \quad \forall \mathbf{v} \in X \qquad \text{(5.8a)}$$

$$\int_\Omega \mathbf{A} \cdot \nabla q = 0 \qquad \forall q \in H_0^1(\Omega) \qquad \text{(5.8b)}$$

The Dirichlet boundary condition (5.7c) on $\mathbf{A}$ imposed directly in the definition of the function space cancels the boundary term of (5.8a)

$$X = H_{\mathbf{A}_D,\text{curl}}(\Omega) = \{v \in H_{\text{curl}}(\Omega) \mid v \times \mathbf{n} = \mathbf{A}_D \text{ on } \partial\Omega\}$$

The variational formulation then consists in finding $(\mathbf{A}, p) \in (H_{\mathbf{A}_D,\text{curl}}(\Omega) \times H_0^1(\Omega))$ such that

$$\int_\Omega \frac{1}{\mu}(\nabla \times \mathbf{A}) \cdot (\nabla \times \mathbf{v}) + \int_{\partial\Omega} \frac{1}{\mu}(\nabla \times \mathbf{A}) \cdot A_D + \int_\Omega \mathbf{v} \cdot \nabla p = \int_\Omega \mathbf{j} \cdot \mathbf{v} \quad \forall \mathbf{v} \in X \qquad \text{(5.9a)}$$

$$\int_\Omega \mathbf{A} \cdot \nabla q = 0 \qquad \forall q \in H_0^1(\Omega) \qquad \text{(5.9b)}$$

We can also impose (5.7c) on weak form, as in section 4.1, we add terms for symmetrization and stabilization, see [Assous and Michaeli, 2013] and [Thomée, 2006], and so avoiding to modify the function space $X = H_{\text{curl}}(\Omega)$, to impose directly the tangential component of $\mathbf{A}$. We denote γ the penalisation coefficient, and h_s the mesh size. The

variational formulation then consists in finding $(\mathbf{A}, p) \in (H_{\text{curl}}(\Omega) \times H_0^1(\Omega))$ such that

$$\begin{aligned} &\int_\Omega \frac{1}{\mu}(\nabla \times \mathbf{A}) \cdot (\nabla \times \mathbf{v}) + \int_\Omega \mathbf{v} \cdot \nabla p \\ &+ \int_{\partial\Omega} \frac{1}{\mu}(\nabla \times \mathbf{A}) \cdot (\mathbf{v} \times \mathbf{n}) + \int_{\partial\Omega} \frac{1}{\mu}(\nabla \times \mathbf{v}) \cdot (\mathbf{A} \times \mathbf{n}) \\ &\quad + \int_{\partial\Omega} \frac{\gamma}{h_s}\frac{1}{\mu}(\mathbf{v} \times \mathbf{n}) \cdot (\mathbf{A} \times \mathbf{n}) = \int_\Omega \mathbf{j} \cdot \mathbf{v} + \int_{\partial\Omega} \frac{1}{\mu}(\nabla \times \mathbf{v}) \cdot \mathbf{A}_D \quad \forall \mathbf{v} \in X \\ &\qquad + \int_{\partial\Omega} \frac{\gamma}{h_s}\frac{1}{\mu}(\mathbf{v} \times \mathbf{n}) \cdot \mathbf{A}_D \end{aligned} \tag{5.10a}$$

$$\int_\Omega \mathbf{A} \cdot \nabla q = 0 \qquad \forall q \in H_0^1(\Omega) \tag{5.10b}$$

5.1.1.2 Discretization

The vectorial magnetic potential $\mathbf{A}$ is approximated using Nédélec finite elements of lowest order, and the scalar multiplier p is approximated with Lagrange finite elements of order k. We denote by $V_h \subset X$ and $Q_h \subset H_0^1(\Omega)$ the associated finite element spaces, respectively. Considering $\{\psi_j\}_{j=1}^n$ (resp. $\{\phi_j\}_{i=1}^m$) the finite element basis functions of V_h (resp. Q_h), the discrete approximations $\mathbf{A}_h$ of $\mathbf{A}$ and p_h of p reads

$$\mathbf{A}_h = \sum_{j=1}^n a_j \psi_j \qquad \text{and} \qquad p_h = \sum_{i=1}^n p_i \phi_i \tag{5.11}$$

Replacing $\mathbf{A}$ and p by their discretizations 5.11 in the variational formulations (5.9a) or (5.10a), the discrete magnetostatic saddle-point formulation can be written

$$\begin{pmatrix} \mathcal{A} & \mathcal{B}^T \\ \mathcal{B} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{A} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix} \tag{5.12}$$

If (5.7c) is imposed on the strong form as in (5.9a), the matrix $\mathcal{A}$ reads

$$\mathcal{A}_{i,j} = \int_\Omega \frac{1}{\mu}(\nabla \times \psi_j) \cdot (\nabla \times \psi_i) \tag{5.13}$$

The enforcement of (5.7c) on weak form (5.10a) adds symmetrization and penalisation in $\mathcal{A}$ which then reads

$$\mathcal{A}_{i,j} = \int_\Omega \frac{1}{\mu}(\nabla\times\psi_j)\cdot(\nabla\times\psi_i) + \int_{\partial\Omega} \frac{1}{\mu}(\nabla\times\psi_j)\cdot(\psi_i\times\mathbf{n}) + \int_{\partial\Omega} \frac{1}{\mu}(\nabla\times\psi_i)\cdot(\psi_j\times\mathbf{n}) + \int_{\partial\Omega} \frac{1}{\mu}\frac{\gamma}{h_s}(\psi_j\times\mathbf{n})\cdot(\psi_i\times\mathbf{n}) \tag{5.14}$$

The matrix $\mathcal{B}$ of (5.12) reads

$$\mathcal{B}_{i,j} = \int_\Omega \psi_j \cdot \nabla \phi_i \tag{5.15}$$

and the right-hand side vector $\mathbf{f}$ is the discretization of the source term $\mathbf{j}$ to which symmetrization and penalisation terms can be added

$$\mathbf{f}_i = \int_\Omega \mathbf{j} \cdot \psi_i + \int_{\partial\Omega} \frac{1}{\mu}(\nabla\psi_i) \cdot \mathbf{A}_D + \int_{\partial\Omega} \frac{1}{\mu}\frac{\gamma}{h_s}(\psi_i \times \mathbf{n}) \cdot \mathbf{A}_D \tag{5.16}$$

5.1.2 Regularized formulation

Here we present an alternative to the use of Coulomb gauge. We consider the Maxwell's equations (5.1) given on their full formulation (with time derivatives), which can describe all electromagnetic phenomena.

The time dependency of these equation can be removed using a time Fourier transform

$$\mathcal{F}[f(t)](\omega) : f(t) \longrightarrow \mathcal{F}[f(t)](\omega) = \int_{-\infty}^{+\infty} f(t)e^{-iwt}dt \tag{5.17}$$

The ungauged magnetostatic problem (5.5) is a special case of the time harmonic Maxwell's equations with $\omega = 0$. The full-maxwell problem expressed in frequency domain is regular for all $\omega > 0$, and then has a unique solution. The idea to regularize the magnetostatic problem (5.5) is based on the consideration that the electromagnetic fields obtained at low frequencies are a good approximation of magnetostatics fields and is detailed in [Bebendorf and Ostrowski, 2009].

From Maxwell-Ampère equation (5.1b) in (5.5) and (5.2), we can deduce

$$\nabla \times \left(\frac{1}{\mu}\nabla \times \mathbf{A}\right) = \mathbf{j} + \sigma\mathbf{E} + \varepsilon\frac{\partial \mathbf{E}}{\partial t} \tag{5.18}$$

And from Faraday's law in (5.5), we have :

$$\nabla \times \mathbf{E} = -\frac{\partial(\nabla \times \mathbf{A})}{\partial t} \;\Rightarrow\; \mathbf{E} = -\frac{\partial \mathbf{A}}{\partial t} \tag{5.19}$$

Combining (5.18) and (5.19), we deduce

$$\nabla \times \left(\frac{1}{\mu}\nabla \times \mathbf{A}\right) + \varepsilon\frac{\partial^2 \mathbf{A}}{\partial t^2} + \sigma\frac{\partial \mathbf{A}}{\partial t} = \mathbf{j} \tag{5.20}$$

The Fourier transform of the derivative terms of (5.20) is obtained combining (5.17) with a simple integration by parts, leading to a multiplication with $i\omega$

$$\mathcal{F}[\frac{\partial \mathbf{A}}{\partial t}](\omega) = i\omega\mathcal{F}[A](\omega)$$

We can then deduce the time harmonic equation

$$\nabla \times \left(\frac{1}{\mu}\nabla \times \mathbf{A}\right) + (\sigma i\omega - \varepsilon\omega^2)\mathbf{A} = \mathbf{j} \tag{5.21}$$

Our operator for magnetostatic equation (5.5) can then be "regularized" adding to it a multiple of magnetic potential $\mathbf{A}$, mimicking the last term $(\sigma i\omega - \varepsilon\omega^2)\mathbf{A}$ of (5.21)

$$\nabla \times \left(\frac{1}{\mu}\nabla \times \mathbf{A}_\varepsilon\right) + \alpha\mathbf{A}_\varepsilon = \mathbf{j} \tag{5.22}$$

The solution $\mathbf{A}_\varepsilon$ of the regularized problem (5.22) converges to the solution $\mathbf{A}$ of the initial problem (5.5) for $\varepsilon \longrightarrow 0$, and we can show that

$$\| \mathbf{A}_\varepsilon \|_{L_2} \leqslant \frac{1}{\varepsilon} \| \mathbf{j} \|_{L_2} \quad \text{and} \quad \exists c \text{ such that } \| \mathbf{A} - \mathbf{A}_\varepsilon \|_{H_{curl}} \leqslant c \| \mathbf{j} \|_{L_2} \tag{5.23}$$

5.1.2.1 Variational formulation

The variational formulation obtained from (5.22) consists in finding $\mathbf{A} \in X \subset H_{\text{curl}}(\Omega)$ such that

$$\int_\Omega \frac{1}{\mu}(\nabla \times \mathbf{A}) \cdot (\nabla \times \mathbf{v}) + \int_{\partial\Omega} \frac{1}{\mu}(\nabla \times \mathbf{A}) \cdot (\mathbf{v} \times \mathbf{n}) + \int_\Omega \alpha \mathbf{A} \cdot \mathbf{v} = \int_\Omega \mathbf{j} \cdot \mathbf{v} \quad \forall \mathbf{v} \in Y \subset H_{\text{curl}}(\Omega) \tag{5.24}$$

with X and Y to be determined.

If the Dirichlet boundary condition (5.6) on A is imposed on strong form, it removes the boundary term of (5.24) and the condition is inherent to the function space $X = H_{\mathbf{A}_D,\text{curl}}(\Omega)$. The variational formulation becomes :
Find $\mathbf{A} \in H_{\mathbf{A}_D,\text{curl}}(\Omega)$ such that

$$\int_\Omega \frac{1}{\mu}(\nabla \times \mathbf{A}) \cdot (\nabla \times \mathbf{v}) + \int_\Omega \alpha \mathbf{A} \cdot \mathbf{v} = \int_\Omega \mathbf{j} \cdot \mathbf{v} \quad \forall \mathbf{v} \in H_{0,\text{curl}}(\Omega) \tag{5.25}$$

We can also impose the Dirichlet boundary conditions on weak form, adding symmetrization and penalisation terms and then avoiding adding condition in X function space, i.e. $X = H_{\text{curl}}(\Omega)$. As previously, γ is the penalisation coefficient and h_s the mesh size. The variational formulation consists then in finding $\mathbf{A} \in H_{\text{curl}}(\Omega)$ such that $\forall \mathbf{v} \in H_{\text{curl}}(\Omega)$

$$\begin{aligned}\int_\Omega \frac{1}{\mu}(\nabla \times \mathbf{A}) \cdot (\nabla \times \mathbf{v}) + \int_\Omega \alpha \mathbf{A} \cdot \mathbf{v} + \int_{\partial\Omega} \frac{\gamma}{h_s}\frac{1}{\mu}(\mathbf{v} \times \mathbf{n}) \cdot (\mathbf{A} \times \mathbf{n}) \\ + \int_{\partial\Omega} \frac{1}{\mu}(\nabla \times \mathbf{A}) \cdot (\mathbf{v} \times \mathbf{n}) + \int_{\partial\Omega} \frac{1}{\mu}(\nabla \times \mathbf{v}) \cdot (\mathbf{A} \times \mathbf{n}) = \int_\Omega \mathbf{j} \cdot \mathbf{v} + \int_{\partial\Omega} \frac{1}{\mu}(\nabla \times \mathbf{v}) \cdot \mathbf{A}_D \\ + \int_{\partial\Omega} \frac{\gamma}{h_s}\frac{1}{\mu}(\mathbf{v} \times \mathbf{n}) \cdot \mathbf{A}_D\end{aligned} \tag{5.26}$$

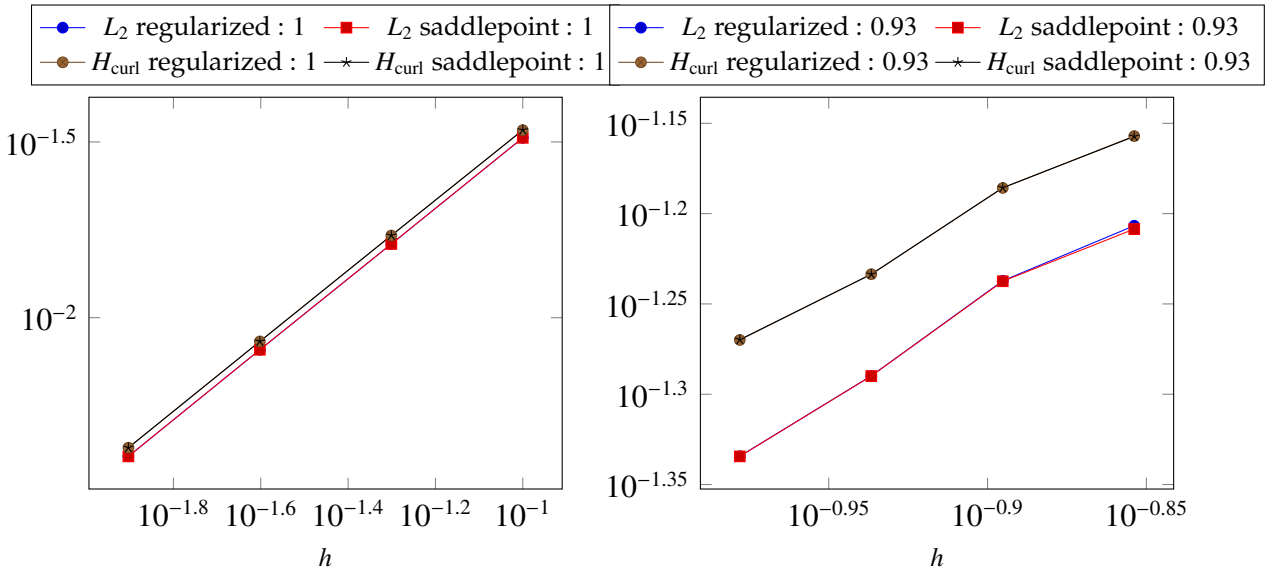
5.1.3 Verification

We need to check the convergence rate of our model. For this we will use the domain $[-1, 1]^d$ where d is the dimension, $d = 2, 3$, and a magnetic permeability $\mu = 1$. The following analytical solution in 2D and 3D will be used:

$$A = \begin{pmatrix} 1 - y^2 \\ 1 - x^2 \end{pmatrix} \qquad A = \begin{pmatrix} (1 - y^2)(1 - z^2) \\ (1 - x^2)(1 - z^2) \\ (1 - x^2)(1 - y^2) \end{pmatrix}$$

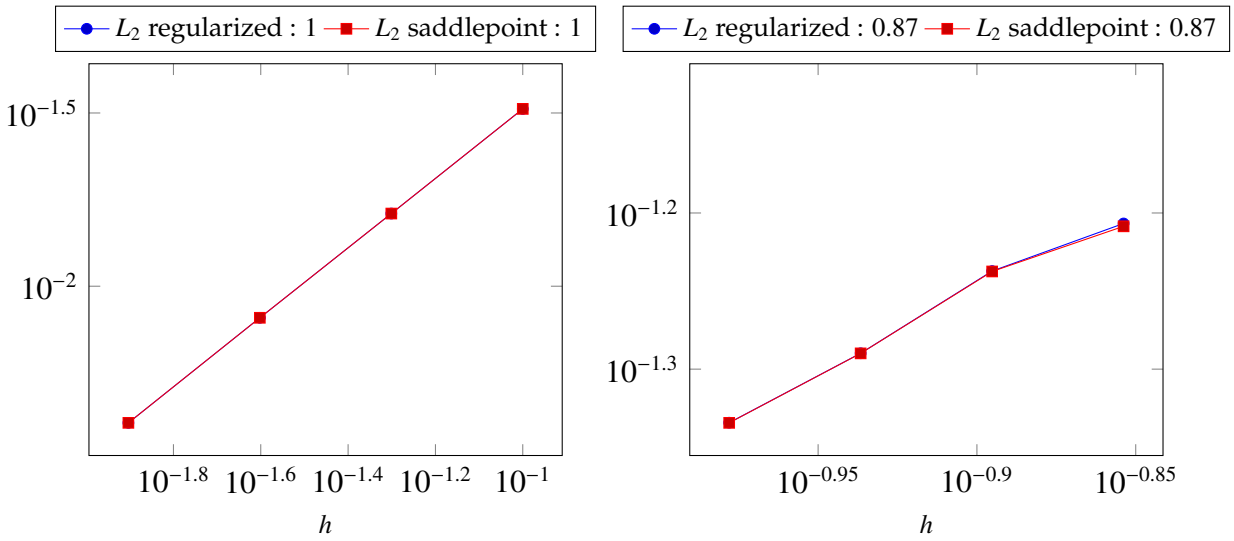
$$B = -2x + 2y \qquad B = \begin{pmatrix} 2(x^2 - 1)(y - z) \\ -2(y^2 - 1)(x - z) \\ 2(z^2 - 1)(x - y) \end{pmatrix}$$

Note that in 2D, the magnetic field B is a scalar field, since the curl of a 2D vectorial field (u_1, u_2) is defined as $\partial_x u_2 - \partial_y u_1$.



(a) 2D (b) 3D

Figure 5.1 – Convergence of the magnetic potential for L_2 and H_{curl} norm with regularized and saddle point formulation



(a) 2D (b) 3D

Figure 5.2 – Convergence of the magnetic field for L_2 norm with regularized and saddle point formulation

In figures 5.1 and 5.2, representing respectively the convergence of the magnetic potential A and of the magnetic field B, we can see that there is no difference between the two methods presented. Both methods reach the order of convergence predicted by the theory.

The use of Nedelec elements, whose degrees of freedom are located on the edges of the mesh, has a high computational cost due to the bad conditioning of the matrix representing the bilinear form. This is especially the case for the saddle point formulation since the system to solve is larger than with the regularized formulation due to the Lagrange multiplier. The resolution of such problems need adapted preconditioning techniques to ease the convergence of the iterative solvers, particularly in 3D. One important condition for those preconditioners to work best is to have the condition $\nabla \cdot \mathbf{j} = 0$ imposed correctly. As we saw in section 2.3, the HDG method allows us to control much more effectively this condition than with the CG method. Such preconditioners have been described in [Greif and Schötzau, 2007] or [Hiptmair and Xu, 2007] and have been implemented in `Feel++`.

5.1.4 Validation

In this example, we will compute the magnetic field generated by a stranded conductor. The geometry of the conductor is chosen such that we can derive the analytical expression of the magnetic field.

5.1.4.1 Problem

The conductor Ω consists in a rectangular cross-section torus, as in section 4.1.2.1, considered only in 3D. The geometry also contains an external domain which is an approximation of $\mathbb{R}^3 \setminus \Omega$.

Name	Description	Value	Unit
r_i	internal radius	1.10^{-3}	m
r_e	external radius	2.10^{-3}	m
H	half height	$2.5.10^{-3}$	m

Table 5.1 – Geometrical parameters

The constant current density $\mathbf{j}$, in A/m^2 can be viewed as an input parameter, or given by the thermoelectric model.
The boundary conditions are given below:

- $\mathbf{A} = \mathbf{0}$ at the infinity
- $\mathbf{A} \times \mathbf{n} = 0$ on symetry plane

For simple conductor geometry, analytical expressions of the magnetic field along the Z-Axis may be found in physics textbooks, like [Jackson, 1999]. The expression of the magnetic field in $\mathbb{R}^3$ is more difficult to derive but may be found in several papers.

As a classical result, we consider only the magnetic field along the Z-Axis, which analytical expression is given below:

$$B_z(z) = \frac{1}{2}\mu_0 J \left[\left[log(r^2 + t^2) \right]_{r=r_1}^{r=r_2} \right]_{t=z-H}^{t=z+H}$$

In figure 5.3, we show the analytical expression and the result of the regularized formulation along the z-axis. We can see that the results are in accordance with the expected expression.

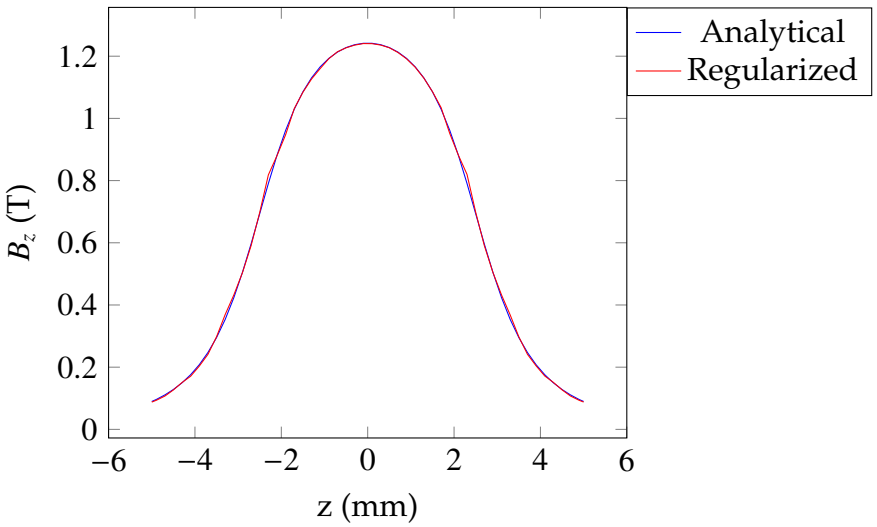


Figure 5.3 – Analytical and numerical value of $\mathbf{B}_z$ along the z-axis

5.2 Biot & Savart

In the case of a steady current $\mathbf{j}$ in a conductor Ω_C, Jean-Baptiste Biot and Félix Savart discovered in 1820 an equation describing the magnetic induction $\mathbf{B}$. The latter can be determined in the domain Ω_{mgn}, where Ω_C and Ω_{mgn} does not intersect.

The equation (5.5) can be seen as a Poisson equation, under the conditions that the magnetic permeability is constant and that we enforce the divergence-free condition with a Coulomb gauge. Using the general solution of this equation, the Green's function $G(\mathbf{x}, \mathbf{r}) = \frac{-1}{4\pi} \frac{1}{|\mathbf{x} - \mathbf{r}|}$, [Jackson, 1999], we can express the magnetic potential $\mathbf{A}$ as:

$$\mathbf{A}(\mathbf{x}) = \frac{\mu_0}{4\pi} \int_{\Omega_C} \frac{\mathbf{j}(\mathbf{r})}{|\mathbf{x} - \mathbf{r}|^3} d\mathbf{r} \quad \forall \mathbf{x} \in \Omega_{mgn}$$

where μ_0 is the vaccum permeability.

From the definition of $\mathbf{B}$ in (5.3), the magnetic field at a point $\mathbf{x}$ is given by:

$$\mathbf{B}(\mathbf{x}) = \frac{\mu_0}{4\pi} \int_{\Omega_C} \frac{\mathbf{j}(\mathbf{r}) \times (\mathbf{x} - \mathbf{r})}{|\mathbf{x} - \mathbf{r}|^3} d\mathbf{r} \quad \forall \mathbf{x} \in \Omega_{mgn} \tag{5.27}$$

5.2.1 Parallel algorithm

Although the computation of this quantity is straightforward in sequential, if the domains Ω_C and Ω_{mgn} are partitioned, we need to devise a parallel algorithm to compute efficiently $\mathbf{B}$.

Let's write the partionning of the domains as:

$$\Omega_C = \bigcup_i \Omega_C^i \qquad \text{and} \qquad \Omega_{mgn} = \bigcup_j \Omega_{mgn}^j$$

We can parallelize the computation on n processors by splitting the integral on the Ω_C^i:

$$\mathbf{B}(\mathbf{x}) = \sum_{i=1}^{n} \frac{\mu_0}{4\pi} \int_{\Omega_C^i} \frac{\mathbf{j}(\mathbf{r}) \times (\mathbf{x} - \mathbf{r})}{|\mathbf{x} - \mathbf{r}|^3} d\mathbf{r} \qquad \forall \mathbf{x} \in \Omega_{mgn}^j \tag{5.28}$$

$$= \sum_{i=1}^{n} \mathbf{B}^i(x) \qquad \forall \mathbf{x} \in \Omega_{mgn}^j \tag{5.29}$$

Thus, to parallelize the computation, we need to be able to transfer data between the processors p_j^{mgn} in which belongs the degrees of freedom of Ω_{mgn}^j and all the processors p_i^C in which belongs Ω_C^i. In parallel computing, namely with `mpi`, the interprocess transfers are handled by so-called communicators.

From this, the algoritm reads as:

1. create subcommunicators $\{p_j^{mgn}, p_1^C, \ldots, p_n^C\} \forall j$, so that processors containing degrees of freedom in Ω_{mgn} can share data with processors containing Ω_C
2. each p_j^{mgn} shares its degrees of freedom of Ω_{mgn}^j in its subcommunicators with all processors p_i^C
3. each proc p_i^C computes $\mathbf{B}^i(\mathbf{x})$ for every $\mathbf{x}$ in Ω_{mgn}^j
4. reduce operation on local contributions $\mathbf{B}^i(\mathbf{x})$: transfer back the values of $\mathbf{B}^i(\mathbf{x})$ to p_j^{mgn} and sum them

Remark. The computation of $\mathbf{B}^i(\mathbf{x})$ can be optimized by computing only once the geometrical transformation and quadrature points for every degree of freedom. And when using Lagrange's finite elements, three degrees of freedom (in 3D) are in fact associated with one point, hence we need to do the precomputation only for each point $\mathbf{x}$.

The performances of this algorithm have been tested on Curie supercomputer (TGCC, France) [Daversin Catty, 2016]. The theoretical speed up should be that with n times more processors, we gain a factor n in computational time. The speed up is defined as:

$$\frac{t_{ref} * n_{ref}}{t * n}$$

where t_{ref} and n_{ref} are reference time and number of processors respectively.

The speed up obtained with this algorithm is shown in figure 5.4. We can see that with optimized computation of the integrals, we reach the ideal speed-up for any number of processors.

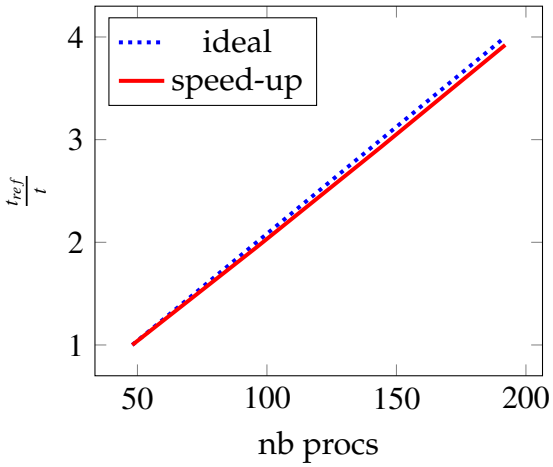


Figure 5.4 – Biot-Savart computation : Speed-up

5.2.2 RB formulation

We want to use the CRB approximation of the thermoelectric problem to be able to compute the magnetic field **B** efficiently. For this, we need to consider two cases, one if the parameters $\boldsymbol{\mu}$ are experimental (current intensity, water temperature,...), and one if the parameters are geometrical. The two cases are presented in the following sections.

5.2.2.1 No geometric parameter

This is the simplest case. We take advantage of the fact that the current density **j** can be written in terms of the approximations of $\sigma(\mathbf{r},\boldsymbol{\mu})$ and $V(\mathbf{r},\boldsymbol{\mu})$:

$$\sigma(\mathbf{r},\boldsymbol{\mu}) \approx \sum_{m_\sigma=1}^{M_\sigma} \beta_{m_\sigma}^{\sigma}(\boldsymbol{\mu}) q_{m_\sigma}^{\sigma}(\mathbf{r})$$

$$V(\mathbf{r},\boldsymbol{\mu}) \approx \sum_{n=1}^{N} V_n(\boldsymbol{\mu}) \xi_n^V(\mathbf{r})$$

$$\mathbf{j}(\mathbf{r},\boldsymbol{\mu}) = -\sigma(\mathbf{r},\boldsymbol{\mu})\nabla V(\mathbf{r},\boldsymbol{\mu}) \approx -\sum_{n=1}^{N}\sum_{m_\sigma=1}^{M_\sigma} V_n(\boldsymbol{\mu})\beta_{m_\sigma}^{\sigma}(\boldsymbol{\mu}) q_{m_\sigma}^{\sigma}(\mathbf{r})\nabla\xi_n^V(\mathbf{r}) \tag{5.30}$$

Injecting this approximation of $\mathbf{j}$ into the Biot & Savart law (5.27) gives:

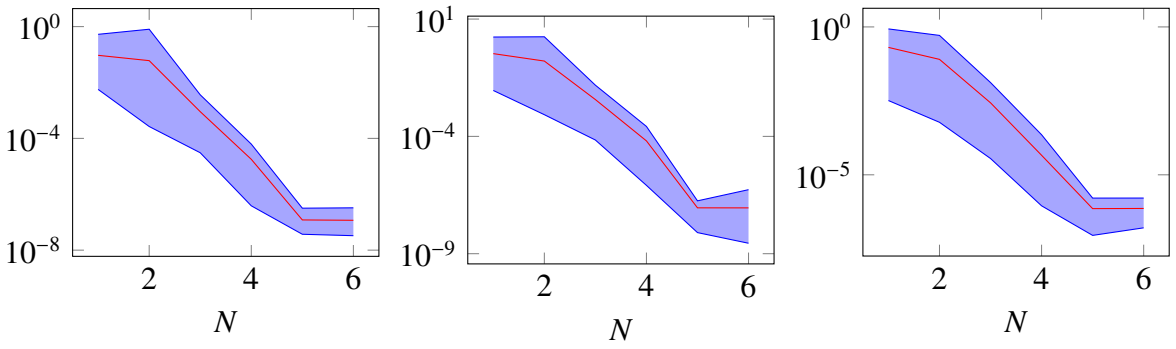
$$\begin{aligned}\mathbf{B}(\mathbf{x};\boldsymbol{\mu}) &= \frac{\mu_0}{4\pi}\int_{\Omega_C}\frac{-\sigma(\mathbf{r},\boldsymbol{\mu})\nabla V(\mathbf{r},\boldsymbol{\mu})\times(\mathbf{x}-\mathbf{r})}{|\mathbf{x}-\mathbf{r}|^3}d\mathbf{r}\\ &= \sum_{n=1}^{N}\sum_{m_\sigma=1}^{M_\sigma} V_n(\boldsymbol{\mu})\beta^\sigma_{m_\sigma}(\boldsymbol{\mu})\underbrace{\frac{\mu_0}{4\pi}\int_{\Omega_C}\frac{-q_{m_\sigma}(\mathbf{r})\nabla\xi^V_n(\mathbf{r})\times(\mathbf{x}-\mathbf{r})}{|\mathbf{x}-\mathbf{r}|^3}d\mathbf{r}}_{\xi^{\mathbf{B}}_{n,m_\sigma}(\mathbf{x})} \qquad (5.31)\\ &= \sum_{k=1}^{N_B} B_k(\boldsymbol{\mu})\xi^B_k(\mathbf{x}) \quad \forall\mathbf{x}\in\Omega_{mgn}\end{aligned}$$

with $B_k(\boldsymbol{\mu}) = V_n(\boldsymbol{\mu})\beta^\sigma_{m_\sigma}(\boldsymbol{\mu})$ and $\boldsymbol{\xi}^B_k = \boldsymbol{\xi}^B_{n,m_\sigma}$ for $k = (m_\sigma - 1)N + n$ and $N_B = NM_\sigma$.

The functions $\boldsymbol{\xi}^B_k$, having no dependence on $\boldsymbol{\mu}$, can be pre-computed during the offline phase. This allows, during the online phase, the computation in real time for any parameter $\boldsymbol{\mu}$ by only computing the coefficients $V_n(\boldsymbol{\mu})$ and $\beta^\sigma_{m_\sigma}(\boldsymbol{\mu})$, and thus $B_k(\boldsymbol{\mu})$.

We will test our implementation on the same problem as in section 4.1.2.1 where we compute the magnetic field $\mathbf{B}$ in a sphere centered at $(0,0,0)$. The parameters are σ,L,α,h,T_w. We use EIM with a trainset of 1000 parameters, but we will use SER to compute only N+1 FE solutions. We use the empirical errors $\|u_n(\boldsymbol{\mu}) - u_{n-1}(\boldsymbol{\mu})\|$ to choose the next parameter for the reduced basis, with a tolerance on the mean temperature of 10^{-3}. We only need 6 basis to reach this tolerance, thus we only need 7 FE resolutions with SER instead of 1006.

The convergences for $V, T, \mathbf{B}$ are presented in figure 5.5, the relative errors with respect to the FEM solution are computed over 50 parameters chosen randomly. We can



(a) Error on T (b) Error on V (c) Error on B

Figure 5.5 – Mean error (red) and min/max error (blue) for 50 random parameters

see that with only 6 basis we do not exceed 2.10^{-6} for $\mathbf{B}$.

5.2.2.2 Geometric parameter (DEIM)

In the case of geometrical parameters, it is more difficult to use the reduced basis method because of the distance in the denominator in the integral. Indeed if the geometry of the conductor depends on the parameters $\boldsymbol{\mu}$ by a geometric transformation

ϕ_μ, the distance between the point $\mathbf{x} \in \Omega_{mgn}$ and a point $\mathbf{r} \in \Omega_C$ also depends on the parameters $\boldsymbol{\mu}$.

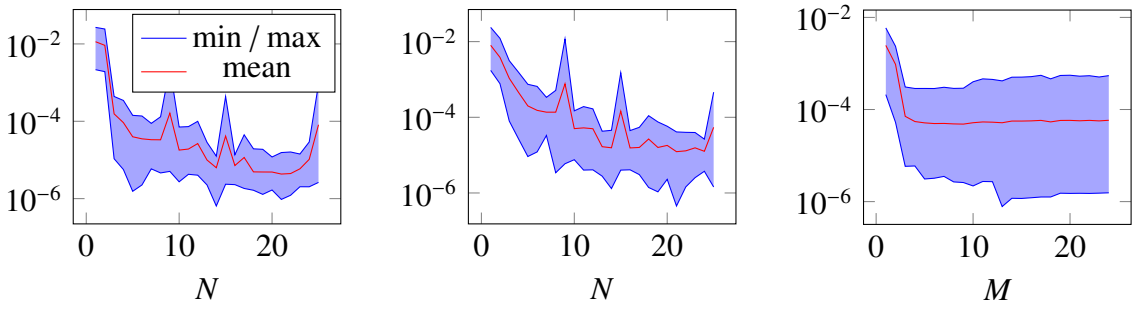
$$\begin{aligned}\mathbf{B}(\mathbf{x},\boldsymbol{\mu}) &= \frac{\mu_0}{4\pi}\int_{\phi_\mu(\Omega_C)} \frac{\mathbf{j}\times(\mathbf{x}-\mathbf{r})}{|\mathbf{x}-\mathbf{r}|^3}d\mathbf{r} \qquad \forall \mathbf{x} \in \Omega_{mgn}\\ &= \frac{\mu_0}{4\pi}\int_{\Omega_C} \frac{-\sigma\nabla V \cdot J_{\phi_\mu}^{-1}\times(\phi_\mu(\mathbf{x})-\mathbf{r})}{|\phi_\mu(\mathbf{x})-\mathbf{r}|^3}\left|J_{\phi_\mu}\right|d\mathbf{r}\end{aligned}$$

Thus, using EIM to remove this dependence is not an option, because we would need to have a different EIM for each point $\mathbf{x}$ in Ω_{mgn}. This is simply not possible due to the number of points in the domain.

The solution is to use the discrete version of EIM, see 3.2.2, to approximate the whole field $\mathbf{B}$.

$$\mathbf{B}(\mathbf{x},\boldsymbol{\mu}) \approx \mathbf{B}^{M_B}(\mathbf{x},\boldsymbol{\mu}) = \sum_{m=1}^{M_B}\Theta_m^B(\boldsymbol{\mu})\mathbf{q}_m^B(\mathbf{x})$$

Using the geometrical transformation presented in section 10.1, figure 5.6 shows different errors on V and $\mathbf{B}$ using the discrete version of EIM. Figure 5.6a plots the relative error of V with respect to the number N of reduced basis used, with DEIM finding the exact decomposition for the linear and bilinear forms. Figure 5.6b shows the relative error of $\mathbf{B}$ computed on the whole domain Ω_{mgn}, using DEIM only to compute V. The error is plotted against the number of reduced basis used for V. Figure 5.6c presents the relative error of $\mathbf{B}^{M_B}$ computed with DEIM, and using V computed with 25 reduced basis and DEIM. The relative errors with respect to the FEM solution are plotted against the number of EIM basis M_B.



(a) Relative error on V (b) Relative error on $\mathbf{B}$ (c) Relative error on $\mathbf{B}^{M_B}$

Figure 5.6 – Min, max and mean errors on the fields, V, $\mathbf{B}$ and $\mathbf{B}^{M_B}$, on 50 parameters

Results with randomly chosen basis, hence the jumps on the errors for V and $\mathbf{B}$. The error on $\mathbf{B}^{M_B}$ shows that we do not need a great number of basis for DEIM to reach an error corresponding to the error on V.

When using EIM, during the online phase we need to solve the problem:

$$\begin{pmatrix}\mathbf{q}_1(\mathbf{t}_1) & \dots & \mathbf{q}_1(\mathbf{t}_{M_B})\\ & \ddots & \vdots \\ 0 & & \mathbf{q}_{M_B}(\mathbf{t}_{M_B})\end{pmatrix}\begin{pmatrix}\Theta_1(\boldsymbol{\mu})\\ \vdots \\ \Theta_{M_B}(\boldsymbol{\mu})\end{pmatrix} = \begin{pmatrix}\mathbf{B}(\mathbf{t}_1,\boldsymbol{\mu})\\ \vdots \\ \mathbf{B}(\mathbf{t}_{M_B},\boldsymbol{\mu})\end{pmatrix} \tag{5.32}$$

and in the right-hand side, we can see that we need to compute the value of the magnetic field at the points of interpolation $\mathbf{t}_i$. This requires to compute an integral on Ω_C, which is of high dimension. So, even if we have reduced the cost of computing the integral on the whole domain Ω_{mgn} to compute it only on the few interpolation points, we lose the real-time computation of the magnetic field.
The idea is to use EQM, presented in section 3.2.4, to compute $\mathbf{B}(\mathbf{t}_i, \boldsymbol{\mu})$ with a reduced set of quadrature points. This way, we may have a real-time computation of the magnetic field when using geometrical parameters.

5.3 Conclusion

In this chapter, we presented several ways of computing the magnetic field, either by the Maxwell equations or by using the Biot-Savart law.
We showed two formulations, saddle-point and regularized for the Maxwell equations and detailed the corresponding discretization, each giving the expected convergence rate.

We also described the parallel algorithm used to compute the integral necessary for the Biot-Savart law, as well as the reduced algorithm to compute it in real time. When using geometrical parameters, the order reduction is more complex, needing the combination of DEIM and EQM to reach real-time computation. We showed the convergence of both methods with respect to the number of reduced basis.

Chapter 6

Linear elasticity model

At full power, the typical constraints within High Field Magnets reach in normal operation 80% of the material yield strength. The elasticity model is, thus, very important to ensure that these constraints remain sustainable.

High Field Magnets are, of course, subject to the constraints originating from the Lorentz forces $\mathbf{f} = \mathbf{j} \times \mathbf{B}$. We also have to consider the thermal dilatation since the temperature in the magnet may reach more than $100°C$ in operation. The pressure induced by the water flow is neglected.

We will first describe the linear elasticity model used, based on [Slaughter and Petrolito, 2002] where more details can be found. We will then present the Continuous Galerkin formulation with its validation, and finally, the Hybrid Discontinuous Galerkin formulation.

Contents

As Ω_T is at equilibrium state, the equation of motions becomes the following equilibrium equation :

$$\nabla \cdot \bar{\bar{\sigma}} + \mathbf{f} = 0 \tag{6.1}$$

where :

- $\bar{\bar{\sigma}}$ is the stress tensor
- $\mathbf{f}$ represents the volume forces applied on Ω

The quantity we focus on is the displacement vector $\mathbf{u}$, which is the unknown in which we write the system (6.1).
We have to introduce the tensor of small deformations $\bar{\bar{\varepsilon}}$:

$$\bar{\bar{\varepsilon}} = \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^T) \tag{6.2}$$

and the Hooke's law allows to link stress tensor $\bar{\bar{\sigma}}$ with tensor of small deformation $\bar{\bar{\varepsilon}}$:

$$\bar{\bar{\sigma}}(\bar{\bar{\varepsilon}}) = \frac{E}{1+\nu}\left(\bar{\bar{\varepsilon}} + \frac{\nu}{1-2\nu} Tr(\bar{\bar{\varepsilon}})I\right) \tag{6.3}$$

where :

- E is Young modulus
- ν is Poisson's ratio
- I is the identity tensor

We introduce the λ and μ the Lamé coefficients :

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \qquad \text{and} \qquad \mu = \frac{E}{2(1+\nu)}$$

which allows us to rewrite (6.3) as:

$$\bar{\bar{\sigma}}(\bar{\bar{\varepsilon}}) = 2\mu\bar{\bar{\varepsilon}} + \lambda Tr(\bar{\bar{\varepsilon}})I$$

To take into account the thermal dilatation of the magnet, we need to add a term to the stress tensor:

$$\bar{\bar{\sigma}}(\bar{\bar{\varepsilon}}) = \bar{\bar{\sigma}}_E(\bar{\bar{\varepsilon}}) + \bar{\bar{\sigma}}_T = 2\mu\bar{\bar{\varepsilon}} + \lambda Tr(\bar{\bar{\varepsilon}})I - \frac{E}{1-2\nu}\alpha_T(T - T_0)I \tag{6.4}$$

Remark 5. Here, we chose to use the temperature on the same domain as the displacement because we made the hypothesis that we have only small deformations not impacting the computation of the temperature and the other fields of interests. But, to be more precise, we should use an ALE map to have the temperature in the reference domain.

For simplicity, we will omit the dependence of $\bar{\bar{\sigma}}$ on $\bar{\bar{\varepsilon}}$ in the following.

We need to complete the system with some boundary conditions:

- where the magnet is attached, we can prescribe the displacement with Dirichlet conditions:

$$\mathbf{u} = \mathbf{u}_D \quad \text{on } \partial\Omega_D \tag{6.5a}$$

- pressure forces can be imposed using Neumann conditions:

$$\bar{\bar{\sigma}} \cdot \mathbf{n} = \mathbf{g} \quad \text{on } \partial\Omega_P \tag{6.5b}$$

where $\partial\Omega = \partial\Omega_D \cup \partial\Omega_P$.

To determine if the deformation of the magnet stays elastic or becomes plastic engineers uses yield surfaces. This allows to have a simple criterion which should not exceed a certain value to know if the magnet is damaged or not. The most common for isotropic materials are the Tresca and Von Mises criteria.

The Tresca yield criterion, also known as the maximum shear stress theory criterion [Tresca, 1864] uses the diagonalized stress tensor $\bar{\bar{\sigma}}^d$ and is determined as:

$$tr_{\bar{\bar{\sigma}}^d} = \max(|\bar{\bar{\sigma}}^d_{11} - \bar{\bar{\sigma}}^d_{22}|, |\bar{\bar{\sigma}}^d_{22} - \bar{\bar{\sigma}}^d_{33}|, |\bar{\bar{\sigma}}^d_{33} - \bar{\bar{\sigma}}^d_{11}|)$$

The Von Mises criterion, also known as the maximum distortion energy criterion [Mises, 1913] can be expressed using the stress tensor $\bar{\bar{\sigma}}$ or the diagonalized stress tensor $\bar{\bar{\sigma}}^d$:

$$vm_{\bar{\bar{\sigma}}} = \sqrt{\frac{1}{2}\left((\bar{\bar{\sigma}}_{11} - \bar{\bar{\sigma}}_{22})^2 + (\bar{\bar{\sigma}}_{22} - \bar{\bar{\sigma}}_{33})^2 + (\bar{\bar{\sigma}}_{33} - \bar{\bar{\sigma}}_{11})^2\right) + 3\left(\bar{\bar{\sigma}}^2_{12} + \bar{\bar{\sigma}}^2_{23} + \bar{\bar{\sigma}}^2_{31}\right)}$$

$$vm_{\bar{\bar{\sigma}}^d} = \sqrt{\frac{1}{2}\left((\bar{\bar{\sigma}}_{11} - \bar{\bar{\sigma}}_{22})^2 + (\bar{\bar{\sigma}}_{22} - \bar{\bar{\sigma}}_{33})^2 + (\bar{\bar{\sigma}}_{33} - \bar{\bar{\sigma}}_{11})^2\right)}$$

6.1 CG formulation

The variational formulation consists in finding $\mathbf{u}$ in X such that:

$$\int_\Omega \bar{\bar{\sigma}} : \nabla\boldsymbol{\varphi} - \int_{\partial\Omega} \bar{\bar{\sigma}} \cdot \mathbf{n} \cdot \boldsymbol{\varphi} = -\int_\Omega \mathbf{f} \cdot \boldsymbol{\varphi} \quad \forall \boldsymbol{\varphi} \in Y \tag{6.6}$$

We now need to make appear the displacement vector $\mathbf{u}$ thanks to the Hooke's law (6.3) in the first component of (6.6):

$$\int_\Omega \bar{\bar{\sigma}} : \nabla\boldsymbol{\varphi} = 2\mu \int_\Omega \bar{\bar{\varepsilon}} : \nabla\boldsymbol{\varphi} + \lambda \int_\Omega Tr(\bar{\bar{\varepsilon}}) I : \nabla\boldsymbol{\varphi} \tag{6.7}$$

We have the following identities:

$$Tr(\bar{\bar{\varepsilon}}) = \nabla \cdot \mathbf{u}, \quad I : \nabla\boldsymbol{\varphi} = \nabla \cdot \boldsymbol{\varphi}, \quad \bar{\bar{\varepsilon}} : \nabla\boldsymbol{\varphi} = \bar{\bar{\varepsilon}} : \bar{\bar{s}} \quad \text{with} \quad \bar{\bar{s}} = \frac{1}{2}(\nabla\boldsymbol{\varphi} + \nabla\boldsymbol{\varphi}^T) \tag{6.8}$$

Which give us the formulation:

$$2\mu \int_\Omega \bar{\bar{\varepsilon}} : \bar{\bar{s}} + \lambda \int_\Omega (\nabla \cdot \mathbf{u})(\nabla \cdot \boldsymbol{\varphi}) - \int_{\partial\Omega} \bar{\bar{\sigma}} \cdot \mathbf{n} \cdot \boldsymbol{\varphi} = -\int_\Omega \mathbf{f} \cdot \boldsymbol{\varphi} \quad \forall \boldsymbol{\varphi} \in X \tag{6.9}$$

Imposing the boundary conditions (6.5a) in a strong form leads to finding $\mathbf{u} \in H^1_{\mathbf{u}_D,\Omega_D}(\Omega)$ such that:

$$2\mu \int_\Omega \bar{\bar{\varepsilon}} : \bar{\bar{s}} + \lambda \int_\Omega (\nabla \cdot \mathbf{u})(\nabla \cdot \boldsymbol{\varphi}) = -\int_\Omega \mathbf{f} \cdot \boldsymbol{\varphi} + \int_{\partial\Omega_P} \mathbf{g} \cdot \boldsymbol{\varphi} \quad \forall \boldsymbol{\varphi} \in H^1_{0,\Omega_D}(\Omega) \tag{6.10}$$

We also can impose the boundary condition (6.5a) in a weak way by adding penalty and consistency terms, giving us the following formulation: Find $\mathbf{u} \in H^1(\Omega)$ such that:

$$\begin{aligned} 2\mu \int_\Omega \bar{\bar{\varepsilon}} : \bar{\bar{s}} + \lambda \int_\Omega (\nabla \cdot \mathbf{u})(\nabla \cdot \boldsymbol{\varphi}) + \int_{\partial\Omega_D} \frac{\gamma}{h_s} \mathbf{u} \cdot \boldsymbol{\varphi} \\ - \int_{\partial\Omega_D} (\bar{\bar{\sigma}}(\bar{\bar{\varepsilon}}) \cdot \mathbf{n}) \cdot \boldsymbol{\varphi} - \int_{\partial\Omega_D} (\bar{\bar{\sigma}}(\bar{\bar{s}}) \cdot \mathbf{n}) \cdot \mathbf{u} = - \int_\Omega \mathbf{f} \cdot \boldsymbol{\varphi} + \int_{\partial\Omega_P} \mathbf{g} \cdot \boldsymbol{\varphi} \qquad \forall \boldsymbol{\varphi} \in H^1(\Omega) \\ + \int_{\partial\Omega_D} \frac{\gamma}{h_s} \mathbf{u}_D \cdot \boldsymbol{\varphi} - \int_{\partial\Omega_D} (\bar{\bar{\sigma}}(\bar{\bar{s}}) \cdot \mathbf{n}) \cdot \mathbf{u}_D \end{aligned} \tag{6.11}$$

In the case where we take into account the thermal dilatation and where we impose the Dirichlet conditions in a strong way, we need to add the following term to the right-hand side:

$$\int_\Omega \frac{E\alpha_T}{1-2\nu}(T - T_0)\nabla \cdot \boldsymbol{\varphi} \tag{6.12}$$

For the weak form of the Dirichlet conditions, add the previous term and the following to the right-hand side:

$$- \int_{\partial\Omega_D} \bar{\bar{\sigma}}_T \cdot \mathbf{n} \cdot \mathbf{u}_D \tag{6.13}$$

and the two following terms to the matrix:

$$- \int_{\partial\Omega_D} \bar{\bar{\sigma}}_T \cdot \mathbf{n} \cdot \boldsymbol{\varphi} - \int_{\partial\Omega_D} \bar{\bar{\sigma}}_T \cdot \mathbf{n} \cdot \mathbf{u} \tag{6.14}$$

6.1.1 Verification

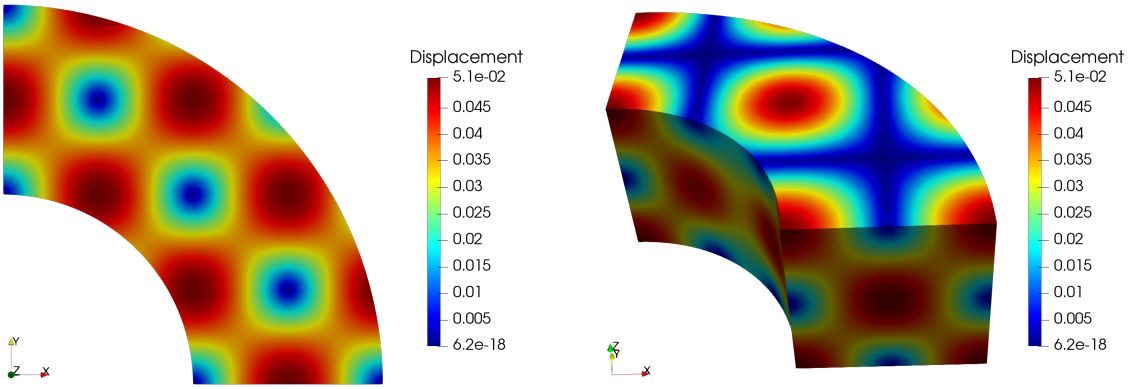
To test our implementation of the CG formulation for the linear elasticity, we will check the convergence orders on a test problem. The geometry is the same as in section 4.1.2.1. In the following $\lambda = \mu = 1$, and the exact solutions are:

- in 2D:

$$\begin{aligned} \mathbf{u} &= \frac{1}{2\pi^2} (\sin(\pi x)\cos(\pi y), \cos(\pi x)\sin(\pi y)) \\ \bar{\bar{\sigma}} &= \frac{1}{\pi} \begin{pmatrix} 2\cos(\pi x)\cos(\pi y) & -\sin(\pi x)\sin(\pi y) \\ -\sin(\pi x)\sin(\pi y) & 2\cos(\pi x)\cos(\pi y) \end{pmatrix} \end{aligned}$$

- in 3D:

$$\begin{aligned} \mathbf{u} &= \frac{1}{2\pi^2} (\sin(\pi x)\cos(\pi y)\sin(\pi z), \cos(\pi x)\sin(\pi y)\sin(\pi z), \cos(\pi x)\cos(\pi y)\cos(\pi z)) \\ \bar{\bar{\sigma}} &= \frac{1}{\pi} \begin{pmatrix} \frac{3}{2}\cos(\pi x)\cos(\pi y)\sin(\pi z) & -\sin(\pi x)\sin(\pi y)\sin(\pi z) & 0 \\ -\sin(\pi x)\sin(\pi y)\sin(\pi z) & \frac{3}{2}\cos(\pi x)\cos(\pi y)\sin(\pi z) & 0 \\ 0 & 0 & -\frac{1}{2}\cos(\pi x)\cos(\pi y)\sin(\pi z) \end{pmatrix} \end{aligned}$$



(a) 2D (b) 3D

Figure 6.1 – Exact solution for the displacement in 2D and 3D

The displacement is presented in figure 6.1.

We use Dirichlet boundary conditions on V_0 and V_1 and Neumann boundary conditions and all the other surfaces.

The errors on the displacement and the stress are presented in figures 6.2 and 6.3 respectively.

We can see that the orders of convergence are in agreement with the theoretical predictions.

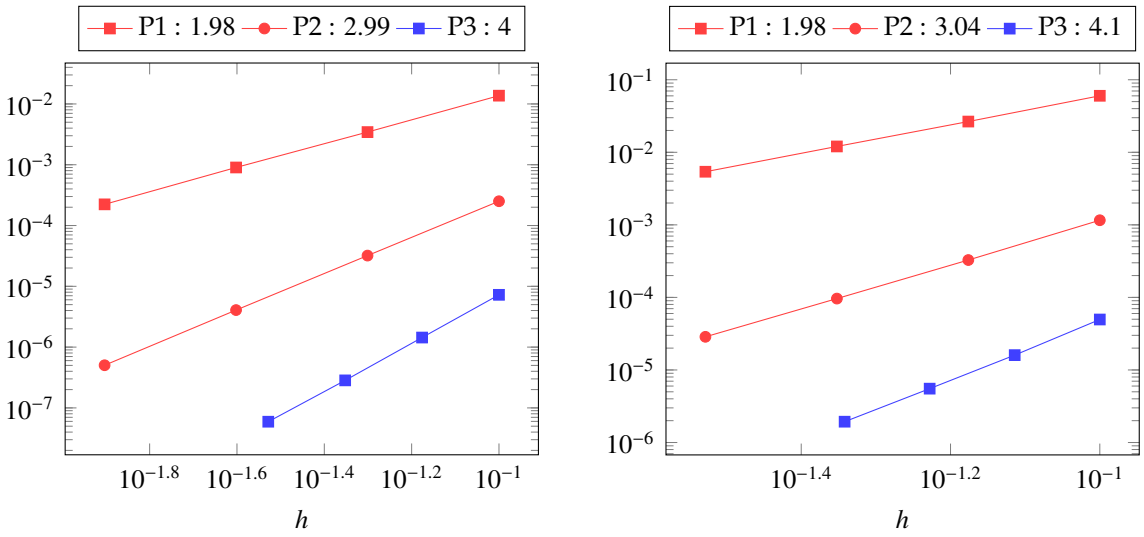
6.1.2 Validation

In this example, we consider a solenoid conductor with finite thickness and infinite length. This allows us to ignore the z components in our equations. We admit that there is only a radial expansion.

6.1.2.1 Problem

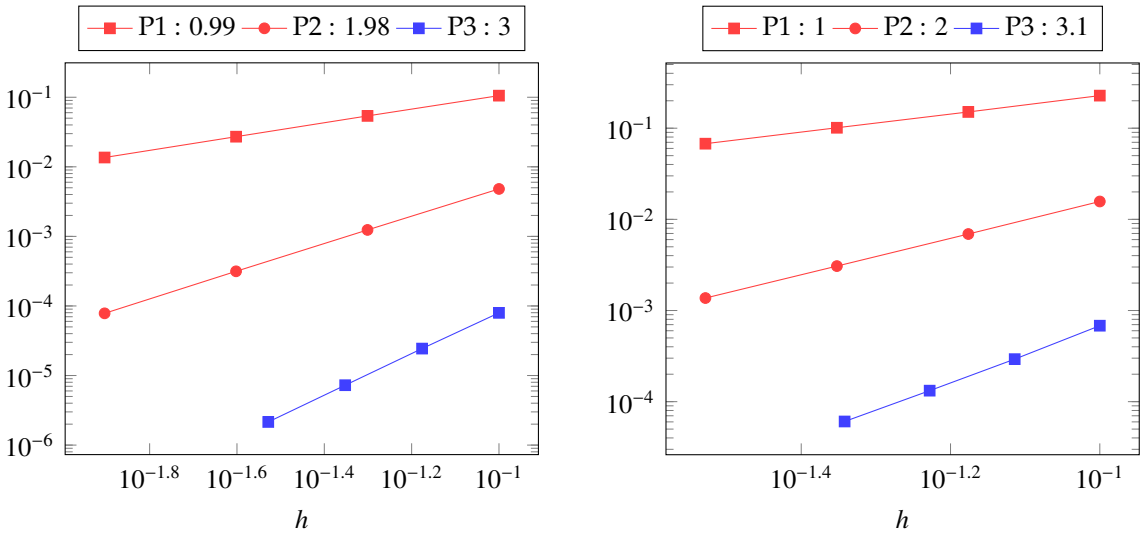
Since we cannot build an infinite geometry, we use a solenoid of length 20 with an internal radius of 1 and an external radius of 2. The volumic force $\mathbf{f}$ take into account the Lorenz forces but not the thermal dilation, hence $\mathbf{f} = \mathbf{j} \times \mathbf{B}$. The current density $\mathbf{j}$ and the magnetic field $\mathbf{B}$ can be seen as input parameters or given by our thermoelectric and magnetostatic models.

The material properties of the solenoid are presented in the table 6.1.



(a) 2D (b) 3D

Figure 6.2 – Displacement convergence for L2 norm



(a) 2D (b) 3D

Figure 6.3 – Stress convergence for L2 norm

Name	Description	Value	Unit
E	Young modulus	128.10^9	$Pa = kg.m^{-1}.s^{-2}$
ν	Poisson's ratio	0.33	-

Table 6.1 – Materials

We impose Neumann conditions on all boundaries of the solenoid, except for the top and bottom, where we will use Dirichlet boundary conditions set to 0 to model the clamped side of the magnet.

The analytical solution of an infinite solenoid differs from this case because of the geometry used. We will then only compare the radial displacement at $z = 10$, to be as far as possible from the top and bottom of the solenoid.

An analytical solution can be found in [Wilson, 1987] or [Montgomery, 1969], and interested readers can find more details in them. The idea to find the analytical solution is to express our quantities in cylindrical coordinates and use the fact that the only non-null component of the current density is its angular component j_θ. We can find the radial component u_r of $\mathbf{u}$ as solution of:

$$\frac{\partial}{\partial r}\left(r\frac{\partial u_r}{\partial r}\right) - \frac{u_r}{r} = -\frac{(1+\nu)(1-2\nu)}{E(1-\nu)} r j_\theta b_z$$

We want to express the solution as

$$u_r = C_1 r + \frac{C_2}{r} + u_p(r)$$

where u_p is a particular solution. Using boundary conditions we can find the two coefficients C_1 and C_2:

$$C_1 = \frac{(1+\nu)(1-2\nu)}{E(1-\nu)} j_\theta r_i \left[\left(b_z(r_i) + \frac{\Delta b_z}{\alpha - 1}\right)\left(\frac{2-\nu}{3}\right)\left(1 - \frac{r_e^2}{(r_e^2 - r_i^2)}\left(1 - \frac{r_e}{r_i}\right)\right)\right.$$
$$\left. + \frac{\Delta b_z}{\alpha - 1}\left(\frac{2\nu - 3}{8}\right)\left(1 - \frac{r_e^2}{(r_e^2 - r_i^2)}\left(1 - \left(\frac{r_e}{r_i}\right)^2\right)\right)\right]$$

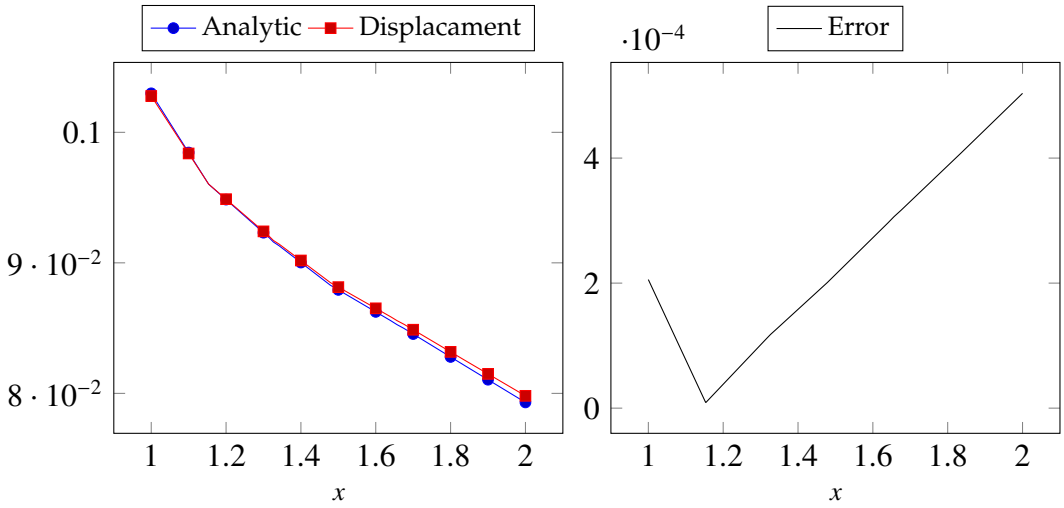
$$C_2 = \frac{-r_i^3 r_e^2 j_\theta}{(r_e^2 - r_i^2)} \frac{(1+\nu)}{E(1-\nu)} \left[\left(b_z(r_i) + \frac{\Delta b_z}{\alpha - 1}\right)\left(\frac{2-\nu}{3}\right)\left(1 - \frac{r_e}{r_i}\right) + \frac{\Delta b_z}{\alpha - 1}\left(\frac{2\nu - 3}{8}\right)\left(1 - \left(\frac{r_e}{r_i}\right)^2\right)\right]$$

where $\alpha = \frac{r_e}{r_i}$ and $\Delta b_z = b_z(r_i) - b_z(r_e)$.
Using the expressions for $\mathbf{j}$ and $\mathbf{B}$, we can find the particular solution u_p. In case of a Bitter coil, as in section 4.1.2.1, the current distribution is in r_i/r and the magnetic field is linear with respect to r, we find:

$$u_p(r) = \frac{(1+\nu)(1-2\nu)}{E(1-\nu)} r_i j_\theta \left[\frac{\Delta b_z}{\ln(\alpha)} r \ln\left(\frac{r}{r_1}\right)^2 - \left(2b_z(r_1) + \frac{\Delta b_z}{\ln(\alpha)}\right) r \ln\left(\frac{r}{r_1}\right)\right]$$

whereas if the current density is uniform in the coil, we have:

$$u_p(r) = \frac{(1+\nu)(1-2\nu)}{E(1-\nu)} r_i j_\theta \left[-\frac{r_i}{3}\left(b_z(r_i) + \frac{\Delta b_z}{\alpha - 1}\right)\left(\frac{r}{r_i}\right)^2 + \frac{r_i}{8}\frac{\Delta b_z}{\alpha - 1}\left(\frac{r}{r_i}\right)^3\right]$$



(a) X component of the displacement

(b) Error on the X component of the displacement

Figure 6.4 – Analytical and computed X component of the displacement

Finally, returning to Cartesian coordinates, the displacement $\mathbf{u}$ can be written as:

$$\mathbf{u} = \begin{pmatrix} \cos(\theta)u_r(r) \\ \sin(\theta)u_r(r) \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{x}{\sqrt{x^2+y^2}} u_r(\sqrt{x^2+y^2}) \\ \frac{y}{\sqrt{x^2+y^2}} u_r(\sqrt{x^2+y^2}) \\ 0 \end{pmatrix}$$

6.1.2.2 Results

In figure 6.4 are presented the X component of the displacement on the line $[0,1]\times\{0\}\times\{10\}$ as well as the error between the analytical solution and computed displacement.

We can see that our model is close to the analytical solution, and the error, even if it increases when the radius increases, does not exceed $1e^{-4}$. We conclude that our model is valid for this problem.

6.2 HDG formulation

The HDG formulation for linear and non-linear elasticity have been described in [Kabaria et al., 2014, Qiu et al., 2013].
By rewriting the equation (6.1) as a system of first order equations, we can write the linear elasticity problem as:

$$\mathcal{A}\bar{\bar{\sigma}} - \bar{\bar{\varepsilon}} = 0$$
$$\nabla \cdot \bar{\bar{\sigma}} + \mathbf{f} = 0$$

with $\mathcal{A}$ the compliance operator, which is a bounded, symmetric, positive definite operator over the set of symmetric matrices $\bar{\bar{S}}(K) \subset \mathbb{R}^{d\times d}$:

$$\mathcal{A}\bar{\bar{v}} = \frac{1}{2\mu}\bar{\bar{v}} + \frac{-\lambda}{2\mu(\lambda+2\mu)} Tr(\bar{\bar{v}})I$$

Using the same notations introduced in chapter 2, the HDG formulation of the problem is:

$$\begin{aligned}
(\mathcal{A}\bar{\bar{\sigma}}_h, \bar{\bar{v}}_h)_\Omega + (\mathbf{u}_h, \nabla\cdot\bar{\bar{v}}_h)_\Omega - \langle\hat{u}_h, \bar{\bar{v}}_h\cdot\mathbf{n}\rangle_\Gamma &= 0\\
(\nabla\cdot\bar{\bar{\sigma}}_h, \mathbf{w}_h)_\Omega - \langle\tau\mathbf{u}_h, \mathbf{w}\rangle_\Gamma + \langle\tau\hat{\mathbf{u}}_h, \mathbf{w}\rangle_\Gamma &= (\mathbf{f}, \mathbf{w})_\Omega\\
\langle\bar{\bar{\sigma}}_h\cdot\mathbf{n}, \boldsymbol{\mu}_h)\rangle_{\mathcal{E}_h^0} - \langle\tau\mathbf{u}_h, \boldsymbol{\mu}_h\rangle_{\mathcal{E}_h^0} + \langle\tau\hat{\mathbf{u}}_h, \boldsymbol{\mu}_h\rangle_{\mathcal{E}_h^0} &= 0\\
\langle\bar{\bar{\sigma}}_h\cdot\mathbf{n}, \boldsymbol{\mu}_h)\rangle_{\Gamma_N} - \langle\tau\mathbf{u}_h, \boldsymbol{\mu}_h\rangle_{\Gamma_N} + \langle\tau\hat{\mathbf{u}}_h, \boldsymbol{\mu}_h\rangle_{\Gamma_N} &= \langle\mathbf{g}, \boldsymbol{\mu}_h\rangle_{\Gamma_N}\\
\langle\hat{\mathbf{u}}_h, \boldsymbol{\mu}_h\rangle_{\Gamma_D} &= \langle\mathbf{u}_D, \boldsymbol{\mu}_h\rangle_{\Gamma_D}
\end{aligned}$$

for all $(\bar{\bar{v}}_h, \mathbf{w}_h, \boldsymbol{\mu}) \in \bar{\bar{V}}_h \times \mathbf{W}_h \times \mathbf{M}_h$ where

$$\begin{aligned}
\bar{\bar{V}}_h &= \{\bar{\bar{v}} \in \mathbb{R}^{d\times d}(\Omega) \,|\, \bar{\bar{v}}_{|K} \in [P_k(K)]^{d\times d} \subset \bar{\bar{S}}(K) \quad \forall K \in \Omega_h\}\\
\mathbf{W}_h &= \{\mathbf{w} \in \mathbb{R}^d(\Omega) \,|\, \mathbf{w}_{|K} \in [P_k(K)]^d \quad \forall K \in \Omega_h\}\\
\mathbf{M}_h &= \{\boldsymbol{\mu} \in \mathbb{R}^d(\Omega) \,|\, \boldsymbol{\mu}_{|K} \in [P_k(F)]^d \quad \forall F \in \mathcal{E}_h\}
\end{aligned}$$

Remark 6. The symmetry of the elements of $\bar{\bar{V}}_h$ is enforced in an essential manner in their definition [Qiu et al., 2013].

6.2.1 Verification

For the verification of the HDG formulation, we use exactly the same problem as in section 6.1.1. The relative errors in norm L_2 for the displacement and the stress are presented in figures 6.5 and 6.6 respectively.

We can see that the orders of convergence are in agreement with the theoretical predictions, see [Qiu et al., 2013].

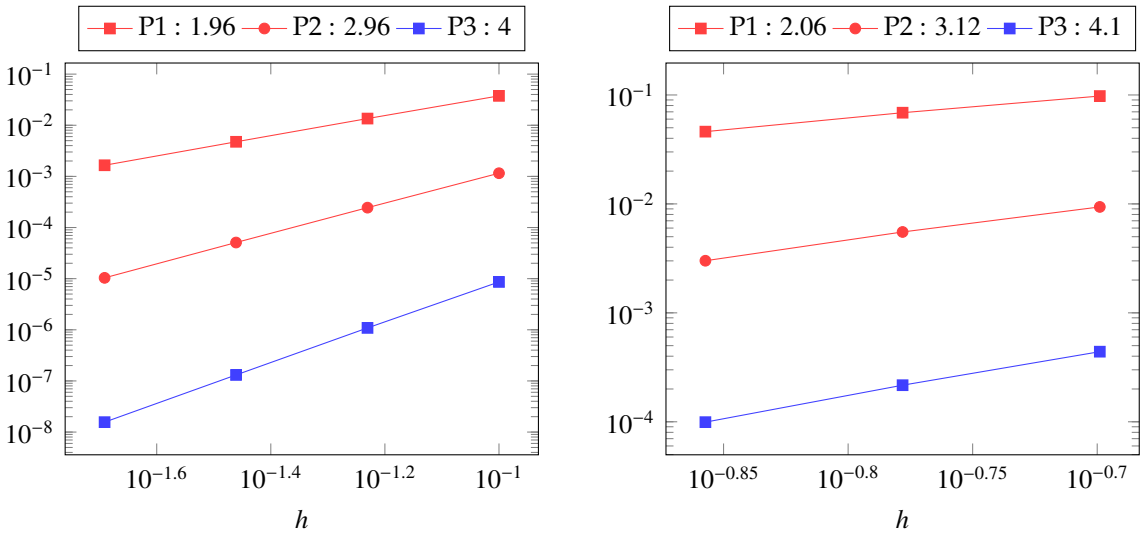
6.2.2 Validation

To validate our model, we use the same configuration as in section 6.1.2. We also compare the X component of the analytical solution on the line $[0, 1] \times \{0\} \times \{10\}$ with the computed solution. The values are presented in figure 6.7a and the error in figure 6.7b.

We can see that although the error is greater than in CG, it is still acceptable and thus validate our HDG model.

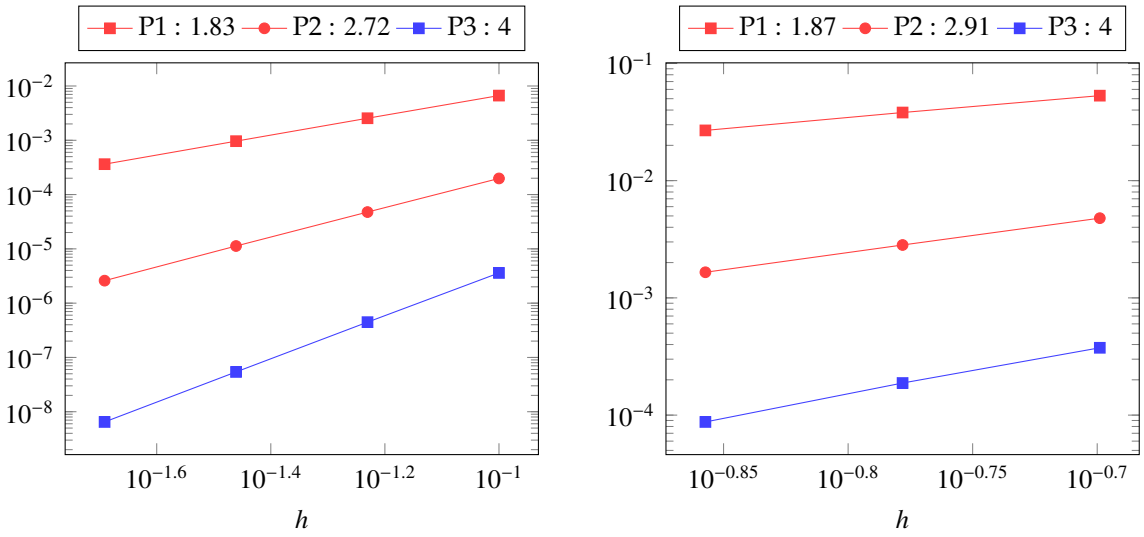
6.3 Conclusion

In this chapter, we have presented the linear elasticity problem and its CG and HDG formulations. We have shown that our models converge at the appropriate orders.



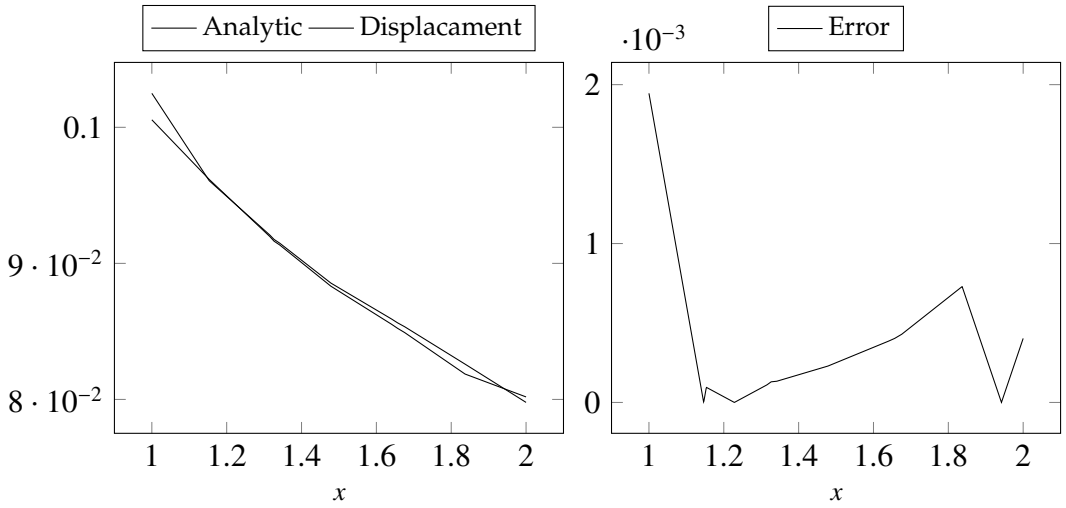
(a) 2D (b) 3D

Figure 6.5 – Displacement convergence for L2 norm



(a) 2D (b) 3D

Figure 6.6 – Stress convergence for L2 norm



(a) X component of the displacement

(b) Error on the X component of the displacement

Figure 6.7 – Analytical and computed X component of the displacement

And we have validated our models on a physical case. Although the HDG method is more demanding in terms of memory, we can be close to the CG results if we increase the polynomial order.

This third problem, with the thermoelectric and magnetostatic problem, allows us to have a complete model for our magnet. We can now compute all relevant quantities for the study of the high field magnets: the temperature, the current density, the magnetic field and the deformation of the magnet.

Part III

Implementations

In this part we describe the implementation of the numerical methods introduced previously using the Finite Element Library in C++ `Feel++` [Prud'Homme et al., 2012]. This library is used by a wide range of users, from mechanical engineers in industry, physicists in complex fluids, computer scientists in biomedical applications to applied mathematicians. This is made possible by the use of the domain specific embedded language (DSEL) as close as possible to the shared mathematical formulation hiding linear algebra and computer science complexities. The DSEL allows to use simply Galerkin methods (FEM, SEM, CG, DG, CRB) in 1D, 2D and 3D for arbitrary polynomial orders on simplices and hypercubes meshes.

This is feasible because `Feel++` takes advantage of the newest C++ standard such as type inference, uses meta-programmation to deal with templates and rely on different well-known third party libraries. The most prominent amid them are OpenMPI for the parallelism, PETSc, SLEPc and Eigen for the linear algebra, Boost for different useful libraries, in particular for the meta-programmation ,and HDF5 to store vectors and meshes in a parallel fashion.

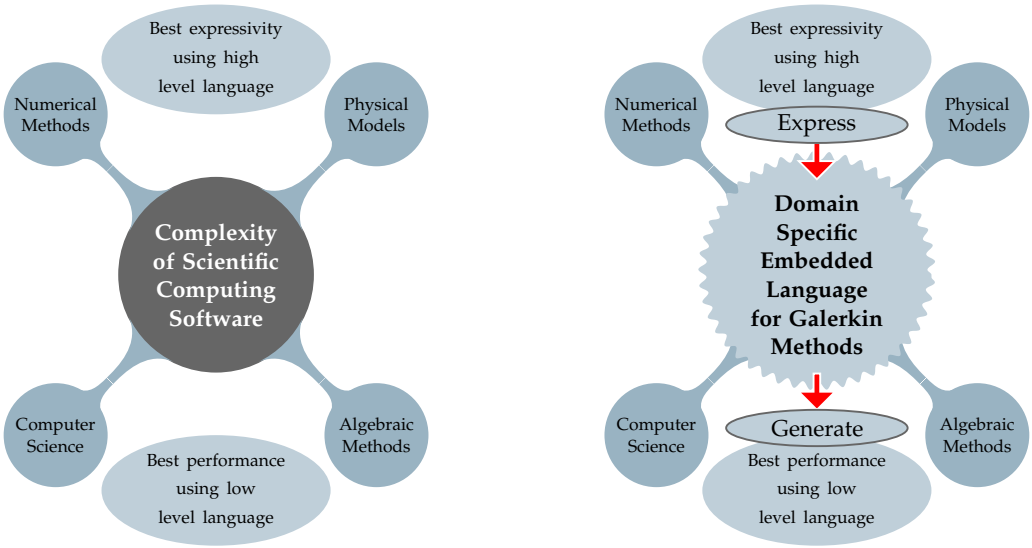


Figure 6.8 – The DSEL offered by `Feel++`provides high level language to break the complexity of scientific computing software while keeping the performances of a low level language.

`Feel++` also provides applications, called *toolboxes*, to solve a variety of problems using only light configuration files. We started this thesis by reimplementing the existing classes to use the latest developement of `Feel++` and use the *toolboxes*, easing the configuration and coupling of the different problems.
The next chapter will describe the classes implemented during this thesis to provide such a toolbox for the HDG mixed Poisson and mixed Elasticity problems. It has been realized with the help of Lorenzo Sala, a PhD student at the University of Strasbourg.
The other chapter of this part will focus on the different classes needed for the reduction of the Biot & Savart problem.

Chapter 7

HDG Method

In this chapter, we discuss the computational framework implementing the HDG methods described in chapters 4 and 6. We will first describe the C++ class `MixedPoisson` in section 7.1 implementing a mixed Poisson problem and show how to use this class to solve the problem. Next we do the same for the `MixedElasticity` class and the linear elasticity problem.

Contents

7.1 `MixedPoisson`

To provide a toolbox for the mixed Poisson problem, we create a C++ class called `MixedPoisson`, child of the superclass `ModelNumerical`. This superclass loads all the information needed for our model, such as the materials used, the boundary conditions and their relation with the mesh by the way of markers.

In Code 7.1, we start by introducing some type definitions needed for the meshes and the function spaces, in particular:

`Dim` is the domain dimension

`Order` is the polynomial order

`G_Order` is the polynomial order of the geometric transformation

`Pdhv_type` represents a vectorial discontinuous Galerkin space

`Pdh_type` represents a scalar discontinuous Galerkin space

`Pch_type` represents a scalar continuous Galerkin space

Code 7.1 – Type definition for HDG Mixed Poisson

```
1    using convex_type = Simplex<Dim,G_Order>;
```

```
2     using mesh_type = Mesh<convex_type>;
3     using face_convex_type = Simplex<Dim-1,G_Order,Dim>;
4     using face_mesh_type = Mesh<face_convex_type>;
5     using Vh_t = Pdhv_type<mesh_type,Order>;
6     using Wh_t = Pdh_type<mesh_type,Order>;
7     using Whp_t = Pdh_type<mesh_type,Order+1>;
8     using Mh_t = Pdh_type<face_mesh_type,Order>;
9     using Ch_t = Pch_type<face_mesh_type,0>;
10    using M0h_t = Pdh_type<face_mesh_type,0>;
11    using product2_space_type = ProductSpaces2<Ch_ptr_t,Vh_ptr_t,Wh_ptr_t,Mh_ptr_t>;
```

Once the types are declared, we can create the mesh on the domain, on the faces of the Integral boundary conditions for λ and on the complement of it in the faces of the domain for the trace $\hat{p}$, as well as the product space $\mathbf{V}_h^k \times W_h^k \times M_h^k \times C_h$.

Code 7.2 – Creation of the meshes and spaces for HDG Mixed Poisson

```
1     M_mesh = loadMesh( new mesh_type);
2     // Mh only on the faces whitout integral condition
3     auto complement_integral_bdy = complement(faces(M_mesh),[this]( auto const& ewrap ) {
4       auto const& e = unwrap_ref( ewrap );
5       for( auto exAtMarker : this->M_IBCList)
6       {
7         if ( e.hasMarker() && e.marker().value() == this->M_mesh->markerName(
              exAtMarker.marker() ) )
8         return true;
9       }
10      return false; });
11
12
13  Then we can build the matrix and the vector for the bilinear and linear forms.    auto
      face_mesh = createSubmesh( _mesh=M_mesh, _range=complement_integral_bdy, _update=0 );
14
15    M_Vh = Pdhv<Order>( M_mesh, true);
16    M_Wh = Pdh<Order>( M_mesh, true );
17    M_Whp = Pdh<Order+1>( M_mesh, true );
18    M_Mh = Pdh<Order>( face_mesh, true );
19    M_M0h = Pdh<0>( face_mesh );
20
21    auto ibc_mesh = createSubmesh( _mesh=M_mesh, _range=markedfaces(M_mesh, ibc_markers),
          _update=0 );
22    M_Ch = Pch<0>( ibc_mesh, true );
23
24    auto ibcSpaces = std::make_shared<ProductSpace<Ch_ptr_t,true> >( M_integralCondition,
          M_Ch);
25    M_ps = std::make_shared<product2_space_type>(product2(ibcSpaces,M_Vh,M_Wh,M_Mh));
```

Then we can build the matrix and the vector for the bilinear and linear forms. At this point, depending on the strategy to solve the problem, monolithic or static condensation, we will determine which pattern to use for the matrix. Likewise, we build the matrix and the vector for the post-process, using a local pattern the problem can be solved in each cell of the mesh. We also create the block bilinear and linear forms associated with the matrix and vector.

Code 7.3 – Matrices and variational forms for HDG Mixed Poisson

```
1     solve::strategy s = M_useSC ? solve::strategy::static_condensation :
          solve::strategy::monolithic;
2     solve::strategy spp = solve::strategy::local;
3     auto pps = product( M_Whp );
4
5     M_A_cst = makeSharedMatrixCondensed<value_type>(s, csrGraphBlocks(*M_ps,
          (s>=solve::strategy::static_condensation)?Pattern::ZERO:Pattern::COUPLED), *M_backend
          );
```

```
6       M_F = makeSharedVectorCondensed<value_type>(s, blockVector(*M_ps), *M_backend, false);
7       M_App = makeSharedMatrixCondensed<value_type>(spp,  csrGraphBlocks(pps,
            (spp>=solve::strategy::static_condensation)?Pattern::ZERO:Pattern::COUPLED),
            backend(), true );
8       M_Fpp = makeSharedVectorCondensed<value_type>(solve::strategy::local, blockVector(pps),
            backend(), false);
9
10      auto bbf = blockform2( *M_ps, M_A_cst);
11      auto blf = blockform1( *M_ps, M_F );
```

Next, we show the implementation of the assembly of the system. In particular:

- the core of the HDG matrix in Code 7.4
- the right-hand side in Code 7.5
- the Dirichlet boundary condition in Code 7.6
- the Neumann boundary condition in Code 7.7
- the Integral boundary condition in Code 7.8

We can access the block (i, j) of the bilinear form using integral constant known at compile time (`0\_c` for example).

Code 7.4 – Assemble core the matrix for HDG Mixed Poisson

```
1       for( auto const& pairMat : modelProperties().materials() )
2       {
3           auto marker = pairMat.first;
4           auto material = pairMat.second;
5           auto cond = material.getScalar(M_conductivityKey, M_paramValues);
6           // (sigma^-1 j, v)
7           bbf(0_c,0_c) += integrate(_range=markedelements(M_mesh,marker),
8                                     _expr=(trans(idt(u))*id(v))/cond );
9       }
10
11      // -(p,div(v))_Omega
12      bbf( 0_c, 1_c ) += integrate(_range=elements(M_mesh),_expr=-(idt(p)*div(v)));
13
14      // <phat,v.n>_Gamma\Gamma_I
15      bbf( 0_c, 2_c ) += integrate(_range=internalfaces(M_mesh),
16                                   _expr=(
                                         idt(phat)*(leftface(normal(v))+rightface(normal(v)))) );
17      bbf( 0_c, 2_c ) += integrate(_range=gammaMinusIntegral,
18                                   _expr=idt(phat)*normal(v));
19
20      // (div(j),q)_Omega
21      bbf( 1_c, 0_c ) += integrate(_range=elements(M_mesh), _expr=id(w)*divt(u));
22
23
24      // <tau p, w>_Gamma
25      bbf( 1_c, 1_c ) += integrate(_range=internalfaces(M_mesh),
26                                   _expr=tau_constant *
27                                   ( leftfacet( idt(p))*leftface(id(w)) +
28                                     rightfacet( idt(p))*rightface(id(w) )));
29      bbf( 1_c, 1_c ) += integrate(_range=boundaryfaces(M_mesh),
30                                   _expr=tau_constant * id(w)*idt(p));
31
32
33      // <-tau phat, w>_Gamma\Gamma_I
34      bbf( 1_c, 2_c ) += integrate(_range=internalfaces(M_mesh),
35                                   _expr=-tau_constant * idt(phat) *
36                                   ( leftface( id(w) )+
37                                     rightface( id(w) )));
38      bbf( 1_c, 2_c ) += integrate(_range=gammaMinusIntegral,
```

```
                                _expr=-tau_constant * idt(phat) * id(w) );


    // <j.n,mu>_Omega/Gamma
    bbf( 2_c, 0_c ) += integrate(_range=internalfaces(M_mesh),
                                _expr=( id(l)*(leftfacet(normalt(u))+
                                               rightfacet(normalt(u))) ) );

    // <tau p, mu>_Omega/Gamma
    bbf( 2_c, 1_c ) += integrate(_range=internalfaces(M_mesh),
                                _expr=tau_constant * id(l) * ( leftfacet( idt(p) )+
                                                               rightfacet( idt(p) )));

    // <-tau phat, mu>_Omega/Gamma
    bbf( 2_c, 2_c ) += integrate(_range=internalfaces(M_mesh),
                                _expr=-sc_param*tau_constant * idt(phat) * id(l) );
```

Code 7.5 – Assemble right hand side for HDG Mixed Poisson

```
    auto itField = modelProperties().boundaryConditions().find( "potential");
    if ( itField != modelProperties().boundaryConditions().end() )
    {
        auto mapField = (*itField).second;
        auto itType = mapField.find( "SourceTerm" );
        if ( itType != mapField.end() )
        {
            for ( auto const& exAtMarker : (*itType).second )
            {
                std::string marker = exAtMarker.marker();
                auto g = expr<expr_order>(exAtMarker.expression());
                if ( !this->isStationary() )
                    g.setParameterValues( { {"t", M_bdf_mixedpoisson->time()} } );
                g.setParameterValues( M_paramValues );
                blf(1_c) += integrate( _range=markedelements(M_mesh,marker),
                                       _expr=inner(g,id(w)) );
            }
        }
    }

    itField = modelProperties().boundaryConditions().find( "flux");
    if ( itField != modelProperties().boundaryConditions().end() )
    {
        auto mapField = (*itField).second;
        auto itType = mapField.find( "SourceTerm" );
        if ( itType != mapField.end() )
        {
            for ( auto const& exAtMarker : (*itType).second )
            {
                std::string marker = exAtMarker.marker();
                auto g = expr<Dim,1,expr_order>(exAtMarker.expression());
                if ( !this->isStationary() )
                    g.setParameterValues( { {"t", M_bdf_mixedpoisson->time()} } );
                blf(0_c) += integrate( _range=markedelements(M_mesh,marker),
                                       _expr=inner(g,id(v)) );
            }
        }
    }
```

Code 7.6 – Assemble Dirichlet boundary condition for HDG Mixed Poisson

```
    bbf( 2_c, 2_c ) += integrate(_range=markedfaces(M_mesh,marker),
                                _expr=idt(phat) * id(l) );
    blf(2_c) += integrate(_range=markedfaces(M_mesh,marker),
                          _expr=id(l)*expr);
```

Code 7.7 – Assemble Neumann boundary condition for HDG Mixed Poisson

```
    // <j.n,mu>_Gamma_N
    bbf( 2_c, 0_c ) += integrate(_range=markedfaces(M_mesh,marker),
                                 _expr=id(l)*normalt(u) );
    // <tau p, mu>_Gamma_N
    bbf( 2_c, 1_c ) += integrate(_range=markedfaces(M_mesh,marker),
                                 _expr=tau_constant * id(l) * idt(p) );
    // <-tau phat, mu>_Gamma_N
    bbf( 2_c, 2_c ) += integrate(_range=markedfaces(M_mesh,marker),
                                 _expr=-tau_constant * idt(phat) * id(l) );
    blf(2_c) += integrate( _range=markedfaces(M_mesh, marker),
                           _expr=id(l)*expr);
```

Code 7.8 – Assemble Integral boundary condition for HDG Mixed Poisson

```
    // <lambda, v.n>_Gamma_I
    bbf( 0_c, 3_c, 0, i ) += integrate( _range=markedfaces(M_mesh,marker),
                                        _expr= idt(uI) * normal(u) );

    // -<lambda, tau w>_Gamma_I
    bbf( 1_c, 3_c, 1, i ) += integrate( _range=markedfaces(M_mesh,marker),
                                        _expr=-tau_constant * idt(uI) * id(w) );

    // <j.n, m>_Gamma_I
    bbf( 3_c, 0_c, i, 0 ) += integrate( _range=markedfaces(M_mesh,marker), _expr=normalt(u) *
        id(nu) );


    // <tau p, m>_Gamma_I
    bbf( 3_c, 1_c, i, 1 ) += integrate( _range=markedfaces(M_mesh,marker),
                                        _expr=tau_constant *idt(p)  * id(nu) );

    // -<lambda2, m>_Gamma_I
    bbf( 3_c, 3_c, i, i ) += integrate( _range=markedfaces(M_mesh,marker),
                                        _expr=-tau_constant * id(nu) *idt(uI) );

    double meas = integrate(  _range=markedfaces(M_mesh,marker),
        _expr=cst(1.0)).evaluate()(0,0);
    // <I_target,m>_Gamma_I
    blf(3_c,i) += integrate( _range=markedfaces(M_mesh,marker), _expr=g*id(nu)/meas );
```

Finally, we solve the system either with the static condensation strategy or with the monolithic one. All the procedure described in section 2.2 is hidden in the solve method which will call the right method depending on if we have IBC or not. Then, we retrieve the flux and the potential in the first and second block of the solution respectively.

Code 7.9 – Solve the system for HDG Mixed Poisson

```
    auto U = M_ps->element();

    bbf.solve(_solution=U, _rhs=blf, _condense=M_useSC, _name=prefix());

    M_up = U(0_c);
    M_pp = U(1_c);
```

At the end, we can solve problem 2.14 to retrieve the post-processed potential.

Code 7.10 – Assemble and solve the post process for HDG Mixed Poisson

```
    auto pps = product( M_Whp );
    auto b = blockform2( pps, M_App);
    b( 0_c, 0_c ) = integrate( _range=elements(M_mesh),
                               _expr=inner(gradt(M_ppp),grad(M_ppp)));

    auto ell = blockform1( pps, M_Fpp);
```

```
    for( auto const& pairMat : modelProperties().materials() )
    {
        auto marker = pairMat.first;
        auto material = pairMat.second;
        auto cond = material.getScalar(M_conductivityKey, M_paramValues);
        ell(0_c) += integrate( _range=markedelements(M_mesh,marker),
                               _expr=-grad(M_ppp)*idv(M_up)/cond);

    b.solve( _solution=PP, _rhs=ell, _name="sc.post", _local=true);
    M_ppp=PP(0_c);
    auto P0dh = Pdh<0>(M_mesh);
    M_ppp -= M_ppp.ewiseMean(P0dh);
    M_ppp += M_pp.ewiseMean(P0dh);
```

We list here the main methods that are available in the API provided by `Feel++` in order to use this toolbox:

- `mesh()`, which allows to access the computational mesh,
- `potentialSpace()`,`fluxSpace()`, which allow to access the primal and dual spaces,
- `potentialField()`,`fluxField()`, which allow to access the primal and dual variables,
- `init(mesh=nullptr)`, which initialize the mesh, spaces, BCs, exporter,... The user can pass the mesh otherwise it is automatically loaded,
- `assembleAll()`, which will assemble all the different blocks of the bilinear and linear form. The user can have more control by calling subfunctions directly, such as `assembleCstPart()`,`assembleNonCstPart()`,`updateConductivityTerm(conducti` The last function can be useful if dealing with non-linear conductivity.
- `solve()`, which solves the system previously built,
- `postProcess()`, which will assemble and solve the post process system. The user can have more control by calling subfunctions directly, such as `assemblePostProcessCst` `assemblePostProcessRhs(conductivity, marker)`.
- `exportResults()`, which exports the results asked in the configuration file.

In Code 7.11, we show a simple example of how to use the toolbox.

Code 7.11 – Example of how to use the API for HDG Mixed Poisson

```
    using mp_type = FeelModels::MixedPoisson<nDim,OrderT,GOrder>;
    auto MP = mp_type::New("hdg.poisson");
    auto mesh = loadMesh( _mesh=new typename mp_type::mesh_type );
    MP->init(mesh);
    MP->assembleAll();
    MP->solve();
    MP->exportResults();
```

7.2 MixedElasticity

To provide a toolbox for the mixed elasticity problem, we create a C++ class called `MixedElasticity`, child of the superclass `ModelNumerical`. This superclass loads

all the information needed for our model, such as the materials used, the boundary conditions and their relation with the mesh by the way of markers.

In Code 7.12, we start by introducing some type definitions needed for the meshes and the function spaces, in particular:

`Dim` is the domain dimension

`Order` is the polynomial order

`G_Order` is the polynomial order of the geometric transformation

`Pdhms_type` represents a symmetric matrix discontinuous Galerkin space

`Pdhv_type` represents a vectorial discontinuous Galerkin space

`Pdh_type` represents a scalar discontinuous Galerkin space

`Pchv_type` represents a vectorial continuous Galerkin space

Code 7.12 – Type definition for HDG Mixed Elasticity

```
1  using convex_type = Simplex<Dim,G_Order>;
2  using mesh_type = Mesh<convex_type>;
3  using face_convex_type = Simplex<Dim-1,G_Order,Dim>;
4  using face_mesh_type = Mesh<face_convex_type>;
5  using Vh_t =  Pdhms_type<mesh_type,Order>;
6  using Wh_t =  Pdhv_type<mesh_type,Order>;
7  using Mh_t =  Pdhv_type<face_mesh_type,Order>;
8  using M0h_t =  Pdh_type<face_mesh_type,0>;
9  using Ch_t = Pchv_type<face_mesh_type,0>;
10 using product2_space_type = ProductSpaces2<Ch_ptr_t,Vh_ptr_t,Wh_ptr_t,Mh_ptr_t>;
```

Once the types are declared, we can create the mesh on the domain, on the faces of the Integral boundary conditions if needed and on the complement of it in the faces of the domain for the trace $\hat{\mathbf{u}}$, as well as the product space $\bar{\bar{V}}_h \times \mathbf{W}_h \times \mathbf{M}_h$.

Code 7.13 – Creation of the meshes and spaces for HDG Mixed Elasticity

```
1  M_mesh = loadMesh( new mesh_type);
2  // Mh only on the faces whitout integral condition
3  auto complement_integral_bdy = complement(faces(M_mesh),[this]( auto const& ewrap ) {
4    auto const& e = unwrap_ref( ewrap );
5    for( auto exAtMarker : this->M_IBCList)
6    {
7      if ( e.hasMarker() && e.marker().value() == this->M_mesh->markerName(
           exAtMarker.marker() ) )
8      return true;
9    }
10   return false; });
11
12 auto face_mesh = createSubmesh( _mesh=M_mesh, _range=complement_integral_bdy, _update=0 );
13
14 M_Vh = Pdhms<Order>( M_mesh, true );
15 M_Wh = Pdhv<Order>( M_mesh, true );
16 M_Mh = Pdhv<Order>( face_mesh, true );
17 M_M0h = Pdh<0>( face_mesh );
18
19 auto ibc_mesh = createSubmesh( _mesh=M_mesh, _range=markedfaces(M_mesh, ibc_markers),
       _update=0 );
20 M_Ch = Pch<0>( ibc_mesh, true );
21
22 auto ibcSpaces = std::make_shared<ProductSpace<Ch_ptr_t,true> >( M_integralCondition,
       M_Ch);
```

```
23      M_ps = std::make_shared<product2_space_type>(product2(ibcSpaces,M_Vh,M_Wh,M_Mh));
```

Then we can build the matrix and the vector for the bilinear and linear forms. At this point, depending on the strategy to solve the problem, monolithic or static condensation, we will determine which pattern to use for the matrix. We also create the block bilinear and linear forms associated with the matrix and vector.

Code 7.14 – Matrices and variational forms for HDG Mixed Elasticity

```
1       solve::strategy s = M_useSC ? solve::strategy::static_condensation :
            solve::strategy::monolithic;
2       solve::strategy spp = solve::strategy::local;
3
4       M_A_cst = makeSharedMatrixCondensed<value_type>(s, csrGraphBlocks(*M_ps,
            (s>=solve::strategy::static_condensation)?Pattern::ZERO:Pattern::COUPLED), *M_backend
            );
5       M_F = makeSharedVectorCondensed<value_type>(s, blockVector(*M_ps), *M_backend, false);
6
7       auto bbf = blockform2( *M_ps, M_A_cst);
8       auto blf = blockform1( *M_ps, M_F );
```

Next, we show the implementation of the assembly of the system. In particular:

- the core of the HDG matrix in Code 7.15
- the right-hand side in Code 7.16
- the Dirichlet boundary condition in Code 7.17
- the Neumann boundary condition in Code 7.18

We can access the block (i, j) of the bilinear form using integral constant known at compile time (`0\_c` for example).

Code 7.15 – Assemble core the matrix for HDG Mixed Elasticity

```
1       for( auto const& pairMat : modelProperties().materials() )
2       {
3           auto material = pairMat.second;
4           auto lambda = material.getScalar("lambda");
5           Feel::cout << "Lambda:␣" << lambda << std::endl;
6           auto mu = material.getScalar("mu");
7           Feel::cout << "Mu:␣" << mu << std::endl;
8           auto c1 = cst(0.5)/mu;
9           auto c2 = -lambda/(cst(2.) * mu * (cst(Dim)*lambda + cst(2.)*mu));
10          Feel::cout << "c1:␣" << mean(_range=elements(M_mesh),_expr=c1) << std::endl;
11          Feel::cout << "c2:␣" << mean(_range=elements(M_mesh),_expr=c2) << std::endl;
12
13          bbf( 0_c, 0_c ) +=
                integrate(_quad=_Q<expr_order>(),_range=elements(M_mesh),_expr=(c1*inner(idt(sigma),id(v)))
                );
14          bbf( 0_c, 0_c ) +=
                integrate(_quad=_Q<expr_order>(),_range=elements(M_mesh),_expr=(c2*trace(idt(sigma))*trace(id(v)))
                );
15      }
16
17
18      bbf( 0_c, 1_c ) +=
            integrate(_quad=_Q<expr_order>(),_range=elements(M_mesh),_expr=(trans(idt(u))*div(v)));
19
20      bbf( 0_c, 2_c) += integrate(_quad=_Q<expr_order>(),_range=internalfaces(M_mesh),
21                                  _expr=-( trans(idt(uhat))*leftface(id(v)*N())+
22                                           trans(idt(uhat))*rightface(id(v)*N())) );
23      bbf( 0_c, 2_c) += integrate(_quad=_Q<expr_order>(),_range=gammaMinusIntegral,
```

```
                                _expr=-trans(idt(uhat))*(id(v)*N()) );

    bbf( 1_c, 0_c) += integrate(_quad=_Q<expr_order>(),_range=elements(M_mesh),
                                _expr=(trans(id(w))*divt(sigma)));

    bbf( 1_c, 1_c) +=
        integrate(_quad=_Q<expr_order>(),_range=internalfaces(M_mesh),_expr=-tau_constant *
                                ( leftfacet(
                                    pow(idv(H),M_tauOrder)*trans(idt(u)))*leftface(id(w)) +
                                  rightfacet(
                                      pow(idv(H),M_tauOrder)*trans(idt(u)))*rightface(id(w)
                                      )));

    bbf( 1_c, 1_c) += integrate(_quad=_Q<expr_order>(),_range=boundaryfaces(M_mesh),
                                _expr=-(tau_constant *
                                    pow(idv(H),M_tauOrder)*trans(idt(u))*id(w)));

    bbf( 1_c, 2_c) += integrate(_quad=_Q<expr_order>(),_range=internalfaces(M_mesh),
                                _expr=tau_constant *
                                ( leftfacet(trans(idt(uhat)))*leftface(
                                    pow(idv(H),M_tauOrder)*id(w))+
                                  rightfacet(trans(idt(uhat)))*rightface(
                                    pow(idv(H),M_tauOrder)*id(w) )));

    bbf( 1_c, 2_c) += integrate(_quad=_Q<expr_order>(),_range=gammaMinusIntegral,
                                _expr=tau_constant * trans(idt(uhat)) *
                                    pow(idv(H),M_tauOrder)*id(w) );


    bbf( 2_c, 0_c) += integrate(_quad=_Q<expr_order>(),_range=internalfaces(M_mesh),
                                _expr=( trans(id(m))*(leftfacet(idt(sigma)*N())+
                                                      rightfacet(idt(sigma)*N())) ) );


    // BC
    bbf( 2_c, 1_c) += integrate(_quad=_Q<expr_order>(),_range=internalfaces(M_mesh),
                                _expr=-tau_constant * trans(id(m)) * (leftfacet(
                                    pow(idv(H),M_tauOrder)*idt(u) )+
                                                                      rightfacet(
                                                                          pow(idv(H),M_tauOrder)*idt(u)
                                                                          )));

    bbf( 2_c, 2_c) += integrate(_quad=_Q<expr_order>(),_range=internalfaces(M_mesh),
                                _expr=sc_param*tau_constant*trans(idt(uhat))*id(m)*
                                      ( leftface( pow(idv(H),M_tauOrder) ) +
                                        rightface( pow(idv(H),M_tauOrder) )));
```

Code 7.16 – Assemble right hand side for HDG Mixed Elasticity

```
    auto itField = modelProperties().boundaryConditions().find("stress");
    if (itField != modelProperties().boundaryConditions().end() )
    {
        auto mapField = (*itField).second;
        auto itType = mapField.find("SourceTerm");

        if ( itType != mapField.end() )
        {
            for ( auto const& exAtMarker : (*itType).second )
            {
                auto g = expr<Dim,1,expr_order> (exAtMarker.expression());
                if ( !this->isStationary() )
                    g.setParameterValues( { {"t", M_nm_mixedelasticity->time()} } );
                blf( 1_c ) += integrate(_quad=_Q<expr_order>(),_range=elements(M_mesh),
                                        _expr=trans(g)*id(w));
            }
        }
    }
```

Code 7.17 – Assemble Dirichlet boundary condition for HDG Mixed Elasticity

```
bbf( 2_c, 2_c) += integrate(_quad=_Q<expr_order>(),_range=markedfaces(M_mesh,marker),
                            _expr=trans(idt(uhat)) * id(m) );
blf( 2_c ) += integrate(_quad=_Q<expr_order>(), _range=markedfaces(M_mesh,marker),
    _expr=trans(id(m))*g);
```

Code 7.18 – Assemble Neumann boundary condition for HDG Mixed Elasticity

```
bbf( 2_c, 0_c) += integrate(_quad=_Q<expr_order>(),_range=markedfaces(M_mesh,marker ),
                            _expr=( trans(id(m))*(idt(sigma)*N()) ));

bbf( 2_c, 1_c) += integrate(_quad=_Q<expr_order>(),_range=markedfaces(M_mesh,marker),
                            _expr=-tau_constant * trans(id(m)) * (
                                pow(idv(H),M_tauOrder)*idt(u) ) );

bbf( 2_c, 2_c) += integrate(_quad=_Q<expr_order>(),_range=markedfaces(M_mesh,marker),
                            _expr=tau_constant * trans(idt(uhat)) * id(m) * (
                                pow(idv(H),M_tauOrder) ) );
blf( 2_c ) += integrate(_quad=_Q<expr_order>(),_range=markedfaces(M_mesh,marker),
                        _expr=trans(id(m))* g );
```

Finally, we solve the system either with the static condensation strategy or with the monolithic one. All the procedure described in section 2.2 is hidden in the solve method which will call the right method depending on if we have IBC or not. Then, we retrieve the stress and the displacement in the first and second block of the solution respectively.

Code 7.19 – Solve the system for HDG Mixed Elasticity

```
auto U = M_ps -> element();

bbf.solve(_solution=U, _rhs=blf, _condense=M_useSC, _name= this->prefix());

M_up = U(0_c);
M_pp = U(1_c);
```

We list here the main methods that are available in the API provided by `Feel++` in order to use this toolbox:

- `mesh()`, which allows to access the computational mesh,
- `displacementSpace()`,`stressSpace()`, which allow to access the primal and dual spaces,
- `fieldDisplacement()`,`fieldStress()`, which allow to access the primal and dual variables,
- `init(mesh=nullptr)`, which initialize the mesh, spaces, BCs, exporter,... The user can pass the mesh otherwise it is automatically loaded,
- `assembleCstPart()`, assembles the matrix of the linear elasticity system, including the matrix contributions due to BCs,
- `assembleNonCstPart()`, assembles the right-hand side of the linear elasticity system,
- `solve()`, which solves the system previously built,
- `exportResults()`, which exports the results asked in the configuration file.

In Code 7.20, we show a simple example of how to use the toolbox.

Code 7.20 – Example of how to use the API for HDG Mixed Elasticity

```
1    typedef FeelModels::MixedElasticity<nDim,OrderT> me_type;
2
3    auto ME = me_type::New("hdg.elasticity");
4    auto mesh = loadMesh( _mesh=new typename me_type::mesh_type );
5    ME -> init(mesh);
6    ME->assembleCst();
7    ME->assembleNonCst();
8    ME->solve();
9    ME->exportResults( mesh );
```

Chapter 8

Biot-Savart Reduced basis

`Feel++` had already a model order reduction framework at the start of this work [Veys, 2014], with the following features:

- CRBM for elliptic and parabolic linear coercive problems
  - primal/dual problem
  - coecivity bounds evaluation with min-θ and SCM
  - offline/online residual evaluation
  - efficient error bounds on the output
- CRBM for non affine linear coercive problems
  - EIM and error estimators
- RBM for non linear and multiphysic problems
  - EIM + SER

First we present the CRB framework of `Feel++` in figure 8.1, where the main classes are:

- `ModelCrbBase` is used to implement a new RB model. Each new RB application is implemented in a new class inherited from `ModelCrbBase`. It provides the necessary interface with the RB framework. The user can then specify the characteristics of the model: geometry, FE space, parameter space, affine decomposition... `ModelCrbBase` also comes with some options used to set up the RB algorithm (transient or steady, linear or not, ...).
- `CRB` class executes the suitable offline algorithm and produces the RB space which is eventually stored in a database.
- `CRBModel` provides the FEM resolution algorithms specific to each problem.
- `EIM` and `DEIM` are used to recover an affine approximation for non-affine or non-linear problems.

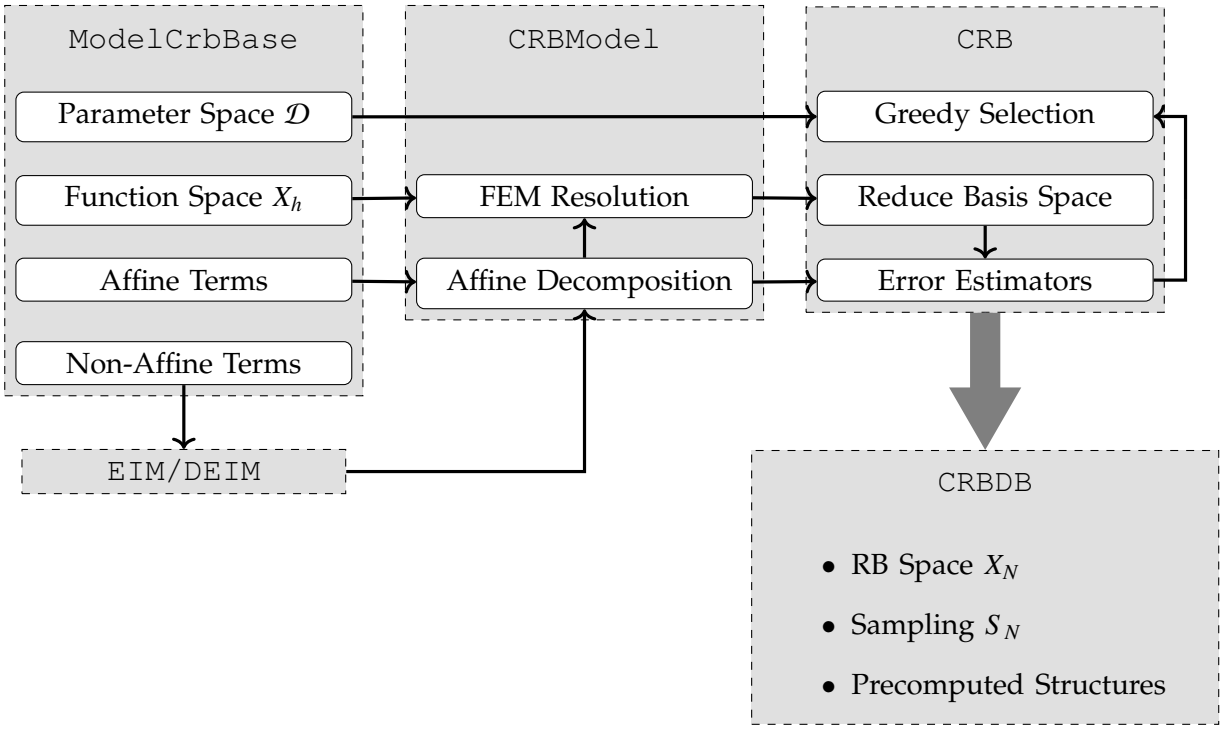


Figure 8.1 – Main classes in `Feel++`RB framework

During this thesis, we helped the implementation of composite reduced spaces for multiphysics problems [Wahl, 2018]. After that we implemented the reduced algorithm for Biot-Savart and the Empirical Quadrature Method, which are detailed respectively in section 8.1 and 8.2.

8.1 Biot & Savart Reduced

The implementation of the algorithm to reduce the computation cost of the Biot-Savart law (see 5.2.2.1) is presented in this section.

In order to be able to use our Biot-Savart code in an existing plugin for Paraview, we wanted to mimic the CRB classes. So we created a class `BiotSavartRB` which would use some of the interface of the class `CRB` to call the actual model and manage the database. It would also be the object launching the offline procedure as shown in Code 8.1. We first call the CRB offline procedure to create the reduced basis associated with the thermoelectric problem, and then we compute the integral for each reduced basis for V and EIM basis for σ.

Code 8.1 – Offline phase of `BiotSavartRB`

```
1     M_model->crbOffline();
2     // try to reload elements
3     if( M_rebuild )
4     {
5         Feel::cout << "Database␣for␣biotsavart␣not␣found,␣computing␣the␣database" <<
              std::endl;
6         for( int n = 0; n < M_model->Qb(); ++n )
```

```
7           {
8               for( int m = 0; m < M_model->mMaxB(); ++m )
9               {
10                  M_N = M_model->computeIntegrals( n, m );
11                  this->saveDB();
12                  this->saveRB();
13              }
14          }
15      }
16      Feel::cout << tc::green << "BiotSavart dimension: " << M_N << tc::reset << std::endl;
```

As in the CRB framework, the class implementing the actual model inherits from `ModelCrbBase`. It manages all the order reduction of the thermo-electric problem, by creating the corresponding object and retrieve the parameter space `Dmu`, as shown in Code 8.2.

Code 8.2 – Construction of `BiotSavart`

```
1       M_teModel = boost::make_shared<thermoelectric_model_type>(prefix);
2       M_crbModel = boost::make_shared<crbmodel_type>( M_teModel, stage);
3       M_crb = crb_type::New( M_teModel->modelName(), M_crbModel, stage);
4       M_ser = boost::make_shared<ser_type>( M_crb, M_crbModel );
5       M_Dmu = M_crb->Dmu();
6       M_N = 0;
```

We then initialize all the necessary information to build our basis, including the materials properties and where to compute the magnetic field and the electric field. Once we initialize the mesh and the FE spaces, we have to setup the communicators to manage the parallel computing of the integrals. Code 8.3 shows all those steps.

Code 8.3 – Initialization of `BiotSavart`

```
1       M_propertyPath = Environment::expand( soption(prefixvm( prefix,"filename").c_str()) );
2       M_modelProps = boost::make_shared<prop_type>(M_propertyPath);
3       M_elecMaterials = M_modelProps->materials().materialWithPhysic("electric");
4       M_magnMaterials = M_modelProps->materials().materialWithPhysic("magneto");
5       M_mesh = M_teModel->mesh();
6       M_XhCond = M_teModel->functionSpace();
7       std::vector<std::string> magnRange;
8       for( auto const& mat : M_magnMaterials )
9           magnRange.push_back( mat.first );
10      M_Xh = functionspace_type::New( _mesh=M_mesh, _range=markedelements(M_mesh, magnRange) );
11      M_XN->setModel( this->shared_from_this() );
12      this->setupCommunicators();
```

To compute the actual reduced basis for Biot-Savart, we have to get the corresponding basis for V and σ:

Code 8.4 – Retrieve the reduced basis

```
1       auto q_m = M_teModel->eimSigmaQ(mat, m);
2       auto xi_n = M_crbModel->rBFunctionSpace()->template
            rbFunctionSpace<0>()->primalBasisElement(n);
3       auto bqm = M_Xh->element();
```

As we need to compute the integrals for all points in the magnetic box, we have to find all coordinates of the dofs. As we use Lagrange elements, we know that to each point corresponds three dofs corresponding to the x, y, z components of the vectorial basis. Thus we can use one every three dofs as they are sorted geometrically, as shown in Code 8.5.

Code 8.5 – Retrieve the coordinates of the magnet box

```
            std::vector<Eigen::Matrix<double,3,1>> coords( dofSize/3 );
            // fill vector of coordinates
            for(int d = 0; d < dofSize; d+= 3)
            {
                auto dof_coord = M_dofMgn.at(i)[d].template get<0>();
                Eigen::Matrix<double,3,1> coord;
                coord << dof_coord[0],dof_coord[1],dof_coord[2];
                coords[d/3] = coord;
            }
```

We then use lambda `_e1v` and the `evaluate` function to compute the value of the integral at each coordinate, without having to recompute all the geometrical transformation each time.

Code 8.6 – Computing the integrals on the coordinates

```
            auto dist = inner( _e1v-P(),_e1v-P(),
                               mpl::int_<InnerProperties::IS_SAME|InnerProperties::SQRT>());
            auto mgnField
                = integrate(_range=markedelements( M_mesh, mat),
                            _expr=-coeff*idv(q_m)*cross(trans(gradv(xi_n)),
                                                        _e1v-P() )/(dist*dist*dist),
                            _quad=_Q<1>() ).template evaluate(coords);
```

Once the reduced basis for Biot-Savart computed, during the online phase, we compute the coefficients for a parameter `mu` by multiplying those of V, `vtN`, and those of σ, `betaSigma`. This step is presented in Code 8.7.

Code 8.7 – Computing the coefficient during the online phase

```
    vectorN_type beta(M_N);
    int index = 0;
    for( auto const& mat : M_elecMaterials )
    {
        auto betaSigma = M_teModel->eimSigmaBeta(mat.first, mu, vtN);
        for( int m = 0; m < M_teModel->mMaxSigma( mat.first ); ++m)
        {
            for( int n = 0; n < M_crb->dimension(); ++n )
            {
                beta(index++) = betaSigma(m)*vtN(n);
            }
        }
    }
    return beta;
```

8.2 Empirical Quadrature

In this section, we will present the three classes needed to implement the Empirical Quadrature Method presented in section 3.2.4. First, the linear program solved during the offline phase is managed by the class `OptimisationLinearProgramming`, interfacing the GLPK library [GLPK, 2020]. Then, we need to be able to evaluate each expression at each quadrature point, for this we implemented the `ExpressionEvaluator` class. Finally, the `EmpiricalQuadrature` class, uses the `ExpressionEvaluator` objects to fill the `OptimisationLinearProgramming` object and solve it. Those classes are summarized in figure 8.2.

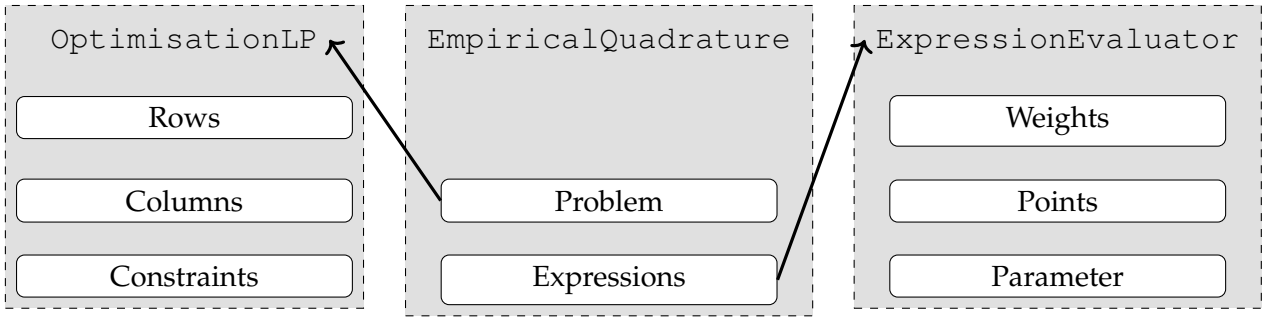


Figure 8.2 – Classes for the Empirical Quadrature Method

8.2.1 Class `OptimisationLinearProgramming`

This class interfaces the GLPK library, so at the initialization, we create the problem, give it a name and chose if we want to minimize or maximize our objective function. We then use a `glp_smcp` object to set all the parameters such as the tolerances, the time or iteration limits, or the type of scaling via command line options. The initialization is presented in Code 8.8.

Code 8.8 – Initialization of GLPK

```
1  M_pb = glp_create_prob();
2  glp_set_prob_name( M_pb, name.c_str() );
3  glp_set_obj_dir( M_pb, direction );
4  M_params = new glp_smcp;
5  glp_init_smcp(M_params);
6  M_params->msg_lev = ioption("glpk.verbosity");
7  M_params->meth = ioption("glpk.method");
8  M_params->tol_bnd = doption("glpk.tolerance-bounds");
9  M_params->tol_dj = doption("glpk.tolerance-dual");
10 M_params->tol_piv = doption("glpk.tolerance-pivot");
11 M_params->it_lim = ioption("glpk.iteration-limit");
12 M_params->tm_lim = ioption("glpk.time-limit");
13 M_params->presolve = ioption("glpk.presolve");
14 M_scaling = ioption("glpk.scaling");
```

To add a row in our problem, we need to give it a name and the bounds of the variables.

Code 8.9 – Add a row

```
1  int r = glp_add_rows( M_pb, 1);
2  glp_set_row_name( M_pb, r, name.c_str() );
3  glp_set_row_bnds( M_pb, r, type, lb, ub );
```

To add a column to our problem, we need to give it a name, the bounds of the variables and the associated coefficient.

Code 8.10 – Add a column

```
1  int c = glp_add_cols( M_pb, 1);
2  glp_set_col_name( M_pb, c, name.c_str() );
3  glp_set_col_bnds( M_pb, c, type, lb, ub );
4  glp_set_obj_coef( M_pb, c, coef );
```

To fill the matrix of constraints, we need three arrays: row indices of each element are stored in the array `ia`, column indices are stored in the array `ja`, and numerical values of corresponding elements are stored in the array `ar`. We then call the

`glp_load_matrix` function with the problem, the size and these arrays as detailed in Code 8.11.

Code 8.11 – Set the constraints

```
int k = 1;
int R = matrix.size();
if( R == 0 )
    return;
int C = matrix[0].size();
int ia[1+C*R], ja[1+C*R];
double ar[1+C*R];
for( int r = 0; r < R; ++r)
{
    for( int c = 0; c < C; ++c)
    {
        ia[k] = r+1;
        ja[k] = c+1;
        ar[k] = matrix[r][c];
        ++k;
    }
}
glp_load_matrix( M_pb, C*R, ia, ja, ar );
```

The last function consists in solving the problem. Before that, we call `glp_scale_prob` to choose between the possible choices: geometric mean scaling, equilibration scaling, round scale factors to nearest power of two, or skip the scaling. Alternatively, we can let GLPK choose the appropriate scaling automatically. After that we can apply the simplex method to our problem to solve it, as shown in Code 8.12.

Code 8.12 – Scale and solve the problem

```
glp_scale_prob(M_pb, M_scaling);
return glp_simplex(M_pb, M_params);
```

8.2.2 Class `ExpressionEvaluator`

This class manages the evaluation of an expression at the quadrature points. As the expressions can depend on parameters or can be non-linear, their type cannot be known beforehand, so we created a base class with the necessary interface, presented in Code 8.13. Then we use templated subclasses by the type of the expression, depending of the dependence of the class on parameters or FE field.

Code 8.13 – Interface for the evaluator

```
virtual void init(int o) = 0;
virtual void update(element_type const& elt) = 0;
virtual bool update(parameterelement_type const& mu) = 0;
virtual int nPoints() = 0;
virtual double weight(int i) = 0;
virtual double eval(int q, int comp) = 0;
virtual int order() = 0;
int component() { return M_comp; }
```

To evaluate an expression on a point, we need to have the context of this point, meaning the element in which it belongs. We also need to construct the evaluator using the geometric map and this context. Finally, as described in Code 8.14, we can also retrieve the quadrature weights of the quadrature rule.

Code 8.14 – Initialization

```
1   auto const eltForInit = boost::unwrap_ref(*boost::get<1>(this->M_range));
2   auto q = _Q(order);
3
4   using iim_type =
        vf::detail::integrate_im_type<decltype(this->M_range),decltype(M_expr),decltype(q),decltype(q)>;
5   auto ims = iim_type::im(q,q,M_expr);
6   auto im = ims.first;
7   auto pts = im.points();
8   auto gm = eltForInit.gm();
9   auto geopc = gm->preCompute( pts );
10
11  M_ctx = gm->template context<vm::POINT|vm::JACOBIAN|expr_type::context>( eltForInit,
        geopc );
12  M_evaluator = std::make_shared<evaluator_type>(M_expr.evaluator( mapgmc(M_ctx) ) );
13  M_weights = im.weights();
```

When we want to evaluate the expression on a specific point, we first need to update the context and the evaluator with this element.

Code 8.15 – Update for a new element

```
1   auto const& elt = unwrap_ref( eltWrap );
2   M_ctx->update( elt );
3   M_evaluator->update( vf::mapgmc( M_ctx ) );
```

After update the context and the evaluator, we can evaluate the expression by passing the index of the quadrature point in the element, and the component that we want to evaluate. We also need to multiply by the Jacobian, given by the context.

Code 8.16 – Evaluation at an interpolation point

```
1   return M_evaluator->evalq(comp,0,q)*M_ctx->J(q);
```

The dependence on a parameter `M_mu` and a FE function `M_u` is managed by the use of reference. When the reference of the parameter or of the function is updated, the expression is automatically updated.

Code 8.17 – Reference for the parameters and field

```
1   parameterelement_type& M_mu;
2   function_element_type& M_u;
```

8.2.3 Class `EmpiricalQuadrature`

This class links the previous two, by collecting the `ExpressionEvaluator` corresponding to the set of expressions g_m and creating the linear programming problem. Each expression is added to a vector of `ExpressionEvaluatorBase` via the method `addExpression` templated by the type of the expression and possibly by the type of the FE function. In each of these functions, the corresponding subclass of `ExpressionEvaluatorBase` is instantiated and added to the collection.

Code 8.18 – Adding expressions

```
1   template<typename ExprT>
2   void addExpression(ExprT& ex, int comp = 0);
3   template<typename ExprT>
4   void addExpression(ExprT& ex, parameterelement_type& mu, int comp = 0);
5   template<typename ExprT, typename FctT>
6   void addExpression(ExprT& ex, parameterelement_type& mu, FctT& u, fct_type<ExprT,FctT>
        const& f, int comp = 0);
```

Once everything is initialized, we can compute all the evaluations of the expressions. For each parameter in the trainset, we update each expression with it, then for each element on the range, for each expression we update the context and evaluator with the element, and we evaluate the expression for each quadrature points. Those loops are presented in Code 8.19.

Code 8.19 – Evaluation of all the quadrature points

```
for( j = 0; j < M_J; ++j )
{
    for( m = 0; m < M_M; ++m )
        M_exprevals[m]->update(M_trainset->at(j));
    n = 0;
    for( auto const& eltWrap : M_range )
    {
        auto const& elt = unwrap_ref( eltWrap );
        if ( elt.processId() != Environment::rank() )
            continue;

        for( m = 0; m < M_M; ++m )
        {
            M_exprevals[m]->update( eltWrap );
            for ( uint16_type q = 0; q < nPts; ++q )
            {
                eval[m][j][n+q] = M_exprevals[m]->eval(q, M_exprevals[m]->component());
                res[m][j] += M_exprevals[m]->weight(q)*eval[m][j][n+q];
            }
        }
        n += nPts;
    }
}
```

Once we have all the evaluations, we create the linear program. We first add the rows corresponding to all the expressions for each parameter of the trainset, with the bounds set to the evaluations multiplied by the weights plus/minus the tolerance. We then add the columns of the problem corresponding to the optimized weights, with a coefficient of 1 and lower bounded by 0. Next we set the matrix with our evaluations and finally we solve our problem to find the non-zero weights for our set of expressions. This procedure is detailed in Code 8.20.

Code 8.20 – Solve the linear program

```
auto glpk = OptimizationLinearProgramming( GLP_MIN, M_prefix );

for( int m = 1; m <= M_M; ++m )
    for( int j = 1; j <= M_J; ++j )
        glpk.addRow((boost::format("x_%1%%2%")%m%j).str(), GLP_DB, res[m-1][j-1]-M_tol,
            res[m-1][j-1]+M_tol );

for( int n = 1; n <= M_N; ++n)
    glpk.addColumn( (boost::format("p_%1%")%n).str(), 1.0, GLP_LO, 0.0, 0.0 );

std::vector<std::vector<double> > eval2(M_M*M_J);
for(int m = 0; m < M_M; ++m)
    for(int j = 0; j < M_J; ++j)
        for(int n = 0; n < M_N; ++n)
            eval2[m*M_J+j].push_back(eval[m][j][n]);
glpk.setMatrix(eval2);

int e = glpk.solve();
```

Part IV

Applications

In this part, we will show results on geometries of real magnets used at the LNCMI. Such geometries are obtained through CAD softwares, such as Salome [Ribes et al., 2017]. Specific plugins have been developed in order to create the different parts of a magnet, the helices, the rings linking them and the inlet and outlet for the electric current. Creating only the geometry can take several hours, and once done, the mesh is generated by MeshGems, which is integrated in Salome.

The first fully coupled model of resistive high field magnet on such geometries has been described in [Daversin Catty, 2016] and [Daversin et al., 2016a] [Daversin et al., 2016b]. During this work, more systematic simulations were carried out with a special focus on validations. The main goal here is to achieve more reliable models by using actual data from magnets in operation and by comparing results with experimental data and data from the magnet monitoring system. The main results obtained will be presented on Chapter 9. We will show how the use of HDG method may improve the quality of the simulation and how CRB methods can be used in this context.

Finally, in Chapter 10, we will explore the potential of CRB methods applied to a shape optimization problem. This final application can open the path to the development of high field magnets with an enhanced homogeneity for the solid state NMR scientific community.

Chapter 9

Identification of Cooling parameters

As previously presented, High Field Magnets are electromagnets powered with up to 31 kA to deliver up to 37 tesla. To dissipate the heat produced by the Joule losses, the magnets are cooled by a water forced flow (up to $140\,l/m^3$ with a pressure drop of 25 bars). Magnets are actually composed of a poly-helices insert, and a bitter insert disposed in a concentric way. The poly-helices insert is the innermost one. Each insert is set into a separate housing with a specific power supply and cooling system. In the sequel, we will refer to "tranche" for these separate systems.

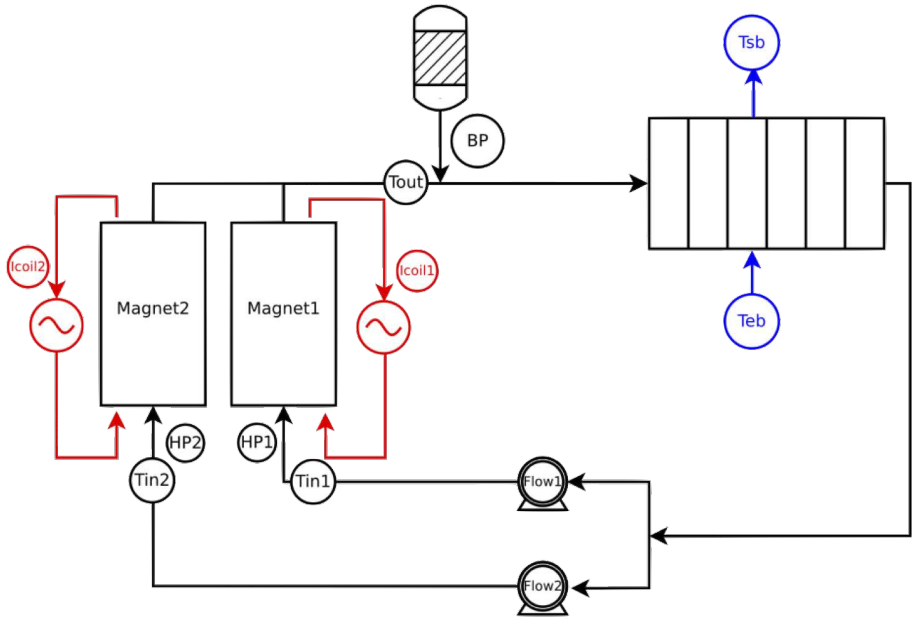


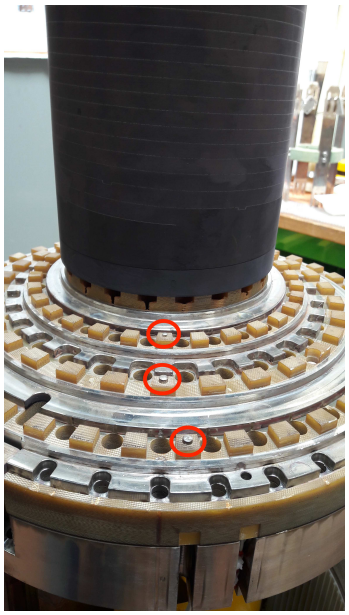
Figure 9.1 – Hydraulic circuit schema with localization of the tensors

Such environment (water and high pressure) makes it difficult to instrument the magnets. The only available measures, depicted in figure 9.1, come from:

- voltage taps on the connection ring between consecutive tube for the poly-helices insert,

- water temperature probes at the inlet and outlet of the cooling circuit,
- flow rate sensor.

On the figure 9.2a, the voltage taps disposed on the BP plateau are circled in red. Once the insert is totally assembled, they are in contact with the rings connecting two consecutive helices. On figure 9.2b is shown the monitoring of a magnet during operation, with the values of the different sensors.



(a) Tension sensors

(b) Logs of magnet monitoring system

Figure 9.2 – Monitoring Magnets

In addition to these data, we also have access to:

- the total voltage drop V,
- the total electric power P,
- and the field factor f, i.e. the ratio of the magnetic field $B_z(0)$ at the magnetic center over the input current I.

f is experimentally obtained by measuring the magnetic field profile along the magnet Oz axis at a given current. Note that this actually relies on the hypothesis of linear dependency of B versus I.

The magnet is actually operated by setting the input current I. In practice, the user requests a value of $B_z(0)$ on a console which, in turn, sends an order to power supply to deliver a current $I = B_z(0)/f$. During this operation, the monitoring system receives data from the voltage taps U_i, water temperatures probes (T_{in}, T_{out}) and flow rate Q sensor.

The control system relies on these data to determine if the operating magnet is safe or not. More precisely it checks the deviation $dR0_i$ of the resistance measured for each

voltage tap (i.e. $R0_i = \frac{U_i}{I}$) from a reference value. A heuristic threshold of 3% is considered as an indicator of a wrong behavior. Once this threshold is reached, a signal is sent to power supply, leading to a controlled power shutdown. This means that I is driven to 0 in a controlled manner (typically in roughly a tenth of second).

The reference values for $R0_i$ are determined during the magnet commissioning following this process:

- For I_n in a set of input current values
- Measure U per helix (or couple of helices)
- Compute $R = \frac{U}{I}$
- Repeat for new I_{n+1} until 30 kA is reached

Then, a fit for each R as a 2nd order polynomial of I is computed. This fit will also be useful to have an estimate of the mean temperature $< T >$ of the helix (or couple of helices):

$$R = R(I = 0)\,(1 + \alpha(< T > -T_0))$$

with T_0 a reference temperature (generally 20 C).

In a first example, we will consider a magnet in operation during a steady state. The simulations will be carried out with several heat exchange cooling models. The goal is to find the cooling parameters that would provide the results that match best the control/command voltage measurements.

Using the best cooling model according to what has been found in the first example, we will simulate the commissioning of a magnet. The objective, here, is to find estimates for the field factor f and the reference resistance for each voltage taps.

Contents

9.1 A Magnet in operation

As stated in introduction, we consider in this section an actual magnet configuration. The setup, we model, is a so-called plateau. The input data - i.e. input currents and voltages, inlet water temperature - are retrieved from the control/monitoring system log file corresponding to the experiment selected.

The objective of this study is to find the water cooling parameters that gives the voltage drop per helices that best match the measured data.

9.1.1 Water temperature and heat exchange coefficients

Since modeling the thermohydraulic problem is out of the scope of this study, the cooling is modeled by Robin conditions on the temperature field, of the type $h(T - T_w)$, as seen in chapter 4. These parameters - namely h the heat exchange coefficient and T_w the water temperature - are not known precisely.

Indeed, as seen in the figure 9.1, if we have access to the flow rate and the inlet temperature for each tranche, we only have the outlet temperature for the whole magnet, that is an average between the water coming out respectively from the Polyhelix and the Bitter insert.

We need to determine the temperature of the water T_w and the heat transfer coefficient h that provide the lowest error between measured and computed electric potential.

We consider different correlations for the heat exchange coefficient:

- Montgomery [Montgomery, 1969]:

$$h = 1426\,(1 + 0.015T_w)\,\frac{u^{0.8}}{D_h^{0.2}}$$

- Colburn [Colburn, 1933]:

$$h = \frac{\kappa(T_w)Nu}{D_h} = \frac{\kappa(T_w)}{D_h}\alpha Re^n Pr^m$$

  with $\alpha = 0.023, n = 0.8, m = 0.3$

- Dittus [Dittus and Boelter, 1985]:
  same as Colburn but with $\alpha = 0.023, n = 0.8, m = 0.4$

- Silberberg [Silberberg et al., 2009]:
  same as Colburn but with $\alpha = 0.015, n = 0.85, m = 0.3$

Since the Nusselt number Nu depends on water temperature and pressure, we retrieve these data from the monitoring system. It also depends on the water velocity, which is not available directly. So we assume that the water velocity is the same per channel, given by the ratio of water flow divided by the total section of the cooling channels. The temperature elevation dT_w along each channel Γ is directly obtained by the calorimetric balance:

$$\frac{\int_{\partial\Gamma} h(T - T_w)ds}{\rho C_p v_w S} \tag{9.1}$$

Several types of water temperatures are then used:

- a constant temperature in all the channels:

$$T_{w,i} = T_{in} + dT_w/2$$

- a global gradient of temperature with z_{min} and z_{max} the minimum and maximum of the whole magnet:

$$T_{w,i} = T_{in} + dT_w(z - z_{min})/(z_{max} - z_{min})$$

- a gradient of temperature per channel with the actual channel's length:

$$T_{w,i} = T_{in} + dT_w(z - z^i_{min})/(z^i_{max} - z^i_{min})$$

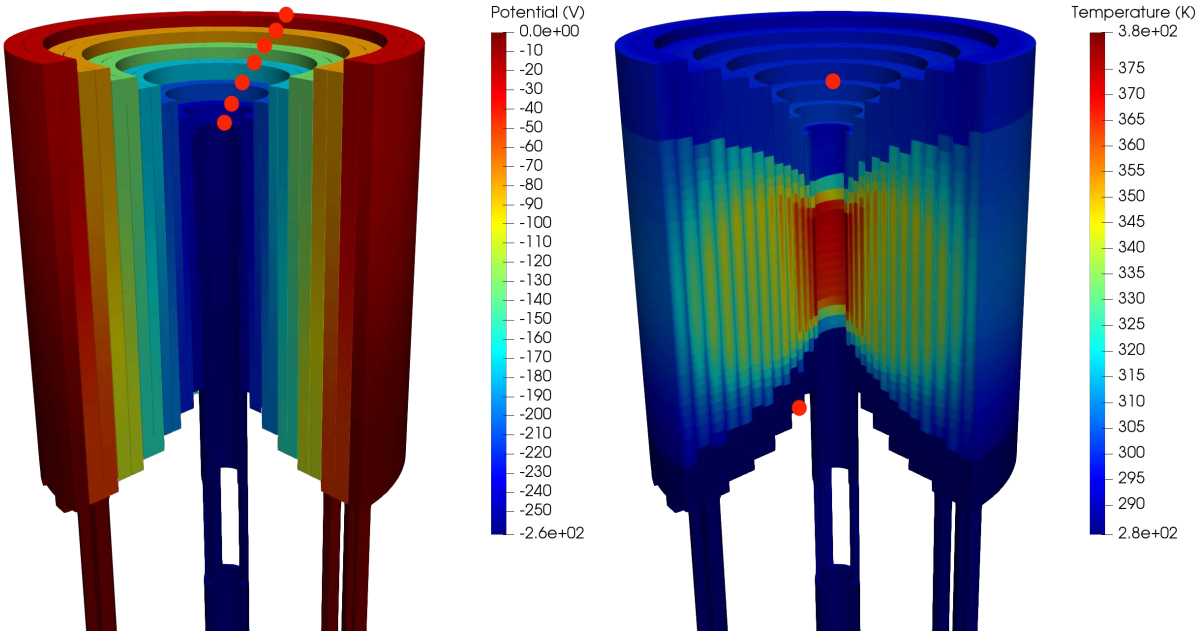
All these configurations have also been tested with dT_w and h computed separately for each channel or in a unique way for all channels. Those represent 24 different configurations.

9.1.2 Modeling a plateau

We ran the simulation on a 14 helices insert, where a voltage tap per helix has been implemented and two additional temperature probes were added in connection rings on either HP and BP side. This configuration corresponds to an experiment carried out in April 2017 : the magnet ran at an input current I_0 of 22148.2 A.
We will compare:

- the voltage tap measurements on HP and BP sides (14 measurements)
- the temperature measurement in the rings H2H3 and H3H4

An image of the electric potential and the temperature on the 14 helices of the magnet can be seen in figure 9.3. The sensors are represented as red dots.



(a) Electric potential (b) Temperature

Figure 9.3 – Actual magnet in operation at I=22148 A

9.1.2.1 Continuous Galerkin

To reach the requested state, we run the code iteratively until we reach the value I_0 and dT_w does not change between two iterations. The stopping criterion is set to 0.01%. The computation has been realized on a cluster, using 2 compute nodes, each with 24 cores on 2 sockets (Intel Weon E5-2680 v4 2.40GHz). The mesh has 22 million elements, and each function space has 4.3 million degrees of freedom. Using Picard algorithm, it takes around 500s per iteration for I_0 and dT_w, and it takes at least 3 iterations to reach convergence of those values.

The results for the voltage taps are presented in Figures 9.4 for the constant water temperature, with h and dT_w computed only once for all the channels. The Figure 9.5 shows the results for the gradient of temperature with the channel's actual lengths. Note that the voltage tap 3 is not shown as it was not functional at the time of the experiment.

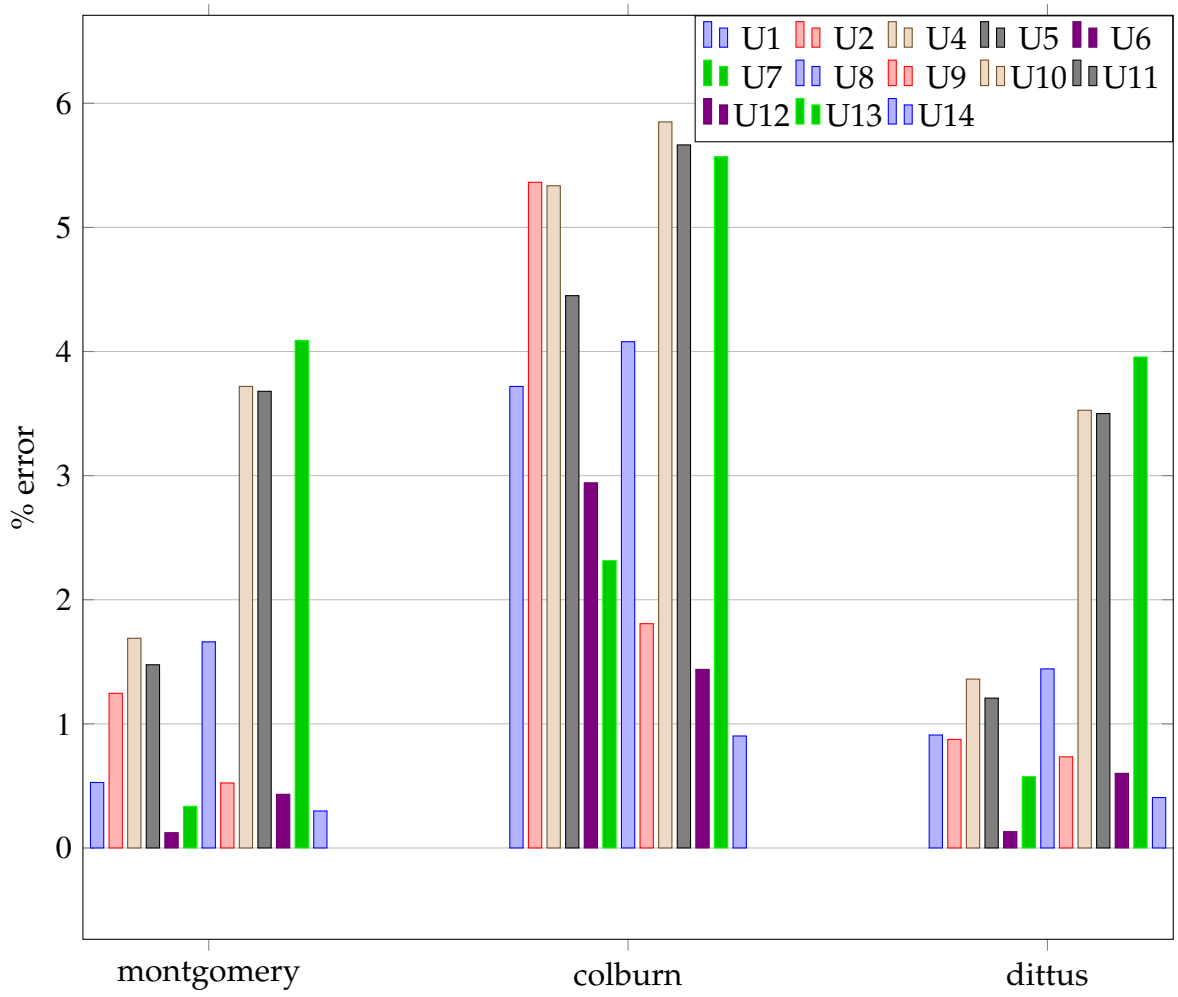


Figure 9.4 – Errors on the tension for CG and constant water temperature

As expected, when modeling the temperature more accurately, we obtain better results. We can see that we overestimate the tension in most of the helices. But, for the Montgomery and Dittus correlations, we retrieve the measured values with less than

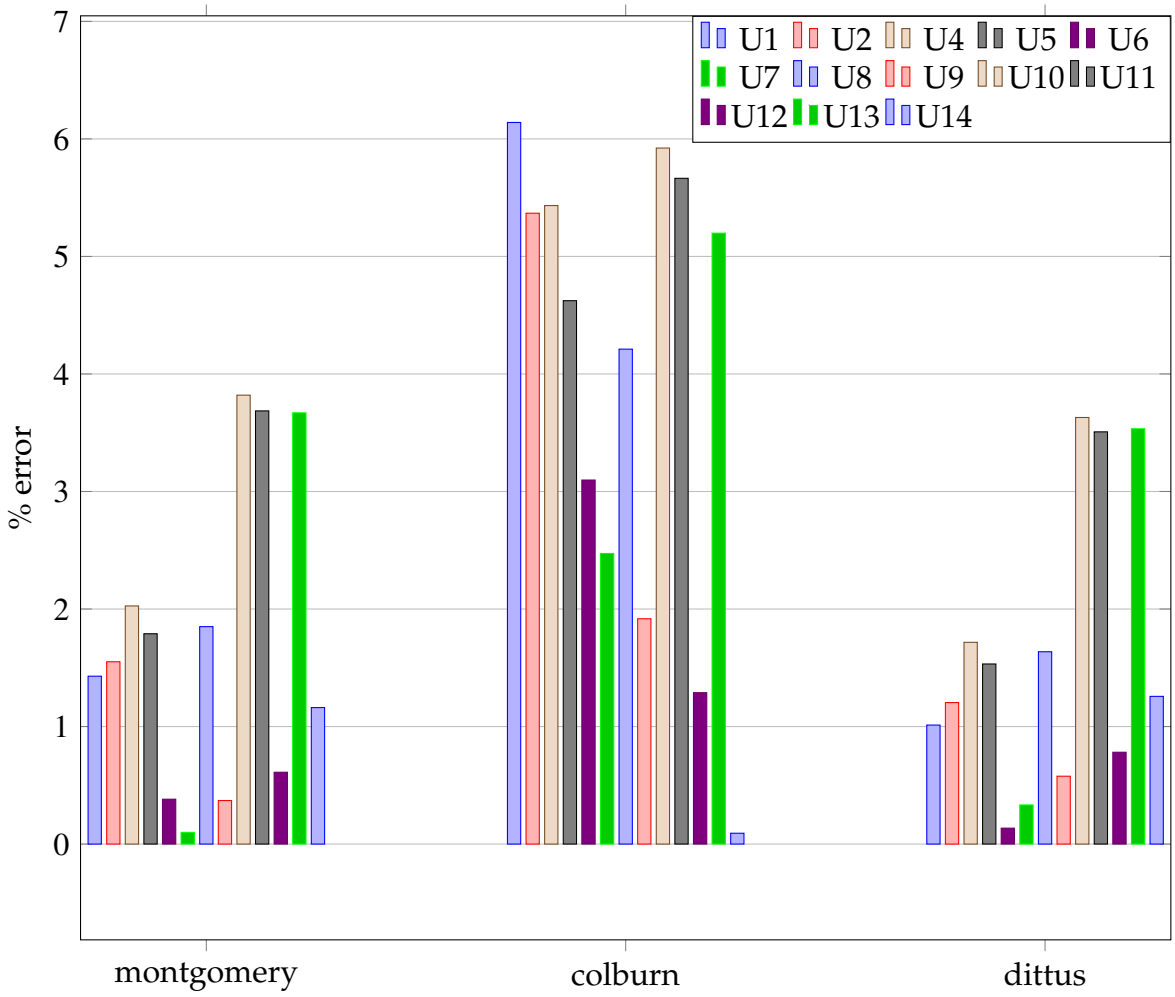


Figure 9.5 – Errors on the tension for CG and gradient water temperature

4% of error.
With the Dittus correlation, the temperatures computed are 284.66 K at HP and 302.35 K at BP, whereas the measured values are 287.95 K and 305.35 K respectively. This means that we have less than 1% of error on the temperature.

9.1.2.2 Hybrid Discontinuous Galerkin

For the HDG model, we use the integral boundary condition to impose directly the value of I_0. The computation has been realized on the same cluster than in the previous chapter, on the same number of cores. The mesh was coarser than in the previous case, it has 3.3 million elements, the discontinuous vectorial space for the flux has 39 million dofs, the discontinuous scalar space for the potential has 13 million dofs and the discontinuous scalar space for the trace has 22 million dofs. Using Picard algorithm, it takes around 1600s to solve the problem.

The results for the gradient of temperature with the channel's actual lengths are presented in figure 9.6 and a comparison for the Dittus correlation between CG and HDG is shown in figure 9.7.

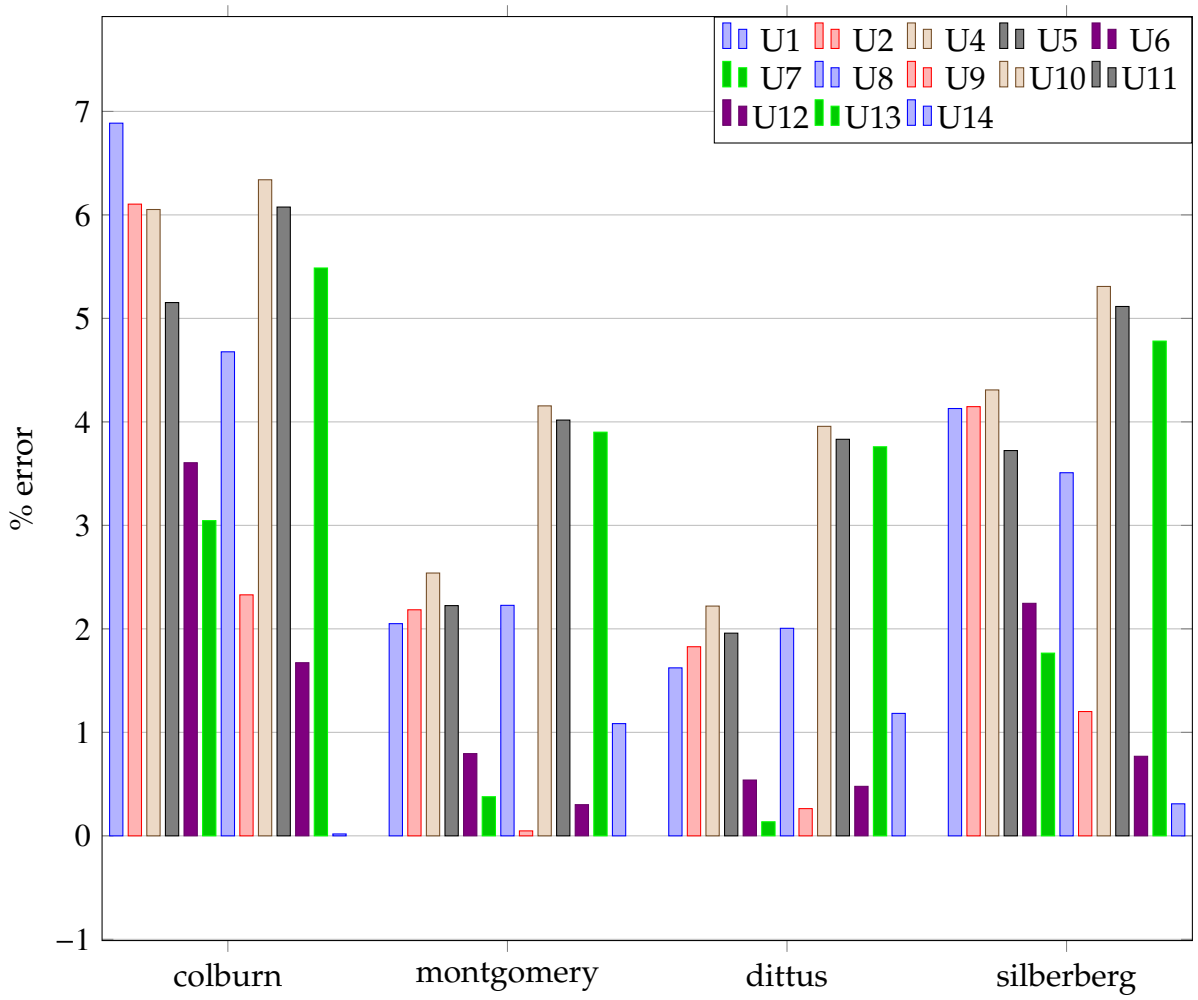


Figure 9.6 – Errors on the tension for HDG

We can see that the Dittus and Montgomery correlations give once again the best results, less than 4% of error for Dittus, even if the errors are slightly above the ones obtained using the CG discretization. For the temperature, we have very similar results with the CG discretization, 284.73 K at HP and 302.20 K at BP, whereas the measured values are 287.95 K and 305.35 K respectively. This means that we have less than 1% of error on the temperature.

9.1.2.3 Reduced order model

Even if we have small errors for all helices, we noticed that the voltage taps on the 10, 11 and 13 th helices present higher errors than the others. This can be due to the state of the magnet on which the measurements have been done, indeed this magnet was not new and thus could have deformed during previous experimentations, thus leading to wrong estimations of the heat transfer coefficients.
Here we will use the model described in section 4.3 to find the heat transfer coefficients and the temperature of the water per channel that allows us to be as close as possible to the measured values.

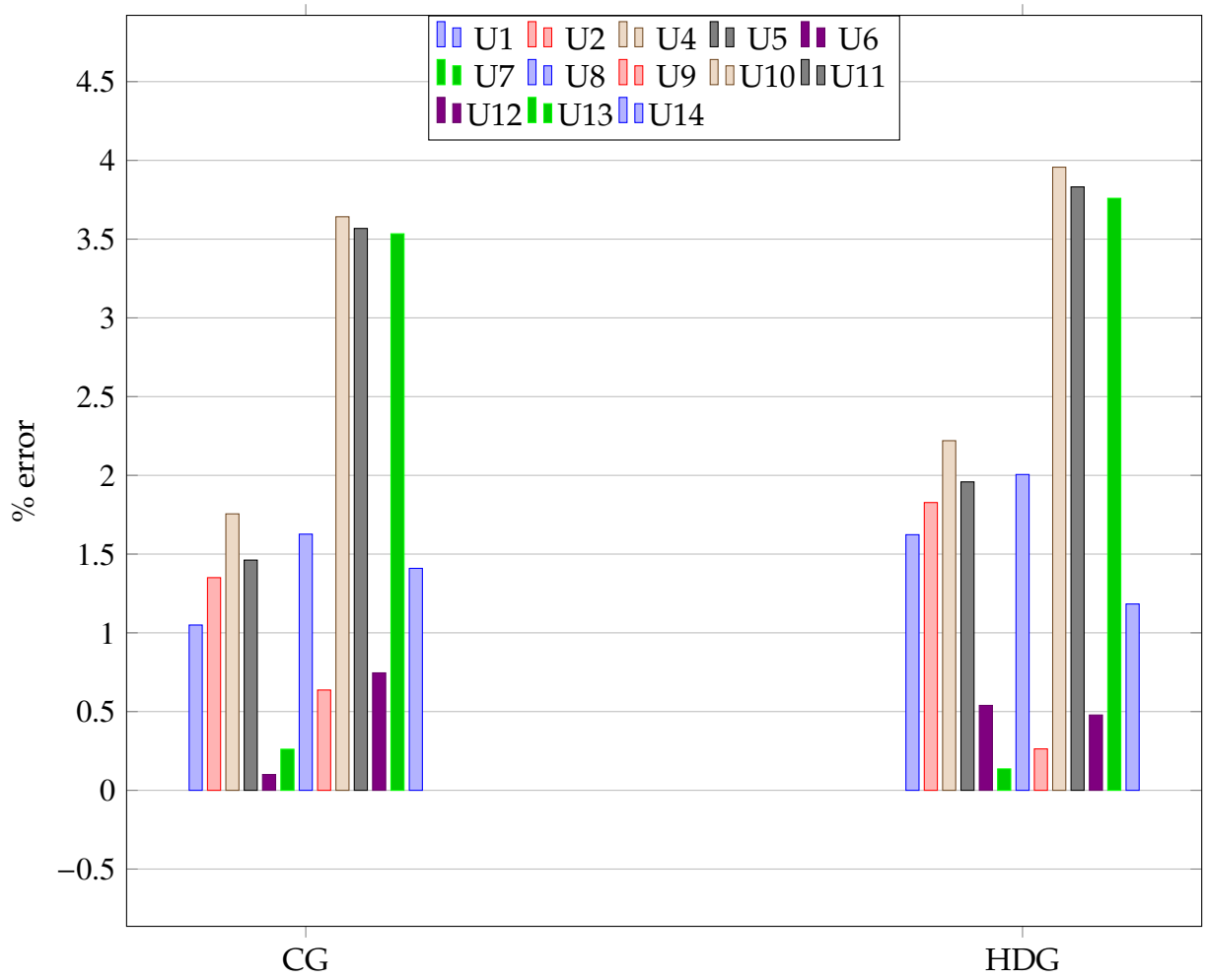


Figure 9.7 – Comparison of the errors on the tension between CG and HDG

Problem

Here, we fix the parameters σ_0, α, L, V_D with the values used in the precedent section. Thus, the parameters of our reduced model are h_i and T_w^i, for $i = 1, ..., 14$, the heat transfer coefficient and water temperature for each channel. The temperature will range from the inlet temperature to the outlet temperature given by the monitoring system, whereas the coefficients will range around 25% of the Montgomery correlation.

We need to solve an optimization problem where we want to minimize the following function:

$$f(\boldsymbol{\mu}) = \sum_{i=1}^{14} \frac{|U_i - U_i^m|}{|U_i^m|} + \frac{|T_{HP} - T_{HP}^m|}{|T_{HP}^m|} + \frac{|T_{BP} - T_{BP}^m|}{|T_{BP}^m|} \tag{9.2}$$

where U_i and U_i^m are respectively the tension at the tap i computed and measured, and T_{HP}, T_{HP}^m, T_{BP} and T_{BP}^m are the temperature on the HP side computed and measured, and the temperature on the BP side computed and measured.

For this problem, we have 28 parameters, and 29 different materials (14 helices, 13 rings to connect them, and the inlet and outlet of current), and thus 59 EIM approxi-

mations. With so much EIMs and dofs, the creation and storing of the matrices take more than 500Go of memory, so it is not viable to use this strategy.
As an alternative, we are investigating the use of DEIM, in order to group all EIMs, and/or to restrict the parameters to only a few helices.

The results will be available soon.

9.2 Magnet Commissioning

Before setting a polyhelices magnet into operation, two experiments are carried out to respectively measure the factor field of the new magnet (i.e. the ratio between the magnetic field value at the center of the magnet and the input current) and to record the resistance of a helix or a couple of helices as a function of the input current. Schematically these resistances data will then be used to control the magnet behavior. Deviations from these recorded values above a given threshold indicate that some problems are occurring into the magnet. This will trigger an alert leading to a controlled magnet shutdown.
The monitoring of the current during the commissioning is shown in figure 9.8, where we can see different plateaus with their intensities used.

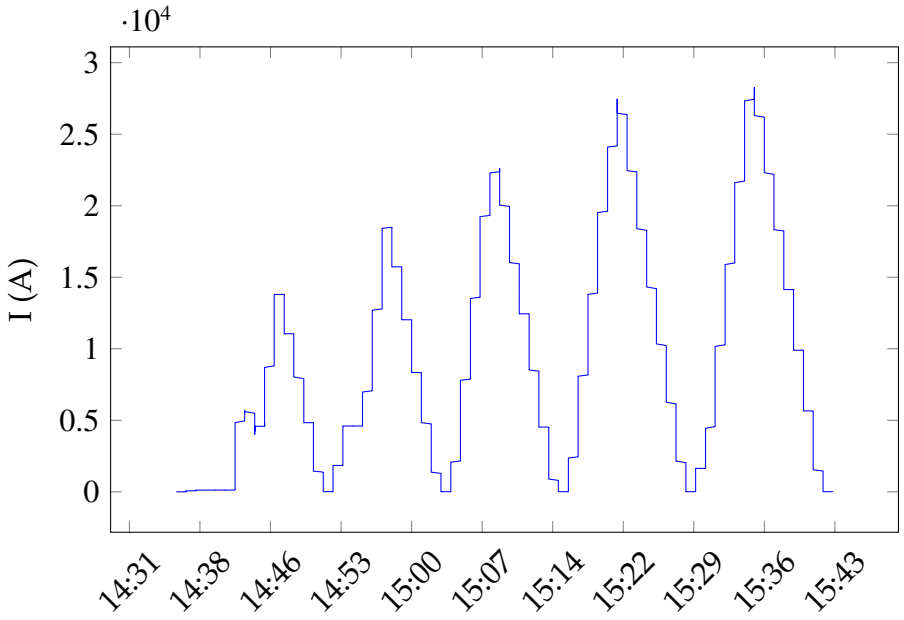


Figure 9.8 – Current for the first tranche of a magnet over time

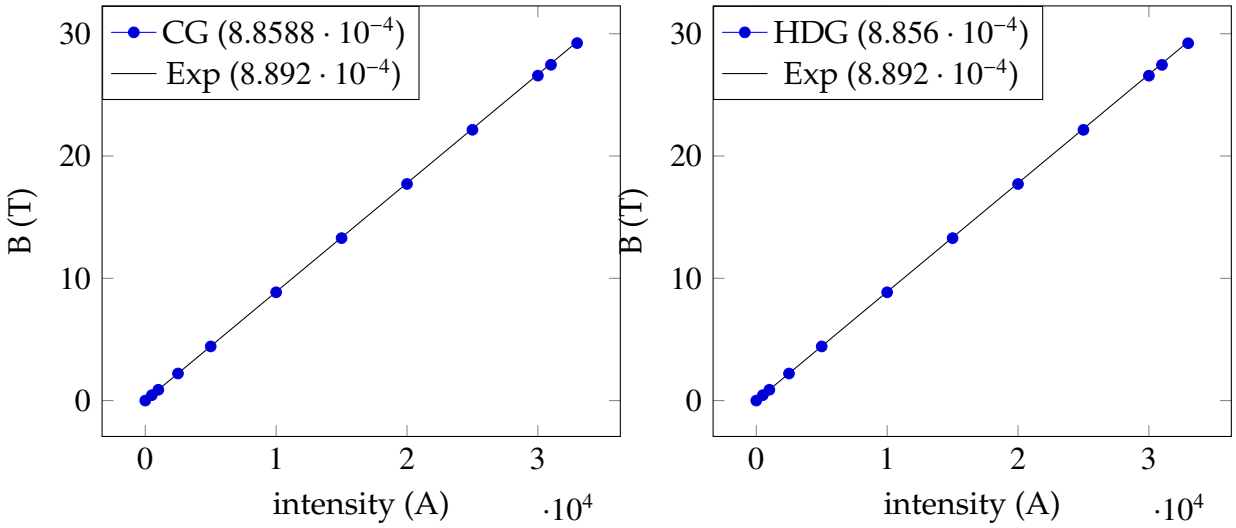
So far, this operation was performed without any prior calculations except for the simulation of the polyhelices magnet at full power. Following the validation of our numerical model on the simulation of a magnet plateau, we have tried to reproduce numerically the commissioning experiments.

In standard operation the water flow Q in each tranche is dependent on the input current I on the corresponding magnet. More precisely, the pump rpm V_{pump} is function of I. We recall that a tranche corresponds to a magnet. In the subsequent simulation

we only consider the polyhelices magnet. We assume that Q is simply proportional to V_{pump}. Similarly the pressure drop in the tranche P is assumed to be proportional to V_{pump}. It is to be noted that this is a rough approximation of the cooling hydraulic circuit. To mimic the commissioning, we will run HiFiMagnet simulations at various I with a Dittus correlation (see section 9.1.1) per cooling channel. The water temperature in each cooling channel will be considered constant and equal to the mean temperature, i.e. $(T_{in} + dT_w)/2$. T_{in} is the input water temperature. The heat coefficients and dT_w per cooling channels are computed from calorimetric balances for each input current.

Field Factor

For the field factor, we will use Biot-Savart to compute the z component of the magnetic field at the center of the magnet. In CG, we need to loop over the difference of potential to approximate the current I wanted, we then compute j as $-\sigma\nabla V$. In HDG, we can directly impose the current I using the Integral Boundary Condition, and we have directly access to j.



(a) CG (b) HDG

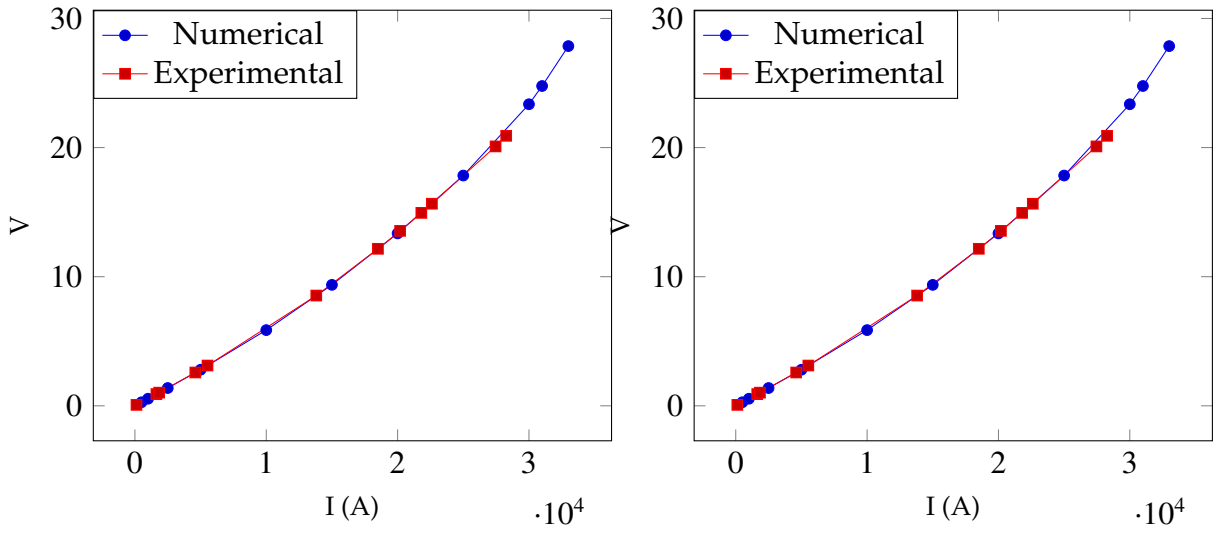
Figure 9.9 – Field factor experimental vs numerical

The experimental value of the field factor for the magnet used is $8.892e^{-4}$ tesla/A. In CG, we find 8.8588e−4 T/A, which gives an error of 0.37%, whereas using HDG, we find $8.856e^{-4}$ T/A, which gives an error of 0.40%, as presented in figure 9.9. We conclude that both discretizations can be used to reproduce numerically the commissioning experiments, even with a coarser mesh for HDG.

Resistance per couple of helices

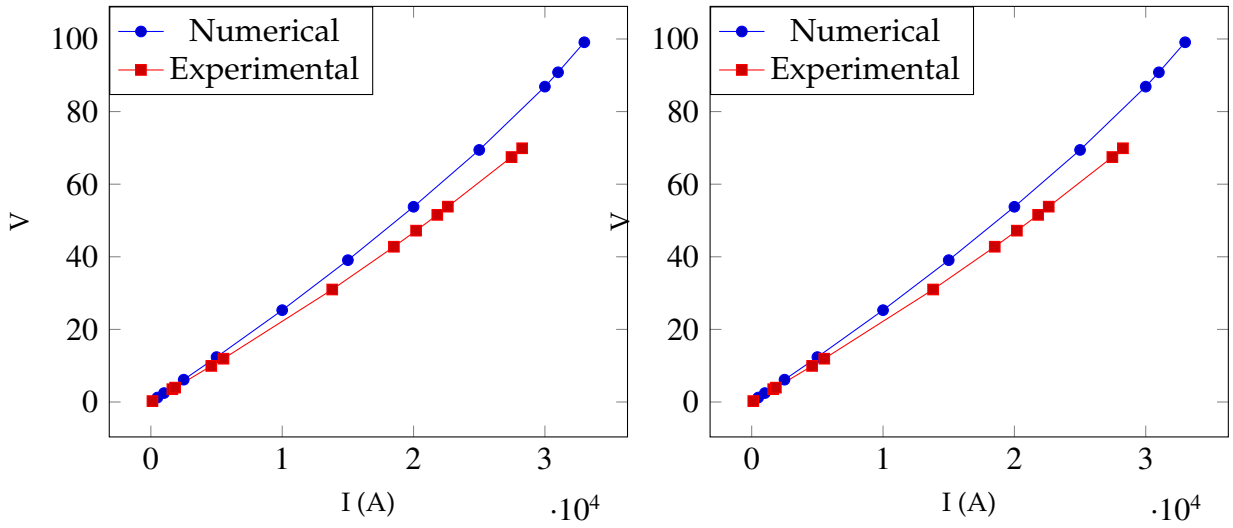
Actual setup of the monitoring system only record the voltage taps per couple of helices. During commissioning we record the resistance per couple of functions of I. These values, as explained in introduction, will serve as a reference. In operation, resistance values are compared to this reference to track deviation further than a threshold.

Using results from previous simulations, we can easily have access to the resistance per couple of helices. Figure 9.10 shows the experimental vs numerical resistance for the first couple of helices, both in CG and HDG, whereas figure 9.11 shows the same for the last couple.



(a) CG (b) HDG

Figure 9.10 – Resistance of couple 1: experimental vs numerical



(a) CG (b) HDG

Figure 9.11 – Resistance of couple 7: experimental vs numerical

For the first couple of helices, we can see that we are on good agreement, both in CG and HDG, with the experimental values of the resistance. But for the last couple of helices, the computation does not coincide to the experimental data, especially for high intensity.
A possible explanation is that the commissioning of the magnets are done relatively quickly, a few minutes, due to the cost of electricity and in order to keep a constant

temperature of the water from the river. The hypothesis is that it is sufficient to reach a plateau, where the dynamic effects are over, and we can use a static model for the simulation. If that is the case for the first couple of helices, it seems that this hypothesis does not hold for the last couple. It can be due to the fact that the last helices are much more massive than the first ones, and so have a higher inertia, taking longer to reach the plateau than the other helices.

9.3 Conclusion

In this chapter we presented an application of our models to identify the cooling parameters that were used in an experimentation conducted at the LNCMI. In particular, we saw the importance of the choice of the correlation for the heat exchange parameter h, with the Dittus and Montgomery ones leading to the best results. It allowed us to retrieve the experimental values with an acceptable error, less than 4% for the tensions and less than 1% for the temperature.
We also retrieve the control parameters of the field factor and the resistance of the magnet for both methods. If for the first helices, the parameters are retrieved accurately, it is less accurate for the last helices. We think that the static model might be not enough to understand correctly what is going on during the commissioning, and a dynamic model could be more accurate in this case.
When trying to enhance the accuracy of our parameters with the use of the reduced basis method, we saw the difficulty of scaling up to real geometries. Naively using the method on such problems induce a huge memory cost, leading to the use of alternative strategy, such as DEIM, or splitting the problems on several helices.
This application, nonetheless, showed that our models can be used by the LNCMI to control efficiently the magnets used in experimentation.

Chapter 10

Geometrical optimization

The use of the reduced order method allows us to compute a large number of simulation for different parameters. This gives us the possibility to do optimization or control that would not have been possible with a full order model.

Contents

10.1 Homogeneity optimization

For NMR solid state physic applications, users are interested in having a magnetic field as homogeneous as possible within the experiment zone. The typical homogeneity $h = \frac{\max_{\mathbf{x}\in\Omega_{mgn}} \mathbf{B}_z(\mathbf{x})}{\min_{\mathbf{x}\in\Omega_{mgn}} \mathbf{B}_z(\mathbf{x})} - 1$ of our magnets is about 10^{-3} in a 1 cm^3 whereas users are looking for 10^{-6}.

So at the LNCMI, a focus is made in reaching a homogeneous field in the region of interest. Following an idea of C. Trophime and A. Janka, we want to modify the shape of the helical cut by applying a geometric transformation, and perform an optimization of this transformation to have the best homogeneity possible.

The shape of the helix is modified by using an angular torsion as described in Figure 10.1. The angle of the torsion is controlled by a Bezier curve $\alpha(z)$. The control points p_k of the curve are the parameters of the optimization $\boldsymbol{\mu}$.

$$\tilde{\alpha}_{\boldsymbol{\mu}}(t) = \sum_{k=0}^{n} B_k^n(t) p_k = \sum_{k=0}^{n} C_n^k t^k (1-t)^{n-k} p_k \tag{10.1}$$

In fact we need to rescale z so that the helices are comprised between 0 and 1 in α, and

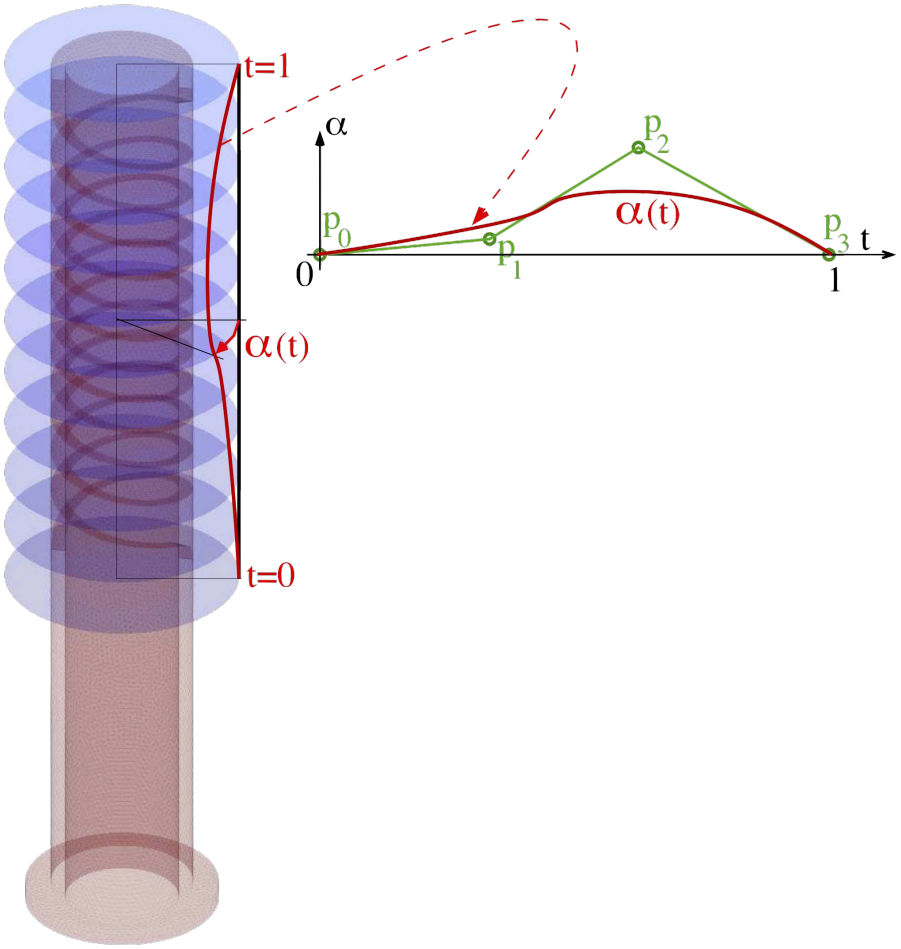


Figure 10.1 – Description of the geometric transformation

that $\alpha = 0$ if we are outside the height of the helices:

$$\alpha_{\boldsymbol{\mu}}(z) = \begin{cases} 0 & \text{if } z < z_b \text{ or } z > z_t \\ \tilde{\alpha_{\boldsymbol{\mu}}}\left(\frac{z-z_b}{z_t-z_b}\right) & \text{else} \end{cases} \tag{10.2}$$

To assure continuity, we impose $p_0 = p_n = 0$ in the control points, so that $\boldsymbol{\mu} = (p_1, \ldots, p_{n-1})$.

The geometric transformation is then written as:

$$\phi_{\boldsymbol{\mu}} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x\cos(\alpha_{\boldsymbol{\mu}}(z)) + y\sin(\alpha_{\boldsymbol{\mu}}(z)) \\ -x\sin(\alpha_{\boldsymbol{\mu}}(z)) + y\cos(\alpha_{\boldsymbol{\mu}}(z)) \\ z \end{pmatrix} \tag{10.3}$$

10.2 Problem

So the weak formulation reads:

$$\begin{aligned}\int_{\phi_\mu(\Omega)} \sigma\nabla V\cdot\nabla\varphi_V + \int_{\phi_\mu(\Omega)} k\nabla T\cdot\nabla\varphi_T + \int_{\phi_\mu(\Gamma_C)} hT\varphi_T \\ - \int_{\phi_\mu(\Gamma_I\cup\Gamma_O)} \sigma\left(\left(\nabla V\cdot\mathbf{n}\right)\varphi_V + \left(\nabla\varphi_V\cdot\mathbf{n}\right)V\right) - \int_{\phi_\mu(\Gamma_I\cup\Gamma_O)} \sigma\frac{\gamma}{h_F}V\varphi_V \\ = \int_{\phi_\mu(\Omega)} \left(\sigma\nabla V\cdot\nabla V\right)\varphi_T + \int_{\phi_\mu(\Gamma_C)} hT_W\varphi_T - \int_{\phi_\mu(\Gamma_O)} \sigma V_D\left(\nabla\varphi_V\cdot\mathbf{n}\right) + \int_{\phi_\mu(\Gamma_O)} \sigma V_D\frac{\gamma}{h_F}\varphi_V\end{aligned}$$

To be able to do the computation on a reference mesh, we need the Jacobian and its inverse. Since α depends on z, we first need to define its derivative:

$$\frac{\partial\alpha_{\boldsymbol{\mu}}}{\partial z}(z) = \frac{n}{z_t - z_b}\sum_{k=0}^{n-1}(p_{k+1}-p_k)B_k^{n-1}\left(\frac{z-z_b}{z_t-z_b}\right) \tag{10.4}$$

For the sake of readability, we will note $\alpha'_{\boldsymbol{\mu}}(z) = \frac{\partial\alpha_{\boldsymbol{\mu}}}{\partial z}(z)$.

And we have:

$$J_{\phi_\mu} = \begin{pmatrix} \cos(\alpha_{\boldsymbol{\mu}}(z)) & \sin(\alpha_{\boldsymbol{\mu}}(z)) & \alpha'_{\boldsymbol{\mu}}(z)(-x\sin(\alpha_{\boldsymbol{\mu}}(z)) + y\cos(\alpha_{\boldsymbol{\mu}}(z))) \\ -\sin(\alpha_{\boldsymbol{\mu}}(z)) & \cos(\alpha_{\boldsymbol{\mu}}(z)) & \alpha'_{\boldsymbol{\mu}}(z)(-x\cos(\alpha_{\boldsymbol{\mu}}(z)) - y\sin(\alpha_{\boldsymbol{\mu}}(z))) \\ 0 & 0 & 1 \end{pmatrix} \tag{10.5}$$

$$J_{\phi_\mu}^{-1} = \begin{pmatrix} \cos(\alpha_{\boldsymbol{\mu}}(z)) & -\sin(\alpha_{\boldsymbol{\mu}}(z)) & -y\alpha'_{\boldsymbol{\mu}}(z) \\ \sin(\alpha_{\boldsymbol{\mu}}(z)) & \cos(\alpha_{\boldsymbol{\mu}}(z)) & x\alpha'_{\boldsymbol{\mu}}(z) \\ 0 & 0 & 1 \end{pmatrix} \tag{10.6}$$

$$|J_{\phi_\mu}| = 1 \tag{10.7}$$

With this geometric transformation, the weak formulation writes as:

$$\begin{aligned}\int_\Omega \sigma\left(\nabla V\cdot J_{\phi_\mu}^{-1}\right)\cdot\left(\nabla\varphi_V\cdot J_{\phi_\mu}^{-1}\right)\left|J_{\phi_\mu}\right| + \int_\Omega k\left(\nabla T\cdot J_{\phi_\mu}^{-1}\right)\cdot\left(\nabla\varphi_T\cdot J_{\phi_\mu}^{-1}\right)\left|J_{\phi_\mu}\right| + \int_{\Gamma_C} hT\varphi_T\left|J_{\phi_\mu}\right| \\ - \int_{\Gamma_I\cup\Gamma_O} \sigma\left(\left(\nabla V\cdot J_{\phi_\mu}^{-1}\cdot\mathbf{n}\right)\varphi_V + \left(\nabla\varphi_V\cdot J_{\phi_\mu}^{-1}\cdot\mathbf{n}\right)V\right)\left|J_{\phi_\mu}\right| - \int_{\Gamma_I\cup\Gamma_O}\sigma\frac{\gamma}{h_F}V\varphi_V\left|J_{\phi_\mu}\right| \\ = \int_\Omega \sigma\left(\nabla V\cdot J_{\phi_\mu}^{-1}\right)\cdot\left(\nabla V\cdot J_{\phi_\mu}^{-1}\right)\left|J_{\phi_\mu}\right| + \int_{\Gamma_C} hT_W\varphi_T\left|J_{\phi_\mu}\right| \\ - \int_{\Gamma_O}\sigma V_D\left(\nabla\varphi_V\cdot J_{\phi_\mu}^{-1}\cdot\mathbf{n}\right)\left|J_{\phi_\mu}\right| + \int_{\Gamma_O}\sigma V_D\frac{\gamma}{h_F}\varphi_V\left|J_{\phi_\mu}\right|\end{aligned} \tag{10.8}$$

Using EIM for the right hand side $\sigma\left(\nabla V\cdot J_{\phi_\mu}^{-1}\right)\cdot\left(\nabla V\cdot J_{\phi_\mu}^{-1}\right)$ and for $J_{\phi_\mu}^{-1}$ such that:

$$\sigma\left(\nabla V\cdot J_{\phi_\mu}^{-1}\right)\cdot\left(\nabla V\cdot J_{\phi_\mu}^{-1}\right) \approx \sum_{m=0}^{M_f}\beta_m^f(\boldsymbol{\mu})q_m^f \qquad J_{\phi_\mu}^{-1} \approx \sum_{m=0}^{M_J}\beta_m^J(\boldsymbol{\mu})Q_m \tag{10.9}$$

we get the following affine decomposition:

$$\begin{aligned}
&\sigma \sum_{m,l=0}^{M_J} \beta_m^J(\boldsymbol{\mu})\beta_l^J(\boldsymbol{\mu}) \int_\Omega (\nabla V \cdot Q_m) \cdot (\nabla \varphi_V \cdot Q_l) \left|J_{\phi_\mu}\right| + k \sum_{m,l=0}^{M_J} \beta_m^J(\boldsymbol{\mu})\beta_l^J(\boldsymbol{\mu}) \int_\Omega (\nabla T \cdot Q_m) \cdot (\nabla \varphi_T \cdot Q_l) \left|J_{\phi_\mu}\right| \\
&\quad - \sigma \sum_{m=0}^{M_J} \beta_m^J(\boldsymbol{\mu}) \int_{\Gamma_I \cup \Gamma_O} \left((\nabla V \cdot Q_m \cdot \mathbf{n})\, \varphi_V + (\nabla \varphi_V \cdot Q_m \cdot \mathbf{n})\, V\right) \left|J_{\phi_\mu}\right| - \int_{\Gamma_I \cup \Gamma_O} \sigma \frac{\gamma}{h_F} V \varphi_V \left|J_{\phi_\mu}\right| \\
&\quad + h \int_{\Gamma_C} T \varphi_T \left|J_{\phi_\mu}\right| \\
&= \sum_{k=0}^{M_f} \sum_{m,l=0}^{M_J} \beta_k^f(\boldsymbol{\mu})\beta_m^J(\boldsymbol{\mu})\beta_l^J(\boldsymbol{\mu}) \int_\Omega q_k^f\, (\nabla V \cdot Q_m) \cdot (\nabla V \cdot Q_l) \left|J_{\phi_\mu}\right| + h T_W \int_{\Gamma_C} \varphi_T \left|J_{\phi_\mu}\right| \\
&\quad - \sigma V_D \sum_{m=0}^{M_J} \beta_m^J(\boldsymbol{\mu}) \int_{\Gamma_O} (\nabla \varphi_V \cdot Q_m \cdot \mathbf{n}) \left|J_{\phi_\mu}\right| + \sigma V_D \int_{\Gamma_O} \frac{\gamma}{h_F} \varphi_V \left|J_{\phi_\mu}\right|
\end{aligned}$$

Another simpler way is to use the discrete EIM and its matrix version MDEIM, see section 3.2.2 to write the linear and bilinear form (10.8) directly respectively as:

$$a(u, v; \boldsymbol{\mu}) \approx \sum_{m=0}^{M_A} \beta_m^A(\boldsymbol{\mu}) a_m \qquad f(v; \boldsymbol{\mu}) \approx \sum_{m=0}^{M_F} \beta_m^F(\boldsymbol{\mu}) f_m \tag{10.10}$$

This reduces the number of terms in the affine decomposition, and in practice, it is able to find the exact decomposition of the matrix and vector.

Once we have the electric potential, we want to use the method seen in 5.2.2.2 to determine the magnetic field in the region of interest, a sphere of radius 6mm in the middle of the magnet.

$$\begin{aligned}
\mathbf{B}(\mathbf{x}, \boldsymbol{\mu}) &= \frac{\mu_0}{4\pi} \int_{\phi_\mu(\Omega_C)} \frac{\mathbf{j} \times (\mathbf{x} - \mathbf{r})}{|\mathbf{x} - \mathbf{r}|^3} d\mathbf{r} && \forall \mathbf{x} \in \Omega_{mgn} \\
&= \frac{\mu_0}{4\pi} \int_{\Omega_C} \frac{-\sigma \nabla V \cdot J_{\phi_\mu}^{-1} \times (\phi_{\boldsymbol{\mu}}(\mathbf{x}) - \mathbf{r})}{|\phi_{\boldsymbol{\mu}}(\mathbf{x}) - \mathbf{r}|^3} \left|J_{\phi_\mu}\right| d\mathbf{r}
\end{aligned}$$

As explained in 5.2.2.2 we need to use the discrete version of EIM to approach this integral and as seen in 3.2.4, we can use the empirical quadrature method to avoid the need of computing the integral respectively for every point in the region of interest and for every point in the magnet.

The homogeneity is defined as:

$$h_B(\boldsymbol{\mu}) = \frac{\max_{\mathbf{x} \in \Omega_{mgn}} \mathbf{B}_z(\mathbf{x}, \boldsymbol{\mu})}{\min_{\mathbf{x} \in \Omega_{mgn}} \mathbf{B}_z(\mathbf{x}, \boldsymbol{\mu})} - 1 \tag{10.11}$$

And the problem is to find the parameters $\boldsymbol{\mu}$ such that the homogeneity is minimal:

$$\tilde{\boldsymbol{\mu}} = \arg\min_{\boldsymbol{\mu} \in \mathbb{P}} h_B(\boldsymbol{\mu}) \tag{10.12}$$

10.3 Results

We will focus on a magnet with 4 helices, and a sphere box at its center. The two innermost helices will be optimized to maximize the homogeneity, and the two outermost helices will be used to provide a background magnetic field, they will not be optimized. The original geometry, clipped for visualization purposes, is presented in figure 10.2.

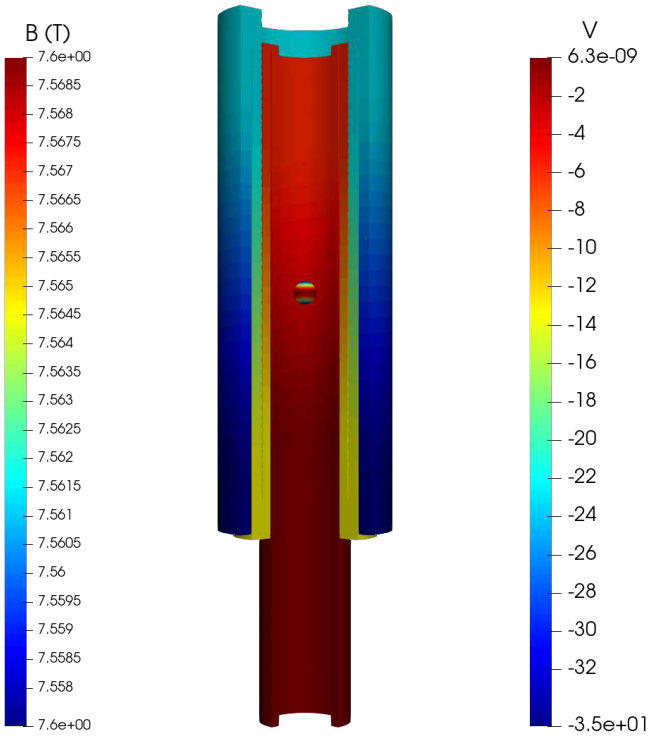


Figure 10.2 – Original geometry

We performed the optimization with the software NLopt [Johnson, Steven G.,]. NLopt is "a free/open-source library for nonlinear optimization, providing a common interface for a number of different free optimization routines available online as well as original implementations of various other algorithms." The difficulty for us is that we do not know the gradient of our objective function, so we have to use derivative free algorithms, which are less effective than gradient base algorithms. On a simplified problem, we computed the homogeneity for parameters between the initial shape and the optimized one. We used a linear combination of the parameters $\boldsymbol{\mu}$, with p_0 the initial parameter and p_1 the optimized parameter, we used $\boldsymbol{\mu} = p_0 + \frac{k}{50}(p_1 - p_0)$. The figure 10.3 shows the continuity of the homogeneity and it seems to be convex.
The other variant of optimization algorithm is the global/local class. A global algorithm will try to find the maximum or minimum over a given set of parameters, whereas a local algorithm will focus on local optimum.

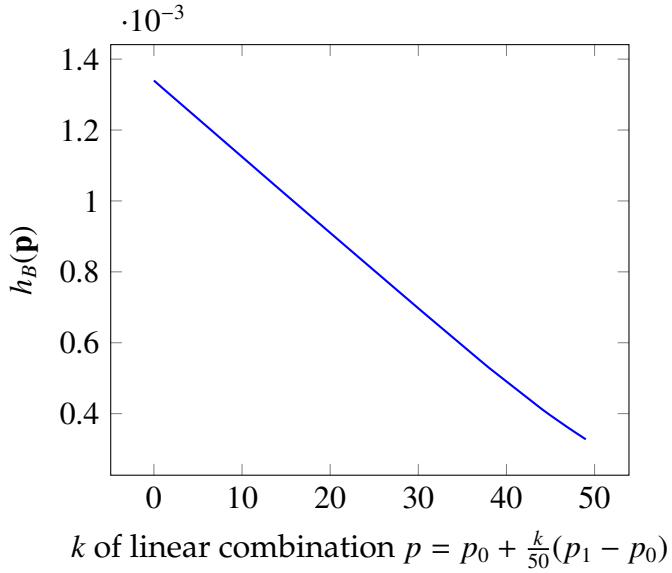
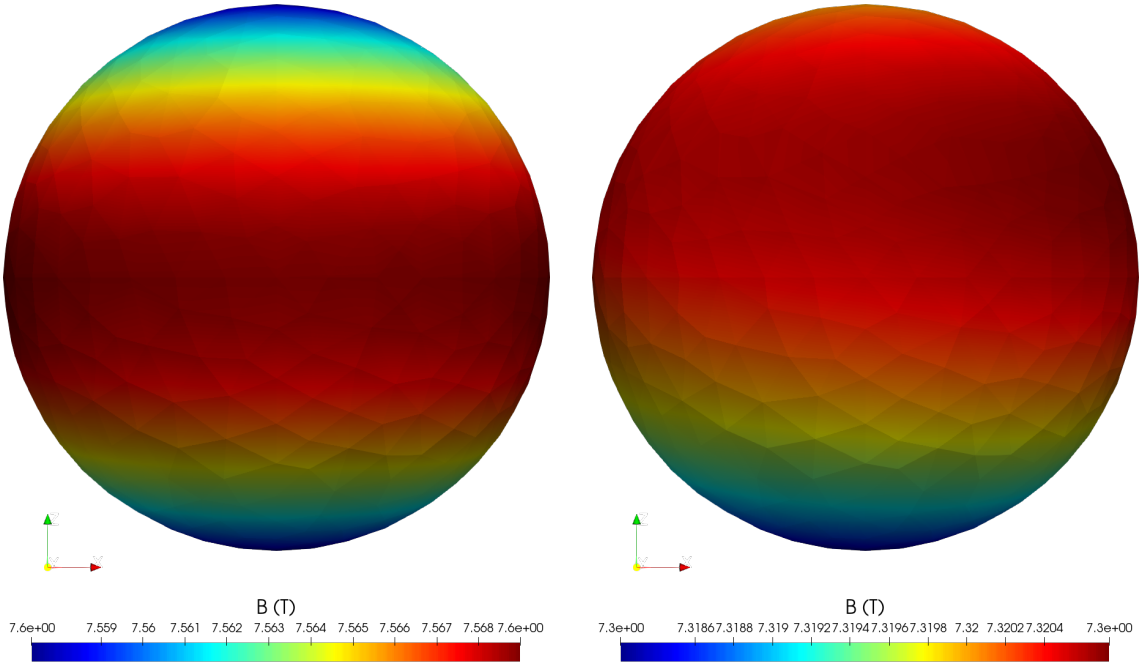


Figure 10.3 – Homogeneity between initial shape (p_0) and optimal shape (p_1)

In order to take advantage of the local algorithm, we need to have an idea of where to search for the optimum. Thus, we first use a global algorithm to find the global parameter space and then we use the global optimum as starting point for the local algorithm. The global algorithm used is DIRECT [JONES et al.,], which search the domain by systematically dividing it into smaller and smaller hyperrectangles in a deterministic way. For the local optimization, we use COBYLA [Powell, 1994], "it constructs successive linear approximations of the objective function and constraints via a simplex of n+1 points (in n dimensions), and optimizes these approximations in a trust region at each step".

Because it is unlikely that the global optimization reach the required tolerance, either for the objective function or the parameters, we need to limit the number of iterations possible. We chose to set this limit to 1000 iterations, since from our experimentations, the subdomain found at this point was enough to start the local algorithm, which took 150 iterations to find the optimum.
Due to the high number of dofs in the magnet, using the Empirical Quadrature Method was not possible, the number of constraints of the linear program exceeded the limit of GLPK. We are currently looking to interface other software which would be able to solve such large linear program, but the following results were obtained without the use of this method, thus by computing the magnetic field at each interpolation point on the magnet with all the quadrature points.
Each finite element evaluation of the homogeneity, including solving the electric problem and computing the magnetic field, took approximately 120s, whereas the reduced evaluation took 10s. Performing the 1150 evaluations necessary to the optimization would have taken more than 38 hours, whereas with the reduced algorithm, it took only 3 hours.

The figures 10.4a and 10.4b show respectively the original z component of the magnetic field in the sphere and the optimized one.



(a) Initial magnetic fied in the sphere (b) Optimised magnetic field in the sphere

Figure 10.4 – Magnetic field in the sphere

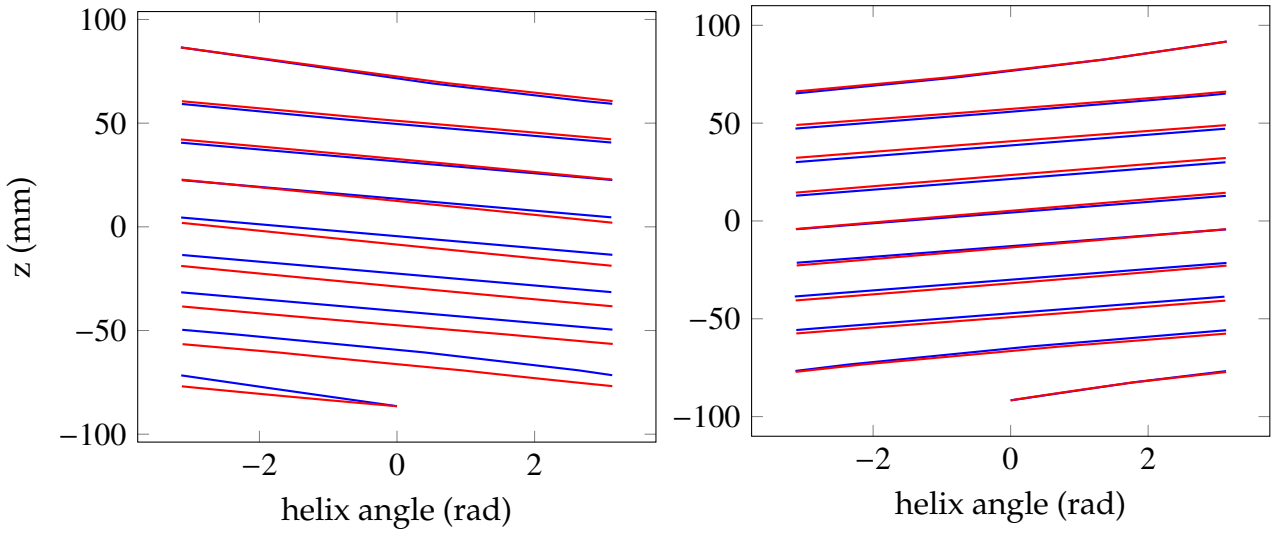
The figure 10.5 shows the differences between the initial (in blue) and the optimal (in red) cuts of the helices.

And the table 10.6 summarizes the initial and optimized quantities. We can see that the loss of magnetic field intensity is very limited, less than 0.001%, but the minimum intensity is much closer to the maximum, leading to an improvement of the homogeneity by a factor 26.

	initial	optimal
$\min \lvert\mathbf{B}_z\rvert$	7.31521	7.33091
$\max \lvert\mathbf{B}_z\rvert$	7.33913	7.33181
$\frac{\max \lvert\mathbf{B}_z\rvert}{\min \lvert\mathbf{B}_z\rvert} - 1$	3.23698×10^{-3}	1.22054×10^{-4}

Figure 10.6 – Initial and optimal homogeneity

In this chapter, we showed an application using a wide range of methods in order to make usable a procedure of geometrical optimization in the context of a high field magnet. We use the reduced basis method in combination with DEIM and EQM to decrease the time needed for the necessary evaluation of the minimum and maximum of the magnetic field. Unfortunately, to scale the method up to such problems, we still need to find a way to solve the linear program for so many constraints, the use of other software than GLPK is investigated. But even without EQM, we can reduce greatly the time needed for the evaluation, and reach interesting results, which could be even



(a) Cuts of the first helix

(b) Cuts of the second helix

Figure 10.5 – Initial (blue) and optimal (red) cuts

improved by using more parameters or more helices. Those results can be used by the LNCMI to enhance the homogeneity of the magnets produced.

Conclusion and Outlook

Conclusion

This thesis built on the work already done in the HifiMagnet project by C. Daversin, aiming to ease the maintainability of the existing models and to have more precise and effective models. The effort was made in order to help the LNCMI to use those methods in their workflow to build better magnets, or better understand their inner workings.

The first part of the thesis consisted of rewriting the models to use the latest feature of `Feel++` and the use of Json files to ease the configuration of the models.
We then implemented the HDG method in `Feel++` and created toolboxes for the thermoelectric and elasticity problems. This method permits to control the flux fields directly, such as the current density, the magnetic field, or the stress, thus having better physical properties, such as the conservation of the current. In particular, we develop the Integral Boundary Conditions, allowing us to be as close as possible to the actual experimental conditions at the LNCMI, by imposing the current intensity instead of the difference of potential. In order to be able to scale up to real geometries, we needed to implement a special partitioning strategy, and the static condensation, which would decrease greatly the computational cost of the method.

Next, we wanted to improve the efficiency of the method by using a order reduction method. We chose the Reduced Basis Method, since it is well adapted to problems discretized with the Finite Element Method and provide an efficient offline/online strategy. We had to create the reduced model for the thermoelectric problem and implement the order reduction for Biot-Savart. We also helped to the implementation of the Discrete Empirical Interpolation Method in the reduced order framework of `Feel++`, which is very convenient when dealing with complex geometrical parameters or problem with many EIMs. To reduce the cost of the computation of Biot-Savart, the Empirical Quadrature Method was also implemented and integrated in the reduced order framework of `Feel++`. If the preliminary results on this method are encouraging, we still need to find a solution to scale the method up for larger problems. All of those methods have been combined to have near real time computation of the magnetic field in a high field magnet.

Finally, we used the methods at our disposal in two applications that can be used by the LNCMI to improve the quality of the magnets produced or ease their operations.
We confronted our models to experimental data from the LNCMI, searching to identify

the cooling parameters corresponding to the experiment. We found that the correlation used for the heat exchange coefficient is important, leading to an error of less than 4% for both the CG and HDG models. We also accurately retrieved the field factor and the resistance of the magnet, used to control the intensity needed and to detect anomalies during the experiment. The use of the order reduction to optimize further those parameters were complicated by the size of the problem and the number of parameters, but alternative strategies are implemented to solve this issue.
The second application consisted in optimizing the cuttings of the inner helices in order to have a better homogeneity inside the magnetic box. It implied the use of geometrical parameters, leading to the use of all methods presented previously (RB, DEIM, EQM, SER), to reduce the cost of the optimization. The results motivate the use of this method with more parameters in order to achieve a significant improvement of the homogeneity in the magnets, providing a lead for their construction by the LNCMI.

Outlook

The HifiMagnet project will continue to improve and develop new methods to help the LNCMI by using the simulation.

First, from a mathematical point of view, we can add the HDG model for the Maxwell equations, enabling the computation of all the physics with HDG. HDG could also be used inside the reduced basis framework, allowing the same control over the reduced flux fields than with the full model. And the Maxwell and Elasticity problems could also be reduced using the RB method, which would open other applications needing real time computations of the magnetic field inside the materials of the magnet and/or the deformation of the magnet.

The modelization of the problem can also be improved, first by using dynamic models, in order to see the evolution of the different quantities over time, an internship is currently working on the subject. Then, we could fully modelize the hydraulics involved in the cooling of the magnet, it would necessitate solving a Navier-Stokes equation for the water flowing through the magnet. And to be even more precise, we should use an ALE map to compute the thermoelectric and magnetostatic problems, in order to take into account the deformation of the magnet.

Finally, the methods developed during the HifiMagnet should be more integrated for the LNCMI engineers, who should be able to launch a simulation in the cloud from their desktop station, using the work done in the MSO4SC European project. The model order reduction methods should also be available from the MOR_DICUS library, a FUI project aiming to industrialize the use of such methods.
And of course, the project should continue to develop other applications of interest for the LNCMI.

Appendices

Appendix A

Hydromorpho

This part details a side project realised during a CEMRACS (2015). This event consist in a 6 modeling weeks where Phd students from several horizons gather to work on one project. We propose to couple the Exner equation with the Stokes equations to model the bedload sediment in geophysical flows. This work is a preliminary study to directly model the hydrodynamic flow by the unsteady Stokes equation instead of the classical shallow water equation. We focus in this proceeding on the coupling applying fluid structure interaction approach to morphodynamical behavior. In other words, we follow the approach of fluid interaction models replacing the structure equation by the Exner equation. The aim of this work is to validate the proposed procedure. These equations are solved by finite element method using the library `Feel++`.

ESAIM: PROCEEDINGS AND SURVEYS, December 2016, Vol. 55, p. 23-40
Emmanuel FRÉNOD, Emmanuel MAITRE, Antoine ROUSSEAU, Stéphanie SALMON and Marcela SZOPOS Editors

HYDROMORPHO: A COUPLED MODEL FOR UNSTEADY STOKES/EXNER EQUATIONS AND NUMERICAL RESULTS WITH FEEL++ LIBRARY *

Nora Aïssiouene[1], Tarik Amtout[2], Matthieu Brachet[3], Emmanuel Frenod[4], Romain Hild[5], Christophe Prud'homme[6], Antoine Rousseau[7] and Stephanie Salmon[8]

Abstract. We propose to couple the Exner equation with the Stokes equations to model the bedload sediment in geophysical flows . This work is a preliminary study to directly model the hydrodynamic flow by the unsteady Stokes equation instead of the classical shallow water equation. We focus in this proceeding on the coupling applying fluid structure interaction approach to morphodynamical behavior. In other words, we follow the approach of fluid interaction models replacing the structure equation by the Exner equation. The aim of this work is to validate the proposed procedure. These equations are solved by finite element method using the library FEEL++.

Résumé. Nous proposons de coupler l'équation d'Exner avec les équations de Stokes afin de modéliser le transport des sédiments par charriage. Ce travail est une étude préliminaire pour modéliser le flux hydrodynamique par les équations instationnaires de Stokes à la place du choix classique des équations de Saint-Venant. Ici, nous nous concentrons sur l'utilisation d'une approche interaction fluide structure pour le couplage, c'est-à-dire remplacer l'équation de structure par l'équation d'Exner. Le but de ce travail est de valider la procédure proposée. Ces équations sont résolues par la méthode des élements finis en utilisant la bibliothèque FEEL++.

Introduction

Many hydrodynamic studies have been done to understand and predict the dynamics of sediments at the bottom of flows which is a significant and complex process for many geophysical situations. Morphodynamics modelling is a broad subject whose principles can be found in several references [34], [35]. We can distinguish two types of sediment transport, the suspended load and the bedload. In this proceeding, we focus on the bedload transport and its impact on the hydrodynamics. The difficulty remains in the necessity to couple the

* *We thank the GDR EGRIN for its financial support and the CIRM for its warm welcome during the CEMRACS.*

[1] Inria - CEREMA - Sorbonne Universités, UPMC Univ Paris 06 - CNRS, UMR 7598, Laboratoire Jacques-Louis Lions
[2] Faculté des Sciences et Techniques Tanger, Maroc
[3] Institut Elie Cartan de Lorraine, Université de Lorraine, Site de Metz, Bât. A Ile du Saulcy, F-57045 Metz Cedex 1
[4] Université Bretagne-Sud, LMBA UMR 6205, Vannes
[5] Institut de Recherche Mathématique Avancée, Strasbourg
[6] Institut de Recherche Mathématique Avancée, Strasbourg
[7] Inria and Institut Montpelliérain Alexander Grothendieck, Team LEMON, Bat 5 - CC05 017, 860 rue Saint-Priest, 34095 Montpellier Cedex 5, France
[8] Laboratoire de Mathématiques de Reims, Fédération ARC CNRS 3399, Université de Reims Champagne-Ardenne

Article published online by EDP Sciences and available at http://www.esaim-proc.org or http://dx.doi.org/10.1051/proc/201655023

sediment transport models with the hydrodynamic models, and then to develop a robust and stable numerical method.

On the one hand, the sediment transport is usually modeled by the classical Exner equation [37] and several laws of transport have been proposed (see [21] to have details on some classical laws) by physical arguments or closure relations. On the other hand, models as shallow water equations are used to model the hydrodynamics, and recently in [12, 21] a model derived from the Navier-Stokes equations that has an energy balance.

Concerning the numerical methods that have been established for these models, the main numerical schemes are developed for the hyperbolic systems with source terms for the hydrodynamic flow (see [9], [25]). Therefore, finite volume schemes are applied for the shallow water system [1, 2, 23, 38]. The problem lies in the coupling of the numerical schemes. Indeed, in the shallow water models, the topography is a source term and the Exner equation gives the evolution of the bottom in terms of the fluid velocity. Then, two strategies are distinguished, the splitting one and the non-splitting one (see [4]). The splitting methods are easier to implement but generate instabilities in specific situations (see [14]). On the contrary, more complicated models, for instance involving relaxation, have to be used to take into consideration the fully coupled model [3, 29].

Notice that the shallow water model is based on a hydrostatic assumption. It is deduced from the Navier Stokes equations, neglecting the vertical acceleration (see [23]). Many other free surface models have been developed to take into account non-hydrostatic effects with vertically averaged models: see [8, 10, 11, 18, 32]. Contrariwise, for this study we choose to conserve the z coordinate in our model, which raises the question of time-depending domain when the bathymetry changes with time: this coupling between fluid motion (including vertical effects) and domain evolution is at the core of this paper.

This objective being stated, we start with the simplest possible model, a 2D $(x - z)$ Stokes equation. We couple this equation for the fluid with the Exner model since our computational domain moves as times goes by. We choose to use the Grass law for the bedload formula (see Equation (10)) which is one possible law among others. As for the time coupling between hydrodynamical and morphological processes, we choose to use a monolithic scheme rather than a splitting method: such refinements (that can prove to be very important, see [14]) are beyond the scope of our work.

Let us now focus on the main feature of this work: the use of fluid-structure interaction techniques (see [13, 26]) for the coupling between Stokes and Exner equations. From the numerical viewpoint, we decided to use finite elements and the open software Feel++ [39–41] that are well adapted to fluid-structure interaction (ALE implementation, see [26]) and parallelization for large 3D computations.

The article is organized as follows, the first part is devoted to the description of the fluid model and the sediment model at the bottom. In a second part, it is explained why a method like the ALE is necessary to couple the models. The third part establishes a complete ALE formulation of the Stokes-Exner model. Then, a variational formulation is given with the different boundary conditions that we explore. Finally, some numerical results are presented to evaluate the model and the method used to solve the problem.

1. The model

In this part, we introduce various equations for our coupled system. Section 1.1 is devoted to the unsteady Stokes equations (dimension 2, $x - z$) that we supplement with appropriate boundary conditions. Section 1.2 is dedicated to the bottom boundary condition, located at the (moving) boundary where the fluid model is coupled with the Exner equation for bedload. Before recalling the complete coupled system in section 1.4, we present in Section 1.3 the ALE implementation of our model.

We start with a model domain $\Omega(t)$ and a specific boundary to represent the topography. We consider the domain as a moving domain depending on the bottom. Let us introduce the domain with the following

definitions:

$$\Omega(t) = \{(x,z) \in \mathbb{R}^2 \quad | \quad 0 \leq x \leq l, \quad b_z(x,t) \leq z \leq 1\} \tag{1}$$

where $l > 0$ is the cavity length and $b_z(x,t)$ is the bottom topography in x at time t. We also denote by $\Gamma(t) = \Gamma_{in}(t) \cup \Gamma_{out}(t) \cup \Gamma_s \cup \Gamma_b(t)$ the boundaries (see Figure 1):

- $\Gamma_{in}(t) = \{0\} \times [b_z(0,t), 1]$,
- $\Gamma_{out}(t) = \{l\} \times [b_z(l,t), 1]$,
- $\Gamma_s = [0,l] \times \{1\}$,
- $\Gamma_b(t) = \{(x,z) \in \mathbb{R}^2 \text{ s.t. } z = b_z(x,t), x \in [0,l]\}$.

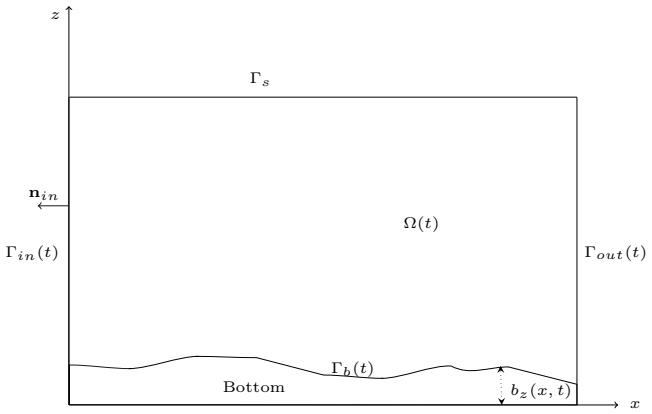


FIGURE 1. Definition of the domain

The coupled model leads to solving the non-steady Stokes problem in the fluid domain $\Omega(t)$ and the Exner equation to give the boundary $\Gamma_b(t)$. The issue is to model the fluid process in interaction with the sediment transport at the bottom. In the following sections, we describe the equations chosen for the fluid in the domain $\Omega(t)$ with usual boundary conditions for the boundary Γ. Then we propose to use Exner equation to make the boundary Γ_b move.

1.1. **Hydrodynamical Model**

We consider the unsteady Stokes problem on the domain $\Omega(t)$

$$\rho \frac{\partial \mathbf{u}}{\partial t} - \mu \triangle \mathbf{u} + \nabla p = 0 \text{ on } \Omega(t), \tag{2}$$

$$\text{div}\,(\mathbf{u}) = 0 \text{ on } \Omega(t), \tag{3}$$

where $\mathbf{u} = (u,w)^T$ is the velocity of the fluid, p is the pressure, $\mu > 0$ is the dynamic viscosity and ρ is the density. From now on, we will use $\rho = 1$. This problem is completed by the boundary conditions detailed hereafter.

A crucial issue is to have judicious boundary conditions at the interface between the fluid and the topography. The physical behavior of the sediment transport studied here implies an impermeability boundary condition, then a constraint on the normal component of the velocity has to be done. Concerning the other boundaries, we will consider a model test case on which we simulate a flow on the pseudo free surface (Γ_s in our case). Then,

we impose the velocity on the surface boundary Γ_s by a Dirichlet condition and free boundary conditions, using Neumann boundary conditions for the velocity, at the inlet and outlet

$$\mathbf{u} = \mathbf{g}_1 \text{ on } \Gamma_s, \tag{4}$$

$$\boldsymbol{\sigma}\mathbf{n} = \mu\frac{\partial \mathbf{u}}{\partial \mathbf{n}} - p\mathbf{n} = \mathbf{g}_2 \text{ on } \Gamma_{in}(t) \cup \Gamma_{out}(t), \tag{5}$$

$$\mathbf{u} \cdot \mathbf{n} = 0 \quad \text{on } \Gamma_b(t), \tag{6}$$

$$\boldsymbol{\sigma}\mathbf{n} \cdot \boldsymbol{\tau} = \mu\frac{\partial \mathbf{u}}{\partial \mathbf{n}} \cdot \boldsymbol{\tau} = g_3 \text{ on } \Gamma_b(t), \tag{7}$$

where $\boldsymbol{\sigma}$ is the stress tensor defined by:

$$\boldsymbol{\sigma} = (\mu\nabla\mathbf{u} - p\mathbb{I}d). \tag{8}$$

The condition (4) imposes the force by using Dirichlet condition.
The condition (5) lets free the velocity at the inlet and at the outlet.
The condition (6) imposes the normal component of the velocity to be null at the bottom. It is the condition of impermeability of the domain.
The condition (7) lets free the tangential component of the velocity at the bottom. It is needed to have a displacement of the bottom.

1.2. **Morphodynamics model**

The sediment dynamics is based on the formulation of a sediment continuity equation stating that the time variation of the sediment layer in a certain volume is due to the net variation of the solid transport through the boundaries of the volume. The mathematical expression of such law is known as the Exner equation [33] presented in this form:

$$\frac{\partial b_z}{\partial t} + \xi\frac{\partial Q}{\partial x} = 0 \quad \forall x \in [0, l], \forall t \in [0, T], \tag{9}$$

where $b_z(x, t)$ is the bed elevation, ξ is defined by $(1 - p)^{-1}$ where p is the material porosity and Q denotes the solid transport discharge along the x coordinate influenced by the velocity $\mathbf{u}$. The formulation of the bedload discharge Q can be based on deterministic laws ([5], [19], [42]) or in probabilistic methods ([20], [30]), often supported by experimentation. Grass [27] discussed one of the most basic sediment transport laws that can be written in one dimension as:

$$Q = a|\mathbf{u}|^{3/2}, \tag{10}$$

where $0 < a < 1$ is an empirical parameter depending of the type of the sediments, it takes into account the effects due to the grain size and the kinematic viscosity . For the problem studied in this work, the velocity taken into consideration in the Grass formula is reduced to the tangential part $\mathbf{u}_\tau$ since we impose an impermeability condition on the interface, see the boundary condition (6).

For the sake of clarity, we will consider the Exner equation under the form:

$$\frac{\partial b_z}{\partial t} + \frac{\partial Q}{\partial x} = 0 \quad \forall x \in [0, l], \forall t \in [0, T], \tag{11}$$

where $Q = \alpha|\mathbf{u}_\tau|^{3/2}$ and $\alpha = \xi a$.

1.3. **Arbitrary Lagrangian Eulerian (ALE) Method**

We now want to couple the two models previously described. The issue is to solve the unsteady Stokes equations with a moving boundary Γ_b. In fluid mechanics, one can enumerate two ways to represent a problem: Lagrangian and Eulerian formulation. On the one hand, Lagrangian formulation is similar to keep track of the location of each fluid particles. The velocity $\mathbf{u}$ and the density ρ depend only on $\mathbf{x_0}$ the initial position of the particles and on t the time. Then, the time derivative of a quantity F is given by the total time derivative :

$$\frac{DF}{Dt}.$$

On the other hand, the main idea of the Eulerian method is to fix a system of coordinates and follow the flux of particles. In this case, the velocity $\mathbf{u}$ and the density ρ depend on $\mathbf{x}$ the position in a global system of coordinates and t the time. Now, for a function F, the time derivative is given by :

$$\frac{\partial F}{\partial t} + \mathbf{u} \cdot \nabla F.$$

The relation between the Eulerian and the Lagrangian time derivatives is:

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla. \tag{12}$$

Then, the idea of the ALE method is to combine the Eulerian and the Lagrangian method in order to take into consideration the boundary displacement at each iteration, which represents the bottom in our case. The goal is to avoid remeshing the domain at each time iteration. This method was first developped for finite difference in [22, 36] and scope to finite element methods in [16, 17] and [6]. In 2004, [13] built a method for great order elements. This has been widely used in Fluid Structure Interaction (FSI) on which it is usual to have a fluid equation like unsteady Stokes or Navier Stokes in the fluid domain and an elasticity equation for the structure. This is used in the simulation of blood flow in arteries for instance (see [13], [26]). In the context of the sediment transport, the bottom plays the role of the structure in the classical methods. Then the idea is to use the analogy of these methods for the coupled Stokes Exner model.
As the goal is to avoid remeshing the domain, the clue is to use a Lagrangian description to describe the bottom displacement and a Eulerian description for the fluid model. First, we define a reference domain on which the topography is described by a Lagrangian description. Secondly, we consider the physical domain $\Omega(t)$ on which the equations of the fluid evolves. Then, it is necessary to define an application able to make the link between the two domains. In the following, we write $\widehat{\cdot}$ all quantities concerning the reference domain. For the sake of clarity, we choose $\widehat{\Omega}$ the rectangular domain $[0, l] \times [0, 1]$ and the following boundaries :

- $\widehat{\Gamma}_{in} = \{0\} \times [0, 1]$,
- $\widehat{\Gamma}_s = [0, l] \times \{1\}$,
- $\widehat{\Gamma}_{out} = \{l\} \times [0, 1]$,
- $\widehat{\Gamma}_b = [0, l] \times \{0\} = \widehat{\gamma}_b \times \{0\}$ where $\widehat{\gamma}_b = [0, l]$.

The relation between the reference domain $\widehat{\Omega}$ and the physical domain $\Omega(t)$, is made by an ALE map (see figure 2), defined by :

$$\mathcal{A}^t : \left\{ \begin{array}{ccc} \widehat{\Omega} & \longrightarrow & \Omega(t) \\ \widehat{\mathbf{x}} & \longmapsto & \mathbf{x}(\widehat{\mathbf{x}}, t) \end{array} \right. . \tag{13}$$

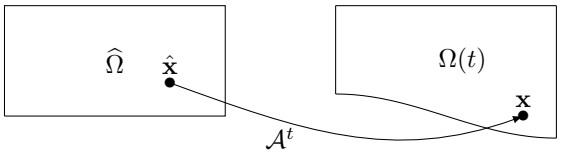


FIGURE 2. The ALE map

Therefore the point $\mathbf{x}(t) \in \Omega(t)$ is obtained by:

$$\mathbf{x}(t) = \mathcal{A}^t(\widehat{\mathbf{x}}) = \widehat{\mathbf{x}} + \widehat{\mathbf{d}}_\delta(\widehat{\mathbf{x}}, t), \tag{14}$$

where $\widehat{\mathbf{d}}_\delta(\widehat{\mathbf{x}}, t)$ is the displacement of $\widehat{\mathbf{x}}$ between $\widehat{\Omega}$ and $\Omega(t)$. Notice that $\mathbf{x}(t)$ is time dependent. Then, we can define the velocity of the mesh:

$$\widehat{\mathbf{w}}(\widehat{\mathbf{x}}, t) = \frac{\partial \mathcal{A}^t}{\partial t}(\widehat{\mathbf{x}}) = \frac{\partial \widehat{\mathbf{d}}_\delta}{\partial t}(\mathbf{x}, t), \tag{15}$$

where $\widehat{\mathbf{w}}(\hat{\mathbf{x}}, t) \in \mathbb{R}^d$ is defined for $\hat{\mathbf{x}} \in \widehat{\Omega} \times \mathbb{R}^+$. To take into consideration the velocity of the mesh into the fluid equation, it is necessary to define it in the fluid domain $\Omega(t)$, namely:

$$\mathbf{w} \quad : \quad (\mathbf{x}, t) \in \Omega(t) \times \mathbb{R}^+ \to \mathbb{R}^d, \tag{16}$$

$$\mathbf{w} \quad = \quad \widehat{\mathbf{w}} \circ \left(\mathcal{A}^t\right)^{-1}. \tag{17}$$

This definition will allow us to rewrite the fluid equation with an Eulerian description, taking into account the displacement of the mesh. The last step of the method leads to determine the equation of the displacement $\widehat{\mathbf{d}}_\delta$ in $\widehat{\Omega}$. In practice, $\mathbf{d}_\delta$ is the solution of a PDE like harmonic or Wislow equation. For the sake of simplicity, we will work with harmonic extension that allows to have a smooth mesh. We often need to transport an equation from $\widehat{\Omega}$ to $\Omega(t)$ and mutually. Let $u : \Omega(t) \times \mathbb{R}^+ \longrightarrow \mathbb{R}^d$, then the corresponding map in $\widehat{\Omega}$ is $\widehat{u} = u \circ \mathcal{A}^t$.
If $\mathcal{D}\mathbf{u}/\mathcal{D}t$ is the time-derivative of u in ALE, we have the following equation:

$$\frac{\mathcal{D}\mathbf{u}}{\mathcal{D}t} = \left.\frac{\partial \mathbf{u}}{\partial t}\right|_x + \mathbf{w} \cdot \nabla \mathbf{u}. \tag{18}$$

- Notice that if $\mathbf{w} = \mathbf{u}$, the mesh is moving with the particles so the description is Lagrangian.
- If $\mathbf{w} = \mathbf{0}$, the mesh does not move and the description is Eulerian.

1.4. **Coupled model**

In this part, we focus on the coupled model. We denote by $\widehat{\mathbf{x}}$ a point in the reference domain $\widehat{\Omega}$ and by $\Omega(t)$ the deformed domain after the transformation. The deformation of the mesh leads to consider the derivative $\mathcal{D}$ defined by (18) which is also called the ALE derivative where the velocity $\mathbf{w}$ is defined as: $\widehat{\mathbf{w}} = \left.\frac{\partial \widehat{\mathbf{d}}_\delta}{\partial t}\right|_{\hat{x}}$ and represents the velocity of the displacement of the mesh, that is to say the velocity of the particle in the referential domain. This allows to write the fluid model with a moving mesh on $\Omega(t)$. Concerning the boundary conditions, we still consider a slip boundary condition at the interface between the topography and the fluid. The complete model is composed of the equation of the fluid in two dimensional domain, the equation of the topography in one dimensional domain and the ALE equation in two dimensional domain. According to (18), the coupled model is written as follows :

1.4.1. *Fluid equation*

The fluid equation is given by :

$$\frac{\mathcal{D}\mathbf{u}}{\mathcal{D}t} - (\mathbf{w}\cdot\nabla)\mathbf{u} - \mu \triangle \mathbf{u} + \nabla p = \mathbf{F} \quad \text{on } \Omega(t), \tag{19}$$

$$\text{div}(\mathbf{u}) = 0 \quad \text{on } \Omega(t), \tag{20}$$

$$+\mathbf{BC}, \tag{21}$$

where the boundary conditions are those of section 1.1. In particular, condition (7) depends on the bottom topography.

1.4.2. *Bottom equation*

We consider the one dimensional domain $\widehat{\gamma}_b = [0, l]$, on which the bottom topography at position $\widehat{x} \in \widehat{\gamma}_b$ and time $t > 0$ is defined by the Exner equation (11):

$$\frac{\partial \hat{b}_z(\hat{x},t)}{\partial t} + \frac{\partial \hat{Q}(\hat{x},t)}{\partial \hat{x}} = 0 \quad \forall \hat{x} \in \widehat{\gamma}_b, t > 0, \tag{22}$$

$$\hat{b}_z(\hat{x},0) = \hat{b}_{z,0}(\hat{x}). \tag{23}$$

Using (10), the sediment law $\hat{Q}$ depends on the fluid velocity and can be written in the reference domain by:

$$\hat{Q}(\hat{x},t) = Q\left(\hat{x} + \hat{b}(\hat{x}), t\right) \quad \forall \hat{x} \in \widehat{\gamma}_b, \tag{24}$$

$$= \alpha u_\tau \left(\hat{x} + \hat{b}(\hat{x})\right)^{3/2}. \tag{25}$$

1.4.3. *Displacement equation*

This displacement needs to be extended in the fluid domain to associate a new ALE map over the mesh. In order to do this, we use a classical harmonic extension (see [13] for more details).

$$-\triangle \hat{\mathbf{d}}_\delta = 0 \quad \text{on } \hat{\Omega}, \tag{26}$$

$$\hat{\mathbf{d}}_\delta = 0 \quad \text{on } \hat{\Gamma}_s, \tag{27}$$

$$\frac{\partial \hat{\mathbf{d}}_\delta}{\partial \mathbf{n}} = 0 \quad \text{on } \hat{\Gamma}_{in} \cup \hat{\Gamma}_{out}, \tag{28}$$

$$\hat{\mathbf{d}}_\delta = (0, \hat{b}_z(\hat{x},t))^T \quad \text{on } \hat{\Gamma}_b. \tag{29}$$

This allows us to have a given displacement defined on $\hat{\Gamma}_b$, to let free the boundaries $\hat{\Gamma}_{in}$ and $\hat{\Gamma}_{out}$, and to fix the boundary $\hat{\Gamma}_s$. The harmonic problem spreads the displacement $\hat{\mathbf{d}}_\delta$ on all the domain.

1.4.4. *Equation for w*

We denote by $\hat{\mathbf{w}}$ the velocity of the displacement

$$\hat{\mathbf{w}}(\widehat{\mathbf{x}},t) = \frac{\partial \hat{\mathbf{d}}(\hat{\mathbf{x}},t)}{\partial t}. \tag{30}$$

Then, using the ALE transformation, we can compute the velocity $\mathbf{w}$ in the domain $\Omega(t)$:

$$\mathbf{w}(\mathbf{x},t) = \hat{\mathbf{w}}\left((\mathcal{A}^t)^{-1}(\mathbf{x}), t\right). \tag{31}$$

2. Variational formulation

This part is devoted to the variational formulation of the problem taking the ALE description into account.

2.1. Variational formulation of the Exner equation

By multiplying the Exner equation and integrating over $\hat{\gamma}_b$, we have the following variational formulation for equations (22)-(23): Taking a test function $\phi \in H^1(\hat{\gamma}_b)$, we have:

$$\frac{d}{dt}\int_{\hat{\gamma}_b} \hat{b}_z(\hat{x},t)\phi(\hat{x})\, d\hat{x} + \int_{\hat{\gamma}_b} \frac{\partial \hat{Q}(\hat{x},t)}{\partial \hat{x}}\phi(\hat{x})\, d\hat{x} = 0, \tag{32}$$

$$\frac{d}{dt}\int_{\hat{\gamma}_b} \hat{b}_z(\hat{x},t)\phi(\hat{x})\, d\hat{x} - \int_{\hat{\gamma}_b} \hat{Q}(\hat{x},t)\phi'(\hat{x})\, d\hat{x} + \Big[\hat{Q}(\hat{x},t)\phi(\hat{x})\Big]_{\hat{\gamma}_b} = 0 \quad \forall \phi \in H^1(\gamma_b). \tag{33}$$

The problem becomes: find $\hat{b}_z$ such that for all $\phi \in H^1(\hat{\gamma}_b)$

$$\frac{d}{dt}\int_{\hat{\gamma}_b} \hat{b}_z(\hat{x},t)\phi(\hat{x})\, d\hat{x} = \int_{\hat{\gamma}_b} \hat{Q}(\hat{x},t)\phi'(\hat{x})\, d\hat{x} - \Big[\hat{Q}(\hat{x},t)\phi(\hat{x})\Big]_{\hat{\gamma}_b}. \tag{34}$$

2.2. Variational formulation of unsteady Stokes Equation

The problem leads to find $\mathbf{u} \in \mathbf{V}$ and $p \in Q$ such that the fluid equation (19) is satisfied. Let $\mathbf{X}$ be the functional set of test functions. Notice that the sets $\mathbf{V}$, $\mathbf{X}$ and W will be defined later. In practice, the sets $\mathbf{V}$ and $\mathbf{X}$ can be different, they depend on the boundary conditions. Multiplying (19) with a test function $\mathbf{v} \in \mathbf{X}$ and (20) with a test function $q \in Q$, and then integrating by part, we get:

$$\int_\Omega \frac{\mathcal{D}\mathbf{u}}{\mathcal{D}t}\cdot\mathbf{v} - \int_\Omega [(\mathbf{w}\cdot\nabla)\mathbf{u}]\cdot\mathbf{v} + \mu\int_\Omega \nabla\mathbf{u} : \nabla\mathbf{v} - \mu\int_\Gamma \frac{\partial\mathbf{u}}{\partial\mathbf{n}}\cdot\mathbf{v} - \int_\Omega p\,\mathrm{div}(\mathbf{v}) + \int_\Gamma p\mathbf{n}\cdot\mathbf{v} = \int_\Omega \mathbf{F}\cdot\mathbf{v}, \tag{35}$$

$$\int_\Omega \mathrm{div}(\mathbf{u})\, q = 0. \tag{36}$$

Then, using the Reynolds transport formula on the first term of (35):

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega \mathbf{u}\cdot\mathbf{v} - \int_\Omega (\nabla\cdot\mathbf{w})\mathbf{u}\cdot\mathbf{v} - \int_\Omega [(\mathbf{w}\cdot\nabla)\mathbf{u}]\cdot\mathbf{v} & \\ + \mu\int_\Omega \nabla\mathbf{u} : \nabla\mathbf{v} - \mu\int_\Gamma \frac{\partial\mathbf{u}}{\partial\mathbf{n}}\cdot\mathbf{v} & \\ - \int_\Omega p\,\mathrm{div}(\mathbf{v}) + \int_\Gamma p\mathbf{n}\cdot\mathbf{v} & = \int_\Omega \mathbf{F}\cdot\mathbf{v}, \end{aligned} \tag{37}$$

$$\int_\Omega \mathrm{div}(\mathbf{u})\, q = 0. \tag{38}$$

We define the following forms:

$$a_1(\mathbf{u},\mathbf{v}) = \int_{\Omega(t)} \mu \nabla \mathbf{u} : \nabla \mathbf{v}\, dx, \qquad \forall \mathbf{u} \in \mathbf{V}, \mathbf{v} \in \mathbf{X}, \tag{39}$$

$$a_2(\mathbf{u},\mathbf{v}) = -\int_\Omega (\nabla \cdot \mathbf{w})\mathbf{u}\cdot\mathbf{v} - \int_\Omega [(\mathbf{w}\cdot\nabla)\mathbf{u}]\cdot\mathbf{v}\, dx, \quad \forall \mathbf{u} \in \mathbf{V}, \mathbf{v} \in \mathbf{X}, \tag{40}$$

$$a(\mathbf{u},\mathbf{v}) = a_1(\mathbf{u},\mathbf{v}) + a_2(\mathbf{u},\mathbf{v}) \tag{41}$$

$$b(\mathbf{u},q) = \int_{\Omega(t)} q\, \mathrm{div}(\mathbf{u})\, dx, \qquad \forall \mathbf{u} \in \mathbf{V}, q \in Q, \tag{42}$$

$$L(\mathbf{v}) = \int_{\Omega(t)} F(t)\cdot\mathbf{v}\, dx, \qquad \forall \mathbf{v} \in \mathbf{X}. \tag{43}$$

We now need to treat the boundary conditions.

$$\mathbf{u} = \mathbf{g}_1 \text{ on } \Gamma_s, \tag{44}$$

$$\boldsymbol{\sigma}\mathbf{n} = \mu\frac{\partial \mathbf{u}}{\partial \mathbf{n}} - p\mathbf{n} = \mathbf{g}_2 \text{ on } \Gamma_{in}(t)\cup\Gamma_{out}(t), \tag{45}$$

$$\mathbf{u}\cdot\mathbf{n} = 0 \quad \text{on } \Gamma_b(t), \tag{46}$$

$$\boldsymbol{\sigma}\mathbf{n}\cdot\boldsymbol{\tau} = \mu\frac{\partial \mathbf{u}}{\partial \mathbf{n}}\cdot\boldsymbol{\tau} = g_3 \text{ on } \Gamma_b(t). \tag{47}$$

2.2.1. *Dirichlet and Neumann boundary conditions*

We impose the Dirichlet condition in a strong way, the condition is embedded directly into the space in which we search the solution. We introduce the spaces:

$$\mathbf{V} = \{\mathbf{u} \in (\mathbf{H}^1(\Omega(t)))^2, \quad \mathbf{u} = \mathbf{g}_1 \text{ on } \Gamma_s\},$$
$$\boldsymbol{X} = \{\mathbf{v} \in (\mathbf{H}^1(\Omega(t)))^2, \quad \mathbf{v} = 0 \text{ on } \Gamma_s\},$$
$$Q = L^2(\Omega(t))$$

The Neumann condition on $\Gamma_{in}\cup\Gamma_{out}$ comes naturally into the formulation. Indeed, the boundary terms can be written with $\mathbf{v} \in \boldsymbol{X}$:

$$\int_{\Gamma_s}\left(p\mathbf{n} - \mu\frac{\partial \mathbf{u}}{\partial \mathbf{n}}\right)\cdot\underbrace{\mathbf{v}}_{0} + \int_{\Gamma_{in}\cup\Gamma_{out}}\underbrace{\left(p\mathbf{n} - \mu\frac{\partial \mathbf{u}}{\partial \mathbf{n}}\right)}_{\mathbf{g}_2}\cdot\mathbf{v} + \int_{\Gamma_b}\underbrace{\left(p\mathbf{n} - \mu\frac{\partial \mathbf{u}}{\partial \mathbf{n}}\right)}_{\boldsymbol{\sigma}\mathbf{n}}\cdot\mathbf{v}. \tag{48}$$

Then, the problem writes:
Find $\mathbf{u} \in \mathbf{V}$, $p \in Q$ such that

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega \mathbf{u}\cdot\mathbf{v} - \int_\Omega [(\mathbf{w}\cdot\nabla)\mathbf{u}]\cdot\mathbf{v} - \int_\Omega (\nabla\cdot\mathbf{w})\mathbf{u}\cdot\mathbf{v} + \mu\int_\Omega \nabla\mathbf{u} : \nabla\mathbf{v} - \int_\Omega p\,\mathrm{div}(\mathbf{v})$$
$$= \int_\Omega \mathbf{F}\cdot\mathbf{v} + \int_{\Gamma_{in}\cup\Gamma_{out}} \mathbf{g}_2\cdot\mathbf{v} + \int_{\Gamma_b}\boldsymbol{\sigma}\mathbf{n}\cdot\mathbf{v} \quad \forall \mathbf{v} \in \boldsymbol{X}, \tag{49}$$

$$\int_\Omega \mathrm{div}(\mathbf{u})\, q = 0 \quad \forall q \in Q. \tag{50}$$

With the notations (39)- (42), the problem writes :
Find $\mathbf{u} \in \mathbf{V}, p \in Q$ such that :

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega \mathbf{u}\cdot\mathbf{v} + a(\mathbf{u},\mathbf{v}) + b(\mathbf{v},q) = L(\mathbf{v}) \qquad \forall \mathbf{v}\in\mathbf{X}, \tag{51}$$

$$b(\mathbf{u},q) = 0 \qquad \forall q \in Q. \tag{52}$$

The bilinear forms a and b are defined by (41)-(42), and L is defined by:

$$L(\mathbf{v}) = \int_{\Omega(t)} \mathbf{F}\cdot\mathbf{v} + \int_{\Gamma_{in}\cup\Gamma_{out}} \mathbf{g}_2\cdot\mathbf{v} + \int_{\Gamma_b} \boldsymbol{\sigma}\mathbf{n}\cdot\mathbf{v}. \tag{53}$$

2.2.2. *Slip boundary conditions*

In this section, we are interested in the interface between the fluid and the topography and we preconise to have a slip boundary condition on Γ_b, which is physically consistent with the sediment transport model chosen in this study, namely the bedload transport. Then, we give the variational formulation with slip boundary condition $\mathbf{u}\cdot\mathbf{n} = 0$ on Γ_b. It is not natural to impose a slip boundary condition in the Stokes problem, and this problem has been widely studied:

- **First variational strategy**
  A first strategy, studied in [15], consists in giving a condition on the stress tensor $\boldsymbol{\sigma}\mathbf{n}\cdot\boldsymbol{\tau} = g_3$. We rewrite the test function $\mathbf{v} = (\mathbf{v}\cdot\mathbf{n})\,\mathbf{n} + (\mathbf{v}\cdot\boldsymbol{\tau})\,\boldsymbol{\tau}$ where $\mathbf{n}$ is the normal component and $\boldsymbol{\tau}$ is the tangential component on Γ_b. Taking $g_3 = 0$ and

$$\mathbf{u}\in\mathbf{W} = \{\mathbf{v}\in\mathbf{V}, \quad \mathbf{v}\cdot\mathbf{n}|_{\Gamma_b} = 0\}, \tag{54}$$

$$\mathbf{Y} = \{\mathbf{v}\in\mathbf{X}, \quad \mathbf{v}\cdot\mathbf{n}|_{\Gamma_b} = 0\}, \tag{55}$$

  it is straightforward to verify that the variational formulation writes

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega \mathbf{u}\cdot\mathbf{v} - \int_\Omega (\nabla\cdot\mathbf{w})\mathbf{u}\cdot\mathbf{v} - \int_\Omega [(\mathbf{w}\cdot\nabla)\mathbf{u}]\cdot\mathbf{v} \tag{56}$$

$$+\mu\int_\Omega \nabla\mathbf{u}:\nabla\mathbf{v} - \int_\Omega p\,\mathrm{div}(\mathbf{v}) = \int_\Omega \mathbf{F}\cdot\mathbf{v} + \int_{\Gamma_{in}\cup\Gamma out} \mathbf{g}_2\cdot\mathbf{v} \quad \forall\mathbf{v}\in\mathbf{Y},$$

$$\int_\Omega \mathrm{div}(\mathbf{u})\,q = 0 \qquad \forall q\in Q. \tag{57}$$

- **Second variational strategy**
  A second strategy consists in giving a condition on the velocity at the boundary, see [28], as follows $\mu\dfrac{\partial\mathbf{u}}{\partial\mathbf{n}}\cdot\boldsymbol{\tau} + \alpha(\mathbf{u}\cdot\boldsymbol{\tau}) = g$ with $\alpha > 0$. To this aim, we notice that

$$\int_{\Gamma_b} \boldsymbol{\sigma}\mathbf{n}\cdot\mathbf{v} = \int_{\Gamma_b} (\boldsymbol{\sigma}\mathbf{n}\cdot\mathbf{n})(\mathbf{v}\cdot\mathbf{n}) + (\boldsymbol{\sigma}\mathbf{n}\cdot\boldsymbol{\tau})(\mathbf{v}\cdot\boldsymbol{\tau}), \tag{58}$$

  and we rewrite the test function $\mathbf{v}$ in terms of the normal component and the tangential component, as for the previous case. Taking

$$\mathbf{u}\in\mathbf{W} = \{\mathbf{v}\in\mathbf{V}, \quad \mathbf{v} = 0 \text{ on } \Gamma_s, \quad \mathbf{v}\cdot\mathbf{n}|_{\Gamma_b} = 0\},$$

the variational formulation writes:

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega \mathbf{u}\cdot\mathbf{v} - \int_\Omega (\nabla\cdot\mathbf{w})\mathbf{u}\cdot\mathbf{v} - \int_\Omega [(\mathbf{w}\cdot\nabla)\mathbf{u}]\cdot\mathbf{v} \\
+\mu\int_\Omega \nabla\mathbf{u}:\nabla\mathbf{v} - \int_\Omega p\,\mathrm{div}(\mathbf{v}) + \int_{\Gamma_b}\alpha(\mathbf{u}\cdot\boldsymbol{\tau})(\mathbf{v}\cdot\boldsymbol{\tau}) &= \int_\Omega \mathbf{F}\cdot\mathbf{v} + \int_{\Gamma_b} g(\mathbf{v}\cdot\boldsymbol{\tau}) \qquad (59)\\
&\quad + \int_{\Gamma_{in}\cup\Gamma out}\mathbf{g}_2\cdot\mathbf{v}, \qquad \forall\mathbf{v}\in\mathbf{Y},\\
\int_\Omega \mathrm{div}(\mathbf{u})\,q &= 0 \qquad \forall q\in Q. \qquad (60)
\end{aligned}$$

- **Third variational strategy**
An other alternative leads to using a penalty method. As in a previous case, we take $\boldsymbol{X} = \{\mathbf{v}\in(\mathbf{H}^1(\Omega(t)))^2, \quad \mathbf{v}=0 \text{ on } \Gamma_s\}$. In order to impose the condition $\mathbf{u}\cdot\mathbf{n}=0$ for the velocity which is not natural in the variational formulation, we consider the formulation (51) and penalize the natural boundary condition:

$$\boldsymbol{\sigma}\mathbf{n}|_{\Gamma_b} = -\frac{1}{\varepsilon}(\mathbf{u}\cdot\mathbf{n})\mathbf{n},$$

where $\varepsilon \ll 1$. The variational formulation becomes:
Find $\mathbf{u}\in\mathbf{V}$ and $p\in Q$ such that:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega \mathbf{u}\cdot\mathbf{v} + \tilde{a}(\mathbf{u},\mathbf{v}) + b(\mathbf{v},p) &= L(\mathbf{v}), \quad \forall\mathbf{v}\in\mathbf{X}, \qquad (61)\\
b(\mathbf{u},q) &= 0 \quad \forall q\in Q, \qquad (62)
\end{aligned}$$

with the bilinear form b defined by (42), L and $\tilde{a}$ defined by

$$\tilde{a}(\mathbf{u},\mathbf{v}) = a_1(\mathbf{u},\mathbf{v}) + a_2(\mathbf{u},\mathbf{v}) + \frac{1}{\varepsilon}\int_{\Gamma_b}(\mathbf{u}\cdot\mathbf{n})(\mathbf{v}\cdot\mathbf{n})\,d\sigma, \qquad (63)$$

$$L(\mathbf{v}) = \int_\Omega \mathbf{F}\cdot\mathbf{v} + \int_{\Gamma_{in}\cup\Gamma out}\mathbf{g}_2\cdot\mathbf{v}. \qquad (64)$$

It is proved by Dione in [15] that this problem converges to the problem with slip boundary conditions when ε tends to zero.

3. Numerical method

3.1. Discretization in time

We will now discretise the time derivative with a Backward Differentiation Formula (BDF) of order 1. It corresponds to the backward Euler method. Let Δt be the time step, $t_0=0$ the initial time, $t_n = n\Delta t$ and x^n a field at time t_n.
We can then rewrite the variational formulation at time t_{n+1} :

$$\begin{aligned}
\int_\Omega \frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t}\cdot\boldsymbol{\varphi} + \tilde{a}(\mathbf{u}^{n+1},\boldsymbol{\varphi}) + b\left(\boldsymbol{\varphi},p^{n+1}\right) &= L\left(\boldsymbol{\varphi}\right), \quad \forall\boldsymbol{\varphi}\in\mathbf{V}, \qquad (65)\\
b(\mathbf{u}^{n+1},q) &= 0, \qquad \forall q\in W, \qquad (66)
\end{aligned}$$

since $\mathbf{u}^n$ is known, this leads to :

$$\int_\Omega \frac{1}{\Delta t}\mathbf{u}^{n+1}\cdot\boldsymbol{\varphi} + \tilde{a}(\mathbf{u}^{n+1},\boldsymbol{\varphi}) + b\left(\boldsymbol{\varphi}, p^{n+1}\right) = L\left(\boldsymbol{\varphi}\right) + \int_\Omega \frac{1}{\Delta t}\mathbf{u}^{n}\cdot\boldsymbol{\varphi}, \quad \forall \boldsymbol{\varphi}\in\mathbf{V}, \tag{67}$$

$$b(\mathbf{u}^{n+1}, q) = 0, \qquad \forall q \in W. \tag{68}$$

3.2. **Discretization in space**

We consider in this section a subdivided domain $\mathcal{T}_h$ and the finite dimensional spaces $\mathbf{V}_h$ and W_h which are the discrete spaces of $\mathbf{V}$ and W. As well, we approximate $\mathbf{u}$ and p by $\mathbf{u}_h$ and p_h. We use a Galerkin approximation taking the fields $\mathbf{u}_h$, and p_h in $\mathbf{V}_h$ and W_h, it reads:

$$\mathbf{u}_h = \sum_{i=1}^{N}\alpha_i\boldsymbol{\varphi}_i, \quad p_h = \sum_{i=1}^{M}\beta_i\psi_i, \tag{69}$$

where we denote $N = \dim\mathbf{V}_h$ and $M = \dim W_h$ and $\boldsymbol{\varphi}$ (resp. ψ) is the basis function of $\mathbf{V}_h$ (resp. W_h). Then, the semi discrete problem writes:

$$\int_\Omega \frac{1}{\Delta t}\mathbf{u}_h^{n+1}\cdot\boldsymbol{\varphi}_h + \tilde{a}(\mathbf{u}_h^{n+1},\boldsymbol{\varphi}_h) + b\left(\boldsymbol{\varphi}_h, p_h^{n+1}\right) = L\left(\boldsymbol{\varphi}_h\right) + \int_\Omega \frac{1}{\Delta t}\mathbf{u}_h^{n}\cdot\boldsymbol{\varphi}_h, \quad \forall \boldsymbol{\varphi}_h\in\mathbf{V}_h, \tag{70}$$

$$b(\mathbf{u}_h^{n+1}, q_h) = 0, \qquad \forall q_h \in W_h. \tag{71}$$

We choose the following compatible spaces (see [24]):

- $\mathbf{V}_h = \left\{\mathbf{v}\in\mathcal{C}^0\left(\overline{\Omega}\right) \text{ s.t. } \mathbf{v}_{|K}\in\mathbb{P}_2^2 \text{ for all } K\in\mathcal{T}_h\right\}$,
- $W_h = \left\{q\in\mathcal{C}^0\left(\overline{\Omega}\right) \text{ s.t. } q_{|K}\in\mathbb{P}_1 \text{ for all } K\in\mathcal{T}_h\right\}\cap L_0^2\left(\Omega\right)$,

where $\mathcal{T}_h$ is the set of mesh elements and $\mathbb{P}_k$ is the set of polynomial function of degree k. The space $\mathbf{V}_h$ can be adjusted for specific boundary conditions.
Throughout the rest of the document, we use these functionals sets.
We denote by U^n, P^n the vectors

$$U^n = \begin{pmatrix}\alpha_1^n\\ \vdots\\ \alpha_N^n\end{pmatrix}, \quad P^n = \begin{pmatrix}\beta_1^n\\ \vdots\\ \beta_M^n\end{pmatrix}, \tag{72}$$

A the matrix $A_{i,j} = \tilde{a}(\boldsymbol{\varphi}_i,\boldsymbol{\varphi}_j)$ for $1\le i,j\le N$ and B the divergence matrix $B^T_{i,j} = b(\varphi_i,\psi_j)$ for $1\le i\le N, 1\le j\le M$. We also note $F = (\gamma_i)^T$ where $\mathbf{f} = \sum_{i=1}^N\gamma_i\boldsymbol{\varphi}_i$ and M the mass matrix.
The problem writes:

$$\begin{pmatrix}A + M/\Delta t & B^T\\ B & 0\end{pmatrix}\begin{pmatrix}U^{n+1}\\ P^{n+1}\end{pmatrix} = \begin{pmatrix}F\\ 0\end{pmatrix} + \begin{pmatrix}M/\Delta t & 0\\ 0 & 0\end{pmatrix}\begin{pmatrix}U^{n}\\ 0\end{pmatrix}. \tag{73}$$

3.3. **Stokes-Exner coupling**

Although the algorithm for the instationary Stokes equations described in the previous section constitutes the main part of the complete method, we give in this part the complete implemented algorithm. We denote by N_T the final time of the discrete problem and $t^n = n\Delta t$. Starting from an initial topography, the first step consists in solving the Stokes equation in the domain Ω^n delimited by the initial bottom using the method presented before. Then, denoting the velocity of the fluid at time t^n, by $\mathbf{u}^n$ and the velocity of the displacement of the mesh by $\mathbf{w}^n$, we solve the Stokes equation for this initial state. This resolution gives the numerical solution $\mathbf{u}^{n+1}$ at time t^{n+1}. We denote this solver by **StokesSolver**$(\Omega^n,\mathbf{u}^n,\mathbf{w}^n)$. This allows to compute the new topography b^{n+1} using the Exner equation and giving the velocity $\mathbf{u}^{n+1}$ and the bottom at time t^n. We note this method by **ExnerSolver**$(b^n,\mathbf{u}^{n+1})$ which allows to compute the displacement of the mesh Ω^n at the

boundary Γ_b, that is to say at the interface. Then, it is necessary to extend the deformation in the domain to compute the new one. To do so, the method $\mathbf{ALESolver}(\Omega^n, \mathbf{d}_{\Gamma_b}^{n+1})$ solves the harmonic problem from the domain at time t^n and then, gives the deformation d^{n+1} that needs to be applied. Finally, the velocity $\mathbf{w}^{n+1}$ and the mesh $\Omega^{n+1}$is computed from the displacement. At this step, all the states are obtained for the time t^{n+1}.
The coupled algorithm can be summarized by the following:

Algorithm 1 Stokes-Exner coupling

Require: Ω^0, b^0, $\mathbf{w}^0$, $\mathbf{u}^0$

for $n = 0$ to $N_T - 1$ **do**
$\mathbf{u}^{n+1} = \mathbf{StokesSolver}(\Omega^n, \mathbf{u}^n, \mathbf{w}^n)$
$b^{n+1} = \mathbf{ExnerSolver}(b^n, \mathbf{u}^{n+1})$
$\mathbf{d}_{\Gamma_b}^{n+1} = b^{n+1} - b^n$
$\mathbf{d}^{n+1} = \mathbf{ALESolver}(\Omega^n, \mathbf{d}_{\Gamma_b}^{n+1})$
$\mathbf{w}^{n+1} = \frac{\mathbf{d}^{n+1} - \mathbf{d}^n}{\Delta t}$
$\Omega^{n+1} = \Omega^n + \mathbf{d}^{n+1}$
end for

4. Numerical tests

4.1. Validation with analytical solutions of the Stokes equations

In order to validate the numerical method proposed and implemented with Feel++, we compare the numerical results with analytical solutions of the Stokes problem.

4.1.1. *Solution of Bercovier-Engelman*

First of all, in order to validate the implementation of the Stokes problem only, we use the solution of Bercovier-Engelman [7], which consists in finding a velocity that satisfies the free divergence condition and is null on the whole boundary. From this velocity and a source term $\mathbf{f}$, we deduce gradient pressure, and then a pressure.

$$\mathbf{v} = \begin{pmatrix} -256z(z-1)(2z-1)x^2(x-1)^2 \\ 256x(x-1)(2x-1)z^2(z-1)^2 \end{pmatrix},$$
$$p = (x-0.5)(z-0.5),$$
$$\mathbf{f} = \begin{pmatrix} 256(x^2(x-1)^2(12z-6) + z(z-1)(2z-1)(12x^2-12x+2)) + (z-0.5) \\ -256(z^2(z-1)^2(12x-6) + x(x-1)(2x-1)(12z^2-12z+2)) + (x-0.5) \end{pmatrix}.$$

We can compare the exact solution and the approximation in Fig 3.

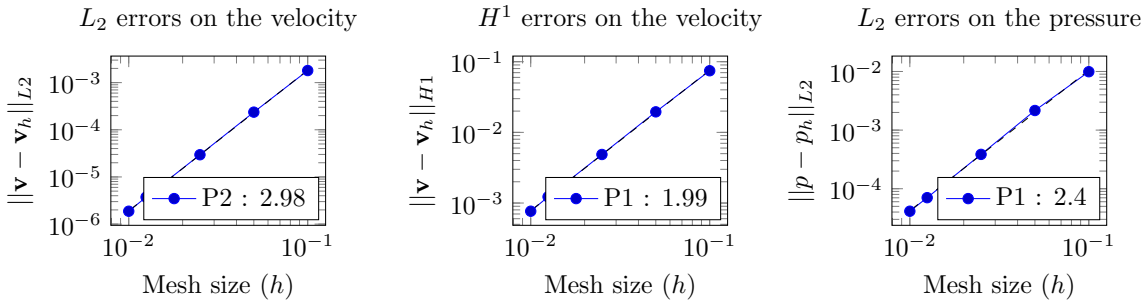
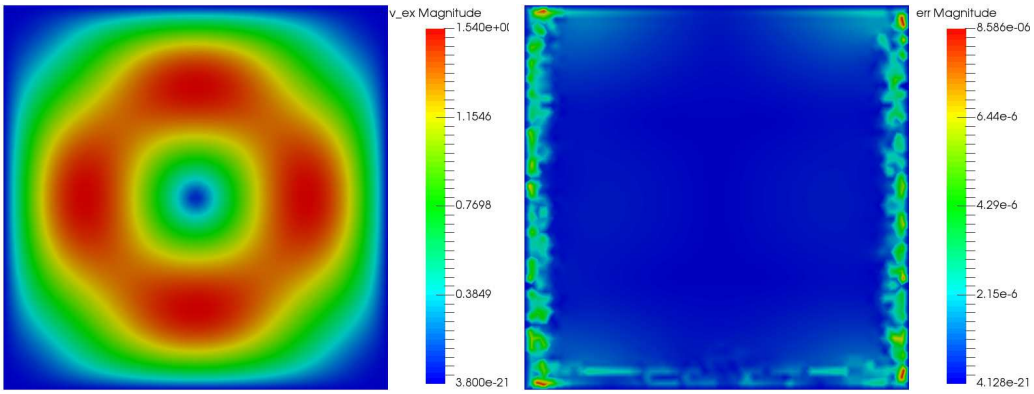


FIGURE 4. Convergence rates for the Bercovier-Engelman solution.



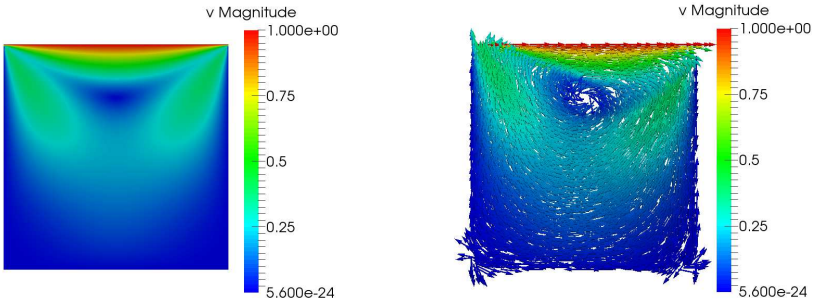
(A) Exact solution (B) Error

FIGURE 3. Velocity field of the exact solution and error with the numerical solution.

In Figure 4, we compute the errors between the exact and the computed solution and plot these errors versus the mesh size (in log-log scale). We can then verify that the method converges and that the convergence orders are 3 for the L2-norm of the velocity and 2 for the pressure, which are those expected by the theory with the finite elements chosen here.

4.1.2. *Driven cavity*

The second test case, the driven cavity, is a very classical test case in fluid dynamics. We verify again that the Stokes problem is well solved but with more physical boundary conditions that will be useful in the sequel. Indeed, to obtain this solution, we impose $v|_{\Gamma_{in} \cup \Gamma_{out} \cup \Gamma_b} = 0$ and $v|_{\Gamma_s} = (1, 0)^T$. The numerical results are shown in Figure 5.



(A) Magnitude field (B) Velocity field

FIGURE 5. Driven cavity.

Notice that the discontinuity of the velocity of the corners of the cavity is due to the discontinuity of the velocity imposed by the Dirichlet condition. It does not infer on the training, but to avoid this result, a polynomial function can be set instead of the constant.

4.2. **Fluvial dune test case**

To validate our complete coupled model with the Exner equation, we take an initial dune given by the equation :

$$b_z(x,0) = 0.2 \times e^{-\frac{(x-2.5)^2}{0.2}}, \forall x \in [0,l], \tag{74}$$

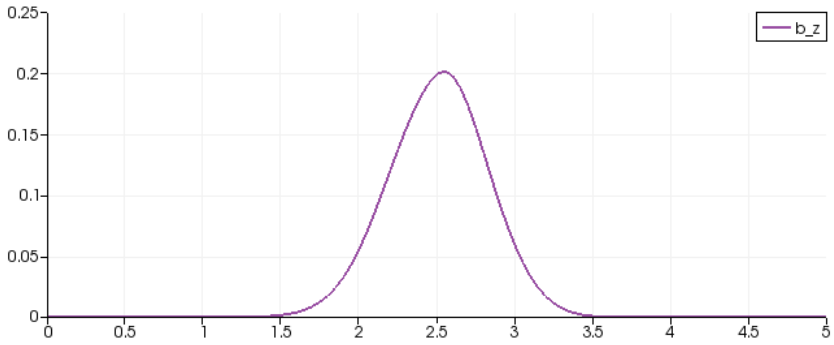
with $l = 5$.

For the unsteady Stokes equation, we use the slip boundary condition (4)-(7) and let free the velocity at the inlet and outlet with a Neumann boundary condition (5) with $\mathbf{g}_2 = \mathbf{0}$. On the top, we impose the same Dirichlet condition than in the driven cavity $\mathbf{u} = (1,0)^T$, driven the fluid to the right. For the Exner model, we use the Grass formula (10) and the initial data given by (74). We use the numerical method presented above : finite element method for spatial discretization and Implicit Euler method for time discretization for all the equations. As the cavity is driven with a moving bottom, we add the ALE formulation and obtain the results showed in Figure 6. We use the penalty method for bottom condition. The result is given in figure 6 and is similar to that of E. J. Kubatko and J. J. Westerink [31] (Fig. 2 of their paper). The same test case has been tested with multiple dunes and a similar result (distortion to the right) was obtained. This kind of solution can be difficult to represent with a numerical scheme because the solution becomes discontinuous but the algorithm stays stable during the simulation. This test case allows us to evaluate the relevance of our method but a comparison with an analytical solution is necessary to validate the method.

Remark 1. *It is not straightforward to validate the method on the complete model with an analytical solution. In [31], it is explained that the Exner model is an hyperbolic equation. Indeed, we can write Exner equation* (11) *(with* $\xi = 1$*) as:*

$$\frac{\partial b_z}{\partial t} + c(b_z)\frac{\partial b_z}{\partial t} = 0, \tag{75}$$

where $c(b_z) = \dfrac{\partial Q(\mathbf{u})}{\partial b_z}(b_z)$ *can be interpreted as a bed velocity (that depends on* b_z *because* $\mathbf{u}$ *in* (2)-(3) *does). It is obvious that the conservative law* (75) *is nonlinear. It is well known that a nonlinear conservative law is sometimes subject to discontinuities. Even if the initial topography* b_0 *is smooth, if characteristics intersect at time* $t_d > 0$*, there is no unique solution and a discontinuity can appear. For a time greater than* t_d*, the model no longer holds, there are several unphysical solutions. To solve this problem, it is classic to add an*



(A) $t = 0s$

(B) $t = 8s$

(C) $t = 16s$

FIGURE 6. Bottom topography at different times

artificial viscosity or use the integral form of the Exner model (see [31] for details). In order to avoid that kind of difficulty, we consider small simulation times and let these considerations to further studies.

5. Conclusion

In this note, we consider a coupling between the Exner equation and the Stokes equations to model the transport sediments in flow phenomena. We focus on a model without free surface and use some numerical tests to evaluate the relevance of the method. The fluid structure interaction theory and method have been applied and the objective is to test the proposed method which can be extend to a free surface model. The library Feel++ and the high computing performance embedded have been used to test the solution method. Therefore, the final goal of this project is to understand the impact of the sediment transport on the flow using Navier-Stokes with a free surface system coupled with the standard Exner equation.

References

[1] E Audusse. A multilayer Saint-Venant model: Derivation and numerical validation. *Discret. Contin. Dyn. Syst. Ser. B*, 5(2):189–214, 2005.

[2] Emmanuel Audusse, François Bouchut, Marie-Odile Bristeau, and Jacques Sainte-Marie. Kinetic Entropy Inequality and Hydrostatic Reconstruction Scheme for the Saint-Venant. *eprint arXiv:1409.3825*, pages 1–21, 2000.

[3] Emmanuel Audusse, Christophe Chalons, Olivier Delestre, Nicole Goutal, Magali Jodeau, Jacques Sainte-Marie, Jan Giesselmann, and Georges Sadaka. Sediment transport modelling : relaxation schemes for Saint-Venant - Exner and three layer models. *ESAIM: Proceedings*, 38:78–98, 2012.

[4] Emmanuel Audusse, Christophe Chalons, and Philippe Ung. A simple three-wave Approximate Riemann Solver for the Saint-Venant–Exner equations. working paper or preprint, September 2015.

[5] M. Larson B. Camenen. A general formula for non-cohesive bed load sediment transport. *Estuarine Coastal Shelf Sci*, pages 249–260, 2005.

[6] T. Belytschko and J. M. Kennedy. Computer models for subassembly simulation. *Nuclear Engineering and Design*, 49, 1978.

[7] Michel Bercovier and Michael Engelman. A finite element for the numerical solution of viscous incompressible flows. *Journal of Computational Physics*, 30(2):181–201, 1979.

[8] P. Bonneton, F. Chazel, D. Lannes, F. Marche, and M. Tissier. A splitting approach for the fully nonlinear and weakly dispersive green-naghdi model. *Journal of Computational Physics*, 230(4):1479 – 1498, 2011.

[9] François Bouchut. *Nonlinear stability of finite volume methods for hyperbolic conservation laws and well-balanced schemes for sources*. Frontiers in mathematics. Birkhäuser, Basel, Boston, Berlin, 2004.

[10] Marie-Odile Bristeau, Nicole Goutal, and Jacques Sainte-Marie. Numerical simulations of a non-hydrostatic shallow water model. *Comput. Fluids*, 47(1):51–64, 2011.

[11] Marie-Odile Bristeau, Anne Mangeney, Jacques Sainte-Marie, and Nicolas Seguin. An energy-consistent depth-averaged Euler system: Derivation and properties. *Discret. Contin. Dyn. Syst. - Ser. B*, 20(4):961–988, 2015.

[12] M. J Castro Diaz, E. D Fernandez, C Pares, and a. M Ferreiro. Two-dimensional sediment transport models in shallow water equations. A second order finite volume approach on unstructured meshes. *Computer Methods in Applied Mechanics and Engineering*, 198:2520–2538, 2009.

[13] Vincent Chabannes. *Vers la simulation des écoulements sanguins*. PhD thesis, Université de Grenoble, 2013.

[14] S. Cordier, M.H. Le, and T. Morales de Luna. Bedload transport in shallow water models: Why splitting (may) fail, how hyperbolicity (can) help. *Adv. Water Resour.*, 34(8):980–989, 2011.

[15] Ibrahima Dione. *Analyse théorique et numérique des conditions de glissement pour les fluides et les solides par la méthode de pénalisation*. PhD thesis, Université de Laval, 2013.

[16] J. Donéa. Analysis of the levelling process based upon an analytic forming model. *Advances Science Publishers*, 3, 1978.

[17] J. Donéa, S. Giuliani, and J.-P. Halleux. An arbitrary lagrangian-eulerian fem for transient dynamic fluid-structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 33, 1982.

[18] A. Duran and F. Marche. Discontinuous-galerkin discretization of a new class of green-naghdi equations. *Communications in Computational Physics*, page pp.130, 2014.

[19] R. Mueller E. Meyer-Peter. Formulae for bed-load transport. *Report on the 2nd Meeting International Association Hydraulic Structure Research*, pages 39–64, 1948.

[20] H.A. Einstein. The bed-load function for sediment transportation in open channel flows. *Tech. Rep. 1026,US Department of Agriculture, Technical Bulletin*, 1950.

[21] E. D. Fernández-Nieto, T. Morales de Luna, G. Narbona-Reina, and J. D. Zabsonré. Formal deduction of the Saint-Venant-Exner model including arbitrarily sloping sediment beds and associated energy. *ArXiv*, pages 1–44, 2015.

[22] R. M. Frank and R. B. Lazarus. Mixed eulerian-lagrangian method. *Academic Press*, 3, 1964.

[23] J.-F. Gerbeau and Benoit Perthame. Derivation of viscous Saint-Venant system for laminar shallow water; Numerical validation. *Discret. Contin. Dyn. Syst. - Ser. B*, 1(1):89–102, 2001.

[24] V. Girault and P-A. Raviart. *Finite element methods for Navier-Stokes equations*, volume 5 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1986. Theory and algorithms.

[25] Edwige Godlewski and Pierre-Arnaud Raviart. *Numerical approximation of hyperbolic systems of conservation laws*. Applied mathematical sciences. Springer, New York, 1996.
[26] C Grandmont. *Analyse mathématique et numérique de problèmes d'interaction fluide-structure. Application à la modélisation de l'appareil respiratoire.* Habilitation à diriger des recherches, Université Pierre et Marie Curie - Paris VI, November 2009. Dans le cadre de l'ANR M3RS (ANR-08-JCJC-0013-01).
[27] J. Grass. Sediments transport by waves and currents. *SERC London Cent. Mar. Technol., Report No. FL29*, 1981.
[28] J. L. Guermond and P. D. Minev. A new class of massively parallel direction splitting for the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 200(23-24):2083–2093, 2011.
[29] Putu Harry Gunawan and Xavier Lhébrard. Hydrostatic relaxation scheme for the 1D shallow water - Exner equations in bedload transport. *Comput. Fluids*, 121:44–50, 2015.
[30] A. Kalinske. Movement of sediment as bed load in rivers. *Earth and Space Sciences News*, pages 615–620, 1947.
[31] Ethan J. Kubatko and Joannes J. Westerink. Exact Discontinuous Solutions of Exner's Bed Evolution Model : Simple Theory for Sediment Bores. *Journal of Hydraulic Engineering*, 2007.
[32] D Lannes and F Marche. A new class of fully nonlinear and weakly dispersive Green-Ngaghdi models for efficient 2 d simulations. *J. Comput. Phys.*, 282:238–268, 2014.
[33] P. Nielsen. Movement of sediment as bed load in rivers. *Cambridge University Press*, pages 615–620, 1947.
[34] P. Nielsen. Coastal Bottom Boundary Layers and Sediment Transport. *Advanced Series on Ocean Engineering*, vol. 4, 1992.
[35] P. Nielsen. Erosion and Sedimentation. *Cambridge University Press*, 1998.
[36] W. F. Noh. CEL : a time-dependent two-space-dimensional. Coupled Eulerian-Lagrangian code. *Academic Press*, 3, 1964.
[37] Gary Parker, Chris Paola, and Suzanne Leclair. Probabilistic Exner Sediment Continuity Equation for Mixtures with No Active Layer, 2000.
[38] B. Perthame and C. Simeoni. A kinetic scheme for the Saint-Venant system with a source term. *Calcolo*, 38(4):201–231, 2001.
[39] Christophe Prud'Homme, Vincent Chabannes, Vincent Doyeux, Mourad Ismail, Abdoulaye Samake, and Gonçalo Pena. Feel++: A Computational Framework for Galerkin Methods and Advanced Numerical Methods. December 2012.
[40] Christophe Prud'Homme, Vincent Chabannes, Vincent Doyeux, Mourad Ismail, Abdoulaye Samake, and Gonçalo Pena. Feel++: A computational framework for galerkin methods and advanced numerical methods. In *ESAIM: Proceedings*, volume 38, pages 429–455. EDP Sciences, 2012.
[41] Christophe Prud'homme, Vincent Chabannes, Stephane Veys, Vincent Huber, Cécile Daversin, Alexandre Ancel, Ranine Tarabay, Vincent Doyeux, Jean-Baptiste Wahl, Christophe Trophime, Abdoulaye Samake, Guillaume Dollé, and Alexandre Ancel. feelpp v0.98.0. May 2014.
[42] G. Smart. Sediment transport formula for steep channels. *J. Hydraul*, 3:267–276, 1984.

Appendix B

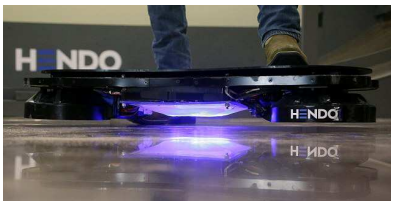
Résumé de thèse en français

B.1 Introduction

Les champs magnétiques sont naturellement présents partout dans notre vie. Par exemple, la Terre émet un champ magnétique de 4.7×10^{-5} Tesla et le cerveau humain émet un champ de 10^{-12} Tesla. Nous l'utilisons aussi dans des objets de tous les jours, tels que des sonnettes, haut-parleurs, moteurs de voiture ou disques durs de nos ordinateurs. D'autres usages incluent la détection de cellules cancéreuses par imagerie par résonance magnétique ou IRM, ou la compensation de la gravité, permettant de reproduire les conditions de l'espace sur Terre de manière moins onéreuse ou de faire léviter des objets, tels que les trains Maglev. Ce type d'application a besoin de champs magnétiques avec des profils spécifiques. Les aimants peuvent aussi être utilisé en tant qu'outils dans de nombreux domaines de recherche, tels que la résonance magnétique nucléaire des solides, pour lequel plus le champ est intense, meilleure est la précision.

(a) Scanner IRM

(b) Train Maglev

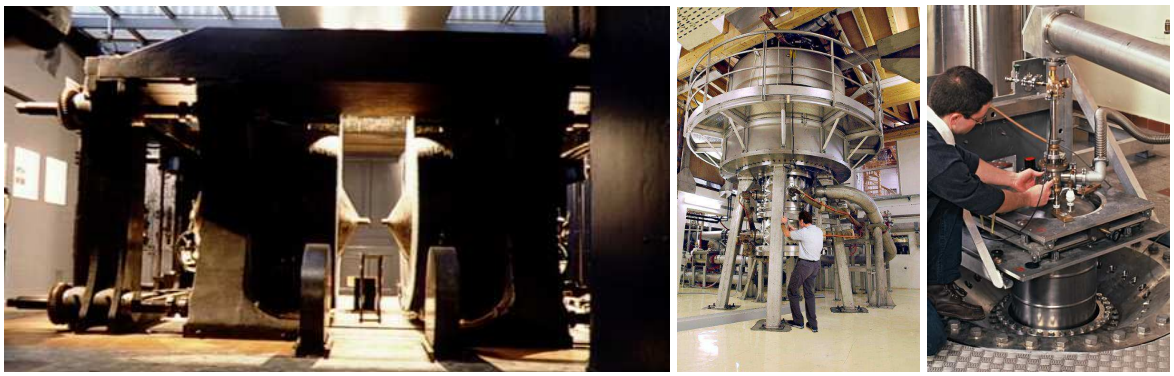


(c) Hendo Hoverboard

Figure B.1 – Applications du champ magnétique

La conception des aimants à hauts champs dans le but de la recherche scientifique a commencé au début du 20e siècle. Dans les années 1940, des techniques plus modernes ont vu le jour avec les travaux de Francis Bitter, voir [Montgomery, 1969]. À partir des années 1960, l'utilisation des matériaux superconducteurs s'est répandue. Le record d'intensité en utilisant de tels matériaux est de $23,5\ T$, et a été atteint au Ultra-High Field European NMR Center à Lyon en France. Mais due aux limitations de la superconductivité, au-dessus de $\approx 24\ T$, les matériaux superconducteurs standards perdent leurs propriétés.

Pour atteindre plus de $24T$, des aimants résistifs sont utilisés. Ceux-ci sont composés de matériels résistifs, tels que des alliages de cuivre. À cause de la chaleur dégagée, ces aimants ont besoin d'être refroidis avec de l'eau, ce qui rend leur construction et leur opération très coûteuses. Peu de laboratoires ont donc la possibilité de faire fonctionner de tels aimants, en Europe, ils sont regroupés dans le European Magnetic Field Laboratory (EMFL). En France, le Laboratoire National des Champs Magnétiques Intenses (LNCMI) est situé sur deux sites, à Toulouse et Grenoble. Ce dernier produit des champs magnétiques de 37 T pendant plusieurs heures.

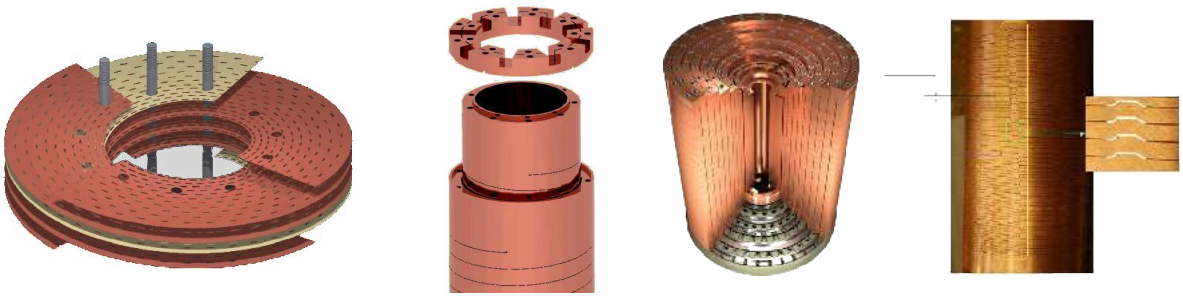


(a) Aimant Bellevue (b) Aimant Hybrid (c) Chambre d'expérimentation du LNCMI

Figure B.2 – L'évolution des aimants dans le temps

Le design d'un aimant joue un rôle important dans l'intensité qu'il peut générer. C'est pourquoi au LNCMI, différentes technologies sont testées et combinées. Le type d'aimant le plus utilisé est l'aimant Bitter, constitué de disques conducteurs, empilés en un solénoïde, et refroidit en passant de l'eau à travers des trous faits dans ces disques. Le record d'intensité avec un tel aimant est de 41 T au NHMFL aux États-Unis [Toth and Bole, 2018]. Les aimants poly-hélices sont un autre type d'aimant, dont la description peut être trouvé dans [Debray et al., 2002] ou [Debray et al., 2012]. Ces aimants sont faits de plusieurs tubes d'alliages de cuivre, coupés hélicoïdalement par électroérosion. Enfin, les aimants hybrides combinent les aimants superconducteurs et résistifs. Le record avec de tels aimants est de 46 T au NHFML et le LNCMI est en train de construire un aimant hybride pouvant générer 42 T.

Le LNCMI a besoin de continuellement améliorer le design de ses aimants pour pouvoir rivaliser avec les autres laboratoires internationaux du point de vue de l'intensité ou de l'homogénéité des champs magnétiques produits. Durant les expériences, le courant électrique peut atteindre 31 kA avec une puissance de 30 MW, nécessitant le refroidissement de l'aimant par un écoulement d'eau de 140 L/s, permettant l'évacuation de $6kW/m^2$, ou environ $150°C$, et les matériaux composant l'aimant sont poussés à 90% de leur limite élastique. Dans de telles conditions, il est important de s'assurer que l'aimant n'est pas endommagé durant l'expérience. En effet, un aimant résistif coûte jusqu'à 300 000 euros et un an pour le concevoir et l'assembler, alors que l'aimant hybride développé à Grenoble coûte jusqu'à 3 millions d'euros. Pour comprendre



(a) Aimant Bitter
(jusqu'à 1m de diamètre)

(b) Aimant Poly-hélices
(jusqu'à 400mm de diamètre)

Figure B.3 – Différentes technologies d'aimants résistifs

les physiques en jeu dans l'aimant ou créer des profils de champs spécifiques, les ingénieurs du LNCMI se servent de simulations numériques. C'est la raison à l'origine du projet HifiMagnet, qui est une collaboration entre le LNCMI et l'université de Strasbourg.

Cette thèse continue le travail entamé par C. Daversin dans [Daversin Catty, 2016] pour fournir un moyen efficace et fiable pour simuler des modèles 3D multi physiques sur des géométries réelles d'aimants. Les simulations doivent être faisables pour des problèmes mono physiques ou multi physiques, sur une partie ou sur l'ensemble de l'aimant, et ainsi doivent êtres aisément configurables. L'un des buts du projet est de permettre aux ingénieurs de LNCMI de lancer des simulations à l'aide du portail MSO4SC dans le cloud, qui fait partie du projet européen H2020 et qui permet d'utiliser des solutions de calculs haute performance à partie du cloud. Durant cette thèse, nous avons essayé d'utiliser la méthode de Galerkin Discontinue Hybride pour approcher plus précisément les variables de flux, tel que la densité électrique ou le champ magnétique. Une publication est en cours pour détailler le traitement de condition aux bords intégrales (IBC) [Guidoboni et al., 2020]. Enfin, pour diminuer le coût de calcul et permettre d'effectuer de l'optimisation ou de la quantification d'incertitudes, nous utilisons la méthode des bases réduites (RB), qui va nous aider à calculer le champ magnétique de manière efficace pour différents paramètres.

Toutes les méthodes présentées ont été implémentées en utilisant la librairie d'éléments finis en C++ `Feel++` [Prud'Homme et al., 2012].

Publications

en préparation	*An implementation of HDG methods with Feel++. Application to problems with integral boundary conditions* with C. Prud'homme, L. Sala, S. Bertoluzza, G. Guidoboni, D. Prada, R. Sacco and M. Szopos

2016	ESAIM: Proceedings and Surveys, EDP Sciences, 2016, CEMRACS 2015: Coupling multi-physics models involving fluids, 55, pp.23-40. *Hydromorpho: A coupled model for unsteady Stokes/Exner equations and numerical results with Feel++ library* with C. Prud'homme, N. Aissiouene, T. Amtout, M. Brachet, E. Frénod, A. Rousseau, S. Salmon

Oral Presentations

2019	**Labex IRMIA**, Strasbourg, France *Control and Optimization of high field magnets*
2018	**WCCM**, New-York, USA *High Reynolds Aerothermal Simulations and Reduced Basis*
2018	**CANUM**, Cap d'Agde, France *Optimization and control of magnetic high fields*
2017	**`Feel++` User Days**, Strasbourg, France *HDG Methods in Feel++*
2017	**ANR CHORUS Workshop**, Paris, France *Model Order Reduction for Mutliphysic Problems, Using the Open-Source Framework Feel++*
2016	**CEMRACS**, Marseille, France *Cemracs 2016 : Numerical Challenges in Parallel Scientific Computing*

Poster

2018	**MoRePaS**, Nantes, France *Towards real time computation of 3D magnetic field in parametrized Polyhelix magnets using a reduced basis Biot-Savart model*
2017	**MT25**, Amsterdam, Pays-Bas *Towards real time computation of 3D magnetic field in parametrized Polyhelix magnets using a reduced basis Biot-Savart model*
2016	**Journée Poster de l'école doctorale**, Strasbourg, France *Optimization and control of magnetic high fields*

B.2 Méthodes Numériques

B.2.1 Méthode de Galerkin Discontinue Hybride

Les méthodes de Galerkin Discontinues (DG) sont intéressantes, car elles sont bien adaptées pour l'adaptabilité *hp* [Berger and Colella, 1989, Bey et al., 1996], permettent d'imposer simplement les conditions aux bords et sont faciles à paralléliser [Baggag et al., 1999]. Mais en contrepartie, l'augmentation considérable du nombre de degrés

de liberté les rend très coûteuses en comparaison des méthodes de Galerkin Continues (CG). Des méthodes DG utilisant une technique d'optimisation appelée *condensation statique* ont été introduites dans [Cockburn et al., 2009b] pour résoudre cet inconvénient.
Dans des problèmes impliquant une variable primale et un flux, tel qu'un potentiel et son gradient, on introduit la trace de la variable primale comme nouvelle variable. En utilisant la condensation statique, on peut exprimer les variables primale et de flux en fonction de cette trace. On peut ensuite calculer la trace en résolvant un problème global, mais seulement sur les faces des éléments. Et les variables primale et de flux peuvent être retrouvées en résolvant des problèmes locaux sur chaque élément.
Ces méthodes ont plusieurs avantages, tels que la convergence optimale pour le flux dans le cas problèmes de convection diffusion, des post traitements permettant d'obtenir une meilleure convergence ou d'avoir des propriétés de conservation pour le flux dans H_{div}, [Cockburn et al., 2009c]. Et grâce aux nombreux problèmes locaux à résoudre, ces méthodes restent hautement parallélisables.

Les méthodes HDG ont été développées pour les équations de Stokes et Navier-Stokes [Nguyen et al., 2010, Nguyen et al., 2011], l'élasticité, linéaire et non linéaire [Kabaria et al., 2014, Qiu et al., 2013] et plus récemment pour les équations de Maxwell [Lu et al., 2015, Chen et al., 2017]. Nous pouvons donc les utiliser pour décrire les physiques du refroidissement, de la déformation et des forces magnétiques des aimants.

De plus, la formulation pour les problèmes elliptiques de second ordre, développé par [Cockburn et al., 2009b] est adaptée aux problèmes thermiques et électriques.

$$\begin{cases} \mathbf{u} + \kappa \nabla p = 0 \text{ in } \Omega \\ \nabla \cdot \mathbf{u} = f \text{ in } \Omega \end{cases}$$

Nous avons développé une méthode permettant d'utiliser des conditions aux bords intégrales (IBC) qui vont nous permettre d'être le plus proche possible des procédures expérimentales du LNCMI du point de vue de la modélisation. En effet, parfois on ne peut pas imposer directement la valeur du champ p, ni celui du flux normal $\nabla p \cdot \mathbf{n}$ sur une surface. À la place, on veut que le champ p soit une constante sur la surface, mais sans connaître sa valeur, et l'on veut imposer la valeur de l'intégrale du flux normal sur cette surface.

$$\int_{\Gamma_I} \mathbf{u} \cdot \mathbf{n} = g_I \text{ et } p \text{ est une constante inconnue sur } \Gamma_I \text{ telle que } |\Gamma_I| p - \int_{\Gamma_I} p = 0$$

C'est intéressant dans notre cas où p est le potentiel électrique. En effet, nous n'avons pas accès durant les expériences à sa valeur, mais le flux global à travers la surface est une donnée de l'expérience. Une publication est en préparation sur cette méthode [Guidoboni et al., 2020] où des résultats théoriques sont montrés et des applications réelles sont montrées.

B.2.2 Méthodes de réduction d'ordre

Même avec une puissance de calcul en constante progression, effectuer beaucoup de calcul avec des méthodes HDG ou même CG peut devenir trop coûteux. C'est

le cas lorsque les équations dépendent de paramètres $\boldsymbol{\mu}$ et que l'on doit résoudre ces équations pour un grand nombre de paramètres.

$$a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f(v; \boldsymbol{\mu}) \quad \forall v \in V$$

Cela peut arriver si l'on veut quantifier les incertitudes de notre problème, optimiser une partie du problème, contrôler l'état d'une expérience ou simplement explorer l'ensemble des solutions.

Pour diminuer la complexité des calculs, nous utilisons les méthodes de réduction d'ordre (MOR). Une possibilité pour réduire l'ordre d'un modèle est de simplifier les physiques rentrant en jeu dans le problème, mais la précision des solutions diminue de la même manière. D'autres méthodes projettent les équations sur un espace avec une dimension réduite par rapport à un modèle haute fidélité. Ces méthodes sont par exemple la décomposition généralisée aux valeurs propres [Chinesta et al., 2011], la décomposition orthogonale aux valeurs propres [Kerschen et al., 2005] ou la méthode des bases réduites [Noor, 1981, Prud'Homme et al., 2001, Patera and Rozza, 2007]. Nous nous intéresserons en particulier à cette dernière.

Pour construire l'espace de dimension réduite, nous allons utiliser une méthode gloutonne. Pour cela nous avons besoin d'un estimateur d'erreur $\eta(\boldsymbol{\mu})$ tel que:

$$\|u_h(\boldsymbol{\mu}) - u_{rb}(\boldsymbol{\mu})\|_{\bar{\mu}} \leq \eta(\boldsymbol{\mu}), \quad \forall \boldsymbol{\mu} \in \mathbb{P}$$

Pour calculer cet estimateur d'erreur de manière efficace, il nous faudra être capables de calculer efficacement le représentant de Riesz du résidu de notre équation, ainsi que la constante de stabilité. Cette dernière peut être calculé à l'aide des méthodes de Min-θ, multi paramètres Min-θ [Machiels et al., 2000], [Veroy, Karen et al., 2002], [Patera and Rozza, 2007] ou bien la méthode des contraintes successives (SCM) [Huynh et al., 2007], [Vallaghé et al., 2011].

L'algorithme glouton est une procédure itérative où à chaque étape, on agrandit la base réduite avec la solution $u_h(\boldsymbol{\mu})$ où $\boldsymbol{\mu}$ est le paramètre qui maximise l'erreur $\eta(\boldsymbol{\mu})$.
Pour pouvoir calculer la solution du problème le plus rapidement possible, nous allons utiliser une stratégie hors ligne/en ligne, où les plus gros calculs seront précalculés. Pour cela, on cherche à écrire notre problème sous la forme d'une décomposition affine:

$$\sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}) a_q(w, v) = \sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}) f_q(v)$$

où les formes bilinéaires et linéaires a_q et f_q ne dépendent plus des paramètres $\boldsymbol{\mu}$.

Dans le cas où nous ne sommes pas capables d'obtenir une telle décomposition, dans le cas de problèmes non linéaires par exemple, nous avons besoin d'outils supplémentaires pour utiliser la stratégie hors ligne/en ligne.
Dans un premier temps, nous avons utilisé la méthode d'interpolation empirique (EIM) décrite dans [Barrault et al., 2004], [Grepl, Martin A. et al., 2007], [Maday et al., 2008] ou

[Daversin, C. et al., 2013] pour approcher une expression paramétrer par une somme de termes:

$$g(x,\boldsymbol{\mu}) \approx g_M(x,\boldsymbol{\mu}) = \sum_{m=1}^{M} \theta_{g,M}^{m}(\boldsymbol{\mu}) q_m(x)$$

Cette méthode utilise encore une stratégie hors ligne/en ligne et un algorithme glouton pour trouver les meilleurs points d'interpolation.
Dans le cas d'opérateurs très complexes, il peut être utile d'utiliser la version discrète de cet algorithme [Chaturantabut and Sorensen, 2010] pour approcher un vecteur ou une matrice, représentant par exemple une forme linéaire ou bilinéaire avec des transformations géométriques.
Dans le cas non linéaire, ces méthodes peuvent avoir un coût très élevé, c'est pourquoi nous avons aussi utilisé une méthode pour construire simultanément la base réduite et les interpolations empiriques (SER) [Daversin and Prud'Homme, 2015].
Enfin, pour réduire le temps de calcul de certaines intégrales, nous avons implémenté la méthode des quadratures empiriques [Yano and Patera, 2019].

B.3 Modèle 3D multi physique non linéaire

Un aimant à hauts champs implique plusieurs physiques couplées. D'abord, le courant électrique parcourant l'aimant produit de la chaleur par les pertes Joule. Les conductivités thermiques et électriques des alliages de cuivre composant l'aimant vont dépendre de la température. L'aimant va être refroidi par un écoulement forcé d'eau pour contrôler la température. La densité de courant va produire un champ magnétique. La dilatation thermique et les forces de Lorentz vont déformer l'aimant.

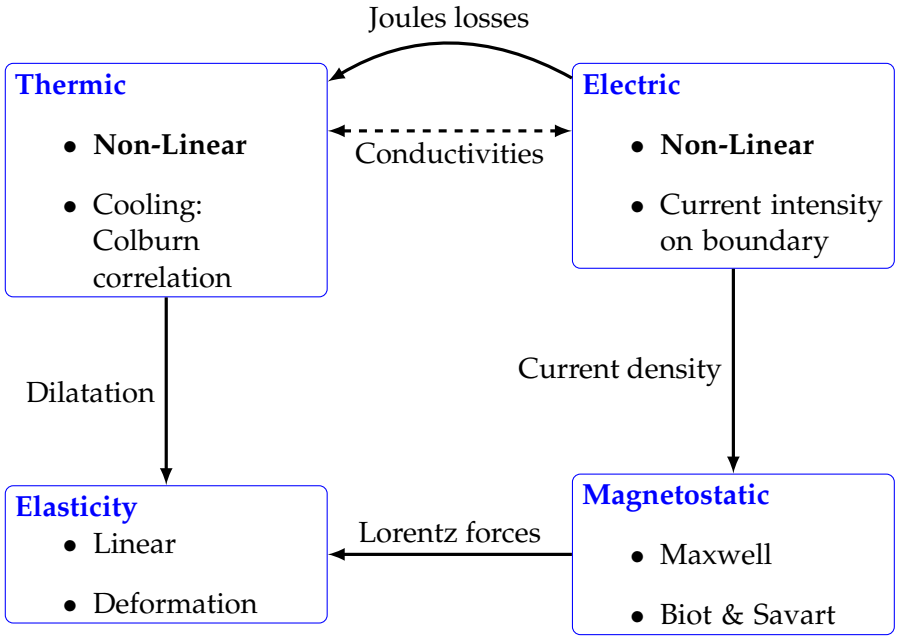


Figure B.4 – Coupled physics in a high field magnet

Modéliser directement un tel problème complexe étant très difficile, nous avons

choisi de faire quelques simplifications. D'abord, nous nous restreignons à un problème stationnaire. Nous simplifions ensuite le très coûteux problème de mécanique des fluides nécessaire pour le refroidissement de l'aimant en introduisant un coefficient d'échange thermique. Et nous supposons les déformations suffisamment petites pour pouvoir travailler sur la géométrie de référence. Ce modèle est présenté dans la figure B.4.

B.3.1 Modèle Electro-Thermique

Les équations de Maxwell-Faraday nous disent que le champ électrique $\mathbf{E}$ est proportionnel à la variation du champ magnétique $\mathbf{B}$ au cours du temps. Dans le cas statique, cela implique qu'il existe un potentiel scalaire V tel que $\mathbf{E} = -\nabla V$. De plus, la loi d'Ohm implique que la densité de courant $\mathbf{j}$ est proportionnelle au champ électrique: $\mathbf{j} = \sigma\mathbf{E}$, où σ est la conductivité électrique du matériau. En utilisant le principe de conservation de la charge dans le cas statique, on a:

$$\nabla \cdot (-\sigma \nabla V) = 0$$

Pour le problème thermique, nous avons comme source de chaleur l'effet Joule, qui peut être décrit comme $\mathbf{j} \cdot \mathbf{E} = \sigma \nabla V \cdot \nabla V$. D'où l'équation de la chaleur dans le cas statique:

$$-\nabla \cdot (\kappa \nabla T) = \sigma \nabla V \cdot \nabla V$$

Les conductivités thermique κ et électrique σ dépendent de la température, et donc le modèle couplé thermo-électrique est non linéaire.
La conductivité électrique décrit la capacité du matériau à transporter les charges électriques. Dans le cas de métaux et d'alliages, σ peut être exprimé en fonction de la température de référence T_0, la conductivité à cette température σ_0 et d'un coefficient de température α:

$$\sigma(T) = \frac{\sigma_0}{1 + \alpha(T - T_0)}$$

La loi de Wiedemann-Franz spécifie que le ratio entre les conductivités thermique et électrique est proportionnel à la température par une constante L nommée le nombre de Lorenz. D'où la conductivité thermique s'écrit:

$$\kappa(T) = \sigma(T) L T$$

Nous considérons l'eau coulant autour de l'aimant comme isolé électriquement. Nous ajoutons donc une condition de Neumann homogène sur ces surfaces, notées Γ_e:

- $-\sigma(T)\nabla V \cdot \mathbf{n} = 0$ sur Γ_e

Dans l'aimant, pour imposer la circulation du courant, une différence de potentiel est créée entre l'amenée et la sortie de courant. Nous pouvons modéliser cette différence de trois manières:

1. comme une condition de Dirichlet

- $V = 0$ sur Γ_{in}
- $V = V_D$ sur Γ_{out}

2. comme une condition de Neumann
   - $V = 0$ sur Γ_{in}
   - $-\sigma(T)\nabla V \cdot \mathbf{n} = \dfrac{I}{|\Gamma_{out}|}$ sur Γ_{out}
     où I est le courant imposé et $|\Gamma_{out}|$ est la surface de sortie
3. comme une condition intégrale
   - $V = 0$ sur Γ_{in}
   - $\displaystyle\int_{\Gamma_{out}} -\sigma(T)\nabla V \cdot \mathbf{n} = I$ et V est constant sur Γ_{out}

Pour modéliser le refroidissement de l'aimant par l'écoulement d'eau, nous supposons que le flux thermique est proportionnel à la différence entre la température de l'aimant et celle de l'eau. Le facteur est appelé le coefficient de transfert de chaleur et est noté h. Nous pouvons donc utiliser une condition de Robin pour modéliser le refroidissement sur ces surfaces Γ_c:

- $-\kappa(T)\nabla T \cdot \mathbf{n} = h(T - T_w)$ sur Γ_c

Le coefficient de transfert de chaleur peut être trouvé à partir de différentes corrélations, les plus utilisés sont celle de Colburn [Colburn, 1933] et Montgomery [Montgomery, 1969].

Sur les autres surfaces, on ne considère aucun transfert thermique. Nous utilisons donc des conditions de Neumann homogènes:

- $-\kappa(T)\nabla T \cdot \mathbf{n} = 0$ sur Γ_t

Le problème revient donc à trouver V et T tel que:

$$\begin{cases} \nabla \cdot (-\sigma(T)\nabla V) = 0 & \text{dans } \Omega \\ \nabla \cdot (\kappa(T)\nabla T) = \sigma(T)\nabla V \cdot \nabla V & \text{dans } \Omega \\ V = 0 & \text{sur } \Gamma_{in} \\ v = V_D & \text{sur } \Gamma_{out} \\ -\sigma(T)\nabla V \cdot \mathbf{n} = 0 & \text{sur } \Gamma_e \\ -\kappa(T)\nabla T \cdot \mathbf{n} = h(T - T_w) & \text{sur } \Gamma_c \\ -\kappa(T)\nabla T \cdot \mathbf{n} = 0 & \text{sur } \Gamma_t \end{cases}$$

Ce problème a été implémenté et résolu en utilisant les méthodes CG et HDG, ainsi que par la méthode des bases réduites pour obtenir des solutions pour différents paramètres.

B.3.2 Modèle Magnétostatique

Les équations de Maxwell décrivent la génération de champs électrique **E** et magnétique **H**, à l'aide des flux électrique **D** et magnétique **B**, la densité de courant **j** et la charge électrique ρ.

$$\begin{aligned} \nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} && \text{(Faraday)} \\ \nabla \times \mathbf{H} &= \mathbf{j} + \frac{\partial \mathbf{D}}{\partial t} && \text{(Maxwell-Ampère)} \\ \nabla \cdot \mathbf{B} &= 0 && \text{(Gauss magnetic law)} \\ \nabla \cdot \mathbf{D} &= \rho && \text{(Gauss electric law)} \end{aligned}$$

Dans le cas statique, les dérivées en temps ne sont pas considérées.

Ces équations doivent être complétées par les lois constitutives liant **B** à **H** et **D** et **E** à partir des propriétés matérielles.

$$\mathbf{B} = \mu \mathbf{H}, \quad \mathbf{D} = \varepsilon \mathbf{E}$$

où μ est la perméabilité magnétique, ε la permittivité magnétique.

En utilisant les lois magnétiques de Gauss pour définir le potentiel magnétique **A** tel que $\nabla \times \mathbf{A} = \mathbf{B}$ et celle de Maxwell Ampère, on obtient le problème magnétostatique:

$$\nabla \times \left(\frac{1}{\mu} \nabla \times \mathbf{A} \right) = \mathbf{j}$$

On considère les conditions aux bords classiques pour la magnétostatique, qui impose la composante tangentielle de potentiel magnétique:

$$\mathbf{A} \times \mathbf{n} = \mathbf{A}_D \text{ on } \partial\Omega$$

où **n** est la normale unité extérieure sur $\partial\Omega$. En général, $\mathbf{A}_D$ est défini à zéro sur $\partial\Omega$ pour imiter le comportement de A à l'infini.

Le problème magnétostatique n'a pas une unique solution à cause du potentiel **A** qui est défini à un gradient prêt. Pour garantir l'unicité de la solution, on utilise classiquement deux méthodes.
La première est d'ajouter une condition sur la divergence de **A**, en particulier la jauge de Coulomb $\nabla \cdot \mathbf{A} = 0$. La solution est ensuite trouvée en résolvant un problème de point-selle.
La seconde méthode consiste à considérer le problème comme un cas spécial du problème de Maxwell dans le domaine des fréquences.

Ces deux méthodes impliquent un grand coût de calcul, nécessitant des préconditionneurs spécifiques à ce type de problème. Une autre manière de déterminer le champ magnétique est d'utiliser la loi de Biot-Savart, permettant dans le cas courant statique **j** dans un conducteur Ω_C, de calculer **B** dans un domaine Ω_{mgn}. En voyant le problème magnétostatique comme une équation de Poisson et en utilisant la solution

générale de cette équation, la fonction de Green, on peut écrire le champ magnétique en un point **x** comme:

$$\mathbf{B}(\mathbf{x}) = \frac{\mu_0}{4\pi} \int_{\Omega_C} \frac{\mathbf{j}(\mathbf{r}) \times (\mathbf{x} - \mathbf{r})}{|\mathbf{x} - \mathbf{r}|^3} d\mathbf{r} \quad \forall \mathbf{x} \in \Omega_{mgn}$$

où μ_0 est la perméabilité dans le vide.

Une méthode pour calculer cette quantité en temps réel à partir d'une solution réduite du problème thermo-électrique a été développée et implémentée.

B.3.3 Modèle linéaire d'Élasticité

À pleine puissance, les contraintes à l'intérieur d'un aimant à haut champ atteignent 80% de la limite d'élasticité des matériaux. Le modèle élastique est donc très important pour assurer que l'intégrité de l'aimant. Le modèle utilisé est décrit dans [Slaughter and Petrolito, 2002].

Comme Ω_T est à l'équilibre, on a l'équation de l'équilibre suivant:

$$\nabla \cdot \bar{\bar{\sigma}} + \mathbf{f} = 0$$

où :

- $\bar{\bar{\sigma}}$ est le tenseur des contraintes
- **f** représente les forces volumiques sur Ω

Nous définissons le tenseur des petites déformations à partir du vecteur déplacement **u**:

$$\bar{\bar{\varepsilon}} = \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^T)$$

et utilisons la loi de Hooke permettant de relier le tenseur des contraintes $\bar{\bar{\sigma}}$ et le tenseur des petites déformations $\bar{\bar{\varepsilon}}$:

$$\bar{\bar{\sigma}}(\bar{\bar{\varepsilon}}) = \frac{E}{1+\nu}\left(\bar{\bar{\varepsilon}} + \frac{\nu}{1-2\nu} Tr(\bar{\bar{\varepsilon}}) I\right)$$

où :

- E est le module de Young
- ν est le ratio de Poisson
- I est le tenseur identité

En introduisant les coefficients de Lamé λ et μ:

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \qquad \text{et} \qquad \mu = \frac{E}{2(1+\nu)}$$

nous pouvons réécrire la loi de Hooke comme:

$$\bar{\bar{\sigma}}(\bar{\bar{\varepsilon}}) = 2\mu\bar{\bar{\varepsilon}} + \lambda Tr(\bar{\bar{\varepsilon}}) I$$

Pour prendre en compte la dilation thermique de l'aimant, nous devons ajouter un terme au tenseur des contraintes:

$$\bar{\bar{\sigma}}(\bar{\bar{\varepsilon}}) = \bar{\bar{\sigma}}_E(\bar{\bar{\varepsilon}}) + \bar{\bar{\sigma}}_T = 2\mu\bar{\bar{\varepsilon}} + \lambda Tr(\bar{\bar{\varepsilon}})I - \frac{E}{1-2\nu}\alpha_T(T - T_0)I$$

Pour compléter le système, nous utilisons les conditions aux bords:

- de Dirichlet là où l'aimant est attaché

$$\mathbf{u} = \mathbf{u}_D \quad \text{sur } \partial\Omega_D$$

- de Neumann pour imposer une pression

$$\bar{\bar{\sigma}} \cdot \mathbf{n} = \mathbf{g} \quad \text{sur } \partial\Omega_P$$

où $\partial\Omega = \partial\Omega_D \cup \partial\Omega_P$.

Pour déterminer si les déformations de l'aimant restent élastiques ou deviennent plastiques, les ingénieurs utilisent les critères de Tresca et de Von Mises.

Le critère de Tresca [Tresca, 1864] utilise le tenseur des contraintes diagonalisé $\bar{\bar{\sigma}}^d$ et est déterminé comme:

$$tr_{\bar{\bar{\sigma}}^d} = \max(|\bar{\bar{\sigma}}^d_{11} - \bar{\bar{\sigma}}^d_{22}|, |\bar{\bar{\sigma}}^d_{22} - \bar{\bar{\sigma}}^d_{33}|, |\bar{\bar{\sigma}}^d_{33} - \bar{\bar{\sigma}}^d_{11}|)$$

Le critère de Von Mises [Mises, 1913] peut être écrit en utilisant $\bar{\bar{\sigma}}$ ou $\bar{\bar{\sigma}}^d$:

$$vm_{\bar{\bar{\sigma}}} = \sqrt{\frac{1}{2}\left((\bar{\bar{\sigma}}_{11} - \bar{\bar{\sigma}}_{22})^2 + (\bar{\bar{\sigma}}_{22} - \bar{\bar{\sigma}}_{33})^2 + (\bar{\bar{\sigma}}_{33} - \bar{\bar{\sigma}}_{11})^2\right) + 3\left(\bar{\bar{\sigma}}^2_{12} + \bar{\bar{\sigma}}^2_{23} + \bar{\bar{\sigma}}^2_{31}\right)}$$

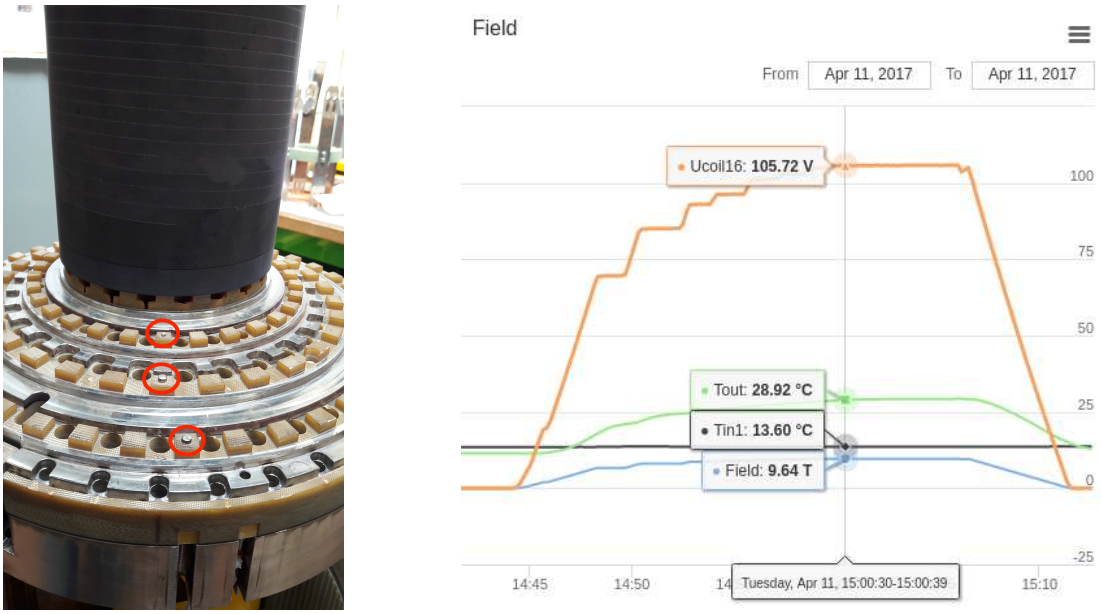
$$vm_{\bar{\bar{\sigma}}^d} = \sqrt{\frac{1}{2}\left((\bar{\bar{\sigma}}_{11} - \bar{\bar{\sigma}}_{22})^2 + (\bar{\bar{\sigma}}_{22} - \bar{\bar{\sigma}}_{33})^2 + (\bar{\bar{\sigma}}_{33} - \bar{\bar{\sigma}}_{11})^2\right)}$$

B.4 Applications

B.4.1 Identification de paramètres de refroidissement

Les aimants à hauts champs sont alimentés jusqu'à 31 kA pour délivrer 37 Teslas. Pour dissiper la chaleur produite, les aimants sont refroidis par un écoulement d'eau (jusqu'à 140 l/m^2). De tels environnements rendent difficile l'instrumentation des aimants, les seules mesures disponibles proviennent de:

- prises de tension sur les cercles de connexions entre les tubes consécutifs des aimants poly-hélices
- sondes de température à l'entrée et à la sortie du circuit de refroidissement
- capteurs de débit



(a) Tension sensors

(b) Logs of magnet monitoring system

Figure B.5 – Monitoring Magnets

Sur la figure B.5, les prises de tension disposées sur le plateau BP sont entourées en rouge, tandis que sur la figure B.5b est montrée la surveillance de l'aimant durant une expérience avec les différents capteurs.

Comme énoncé précédemment, le refroidissement est modélisé par des conditions de Robin du type $h(T - T_w)$. Ces paramètres, le coefficient d'échange thermique h et la température de l'eau T_w, ne sont pas connus précisément. Le but est de déterminer quels paramètres vont produire les plus petites erreurs entre les mesures expérimentales et simulées. Pour cela, nous considérons différentes corrélations pour le coefficient d'échange thermique, Montgomery [Montgomery, 1969], Colburn [Colburn, 1933], Dittus [Dittus and Boelter, 1985] et Silberberg [Silberberg et al., 2009].

Nous avons simulé une expérience menée en avril 2017 au LNCMI, un aimant de 14 hélices alimenté par un courant I_0 de 22148.2 A. Nous comparons les 14 prises de tension ainsi que deux mesures de températures.

Les différentes erreurs pour les prises de tension en utilisant les méthodes CG et HDG pour la corrélation de Dittus sont présentées dans l'image B.6.

On peut voir que dans les deux cas les erreurs ne dépassent pas 4%. Les plus grandes erreurs sur 3 hélices peuvent s'expliquer par des hélices ayant déjà servies et ayant possiblement été déformés.

Avant de mettre un aimant en service, deux expériences sont menées pour mesurer le facteur de champs $f = \mathbf{B}/I$ et la résistance des hélices ou couples d'hélices en fonction du courant. Ces mesures sont ensuite utilisées pour contrôler l'état de l'aimant durant les expériences, si les mesures divergent trop de celles de références, l'aimant risque d'être endommagé.

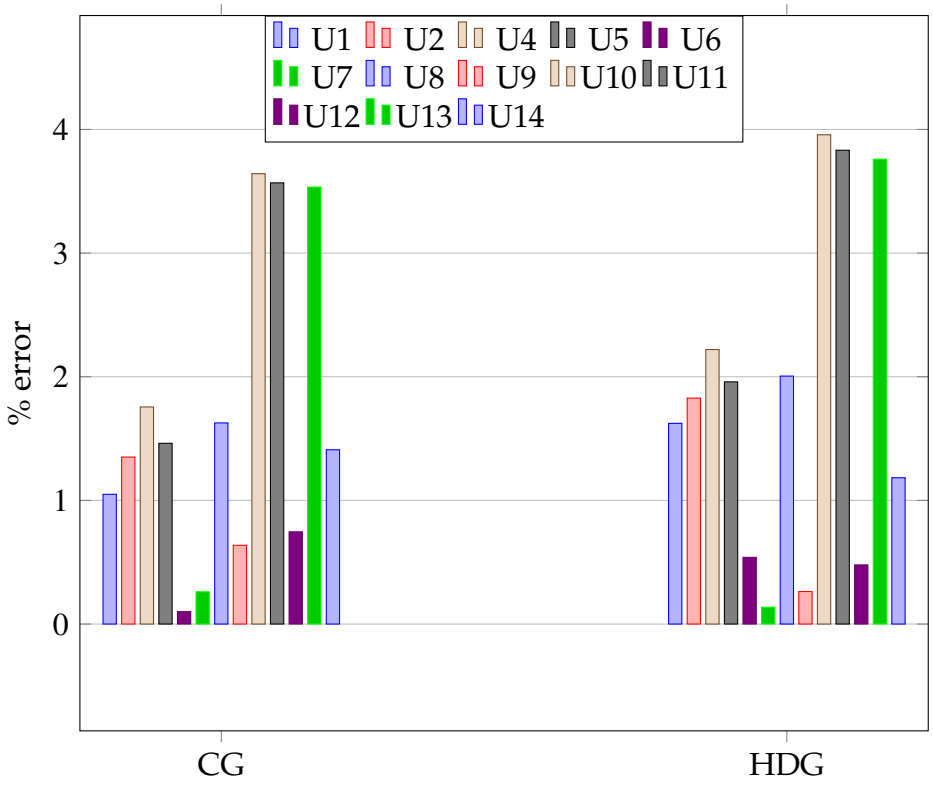


Figure B.6 – Comparison of the errors on the tension between CG and HDG

Nous avons essayé de reproduire ces opérations par la simulation. Pour le facteur de champs, la valeur du champ magnétique au centre de l'aimant est calculée en utilisant la loi de Biot-Savart.

Comme on peut le voir sur la figure B.7, les deux méthodes CG et HDG mènent à des erreurs inférieures à 0.5%, validant notre modèle.

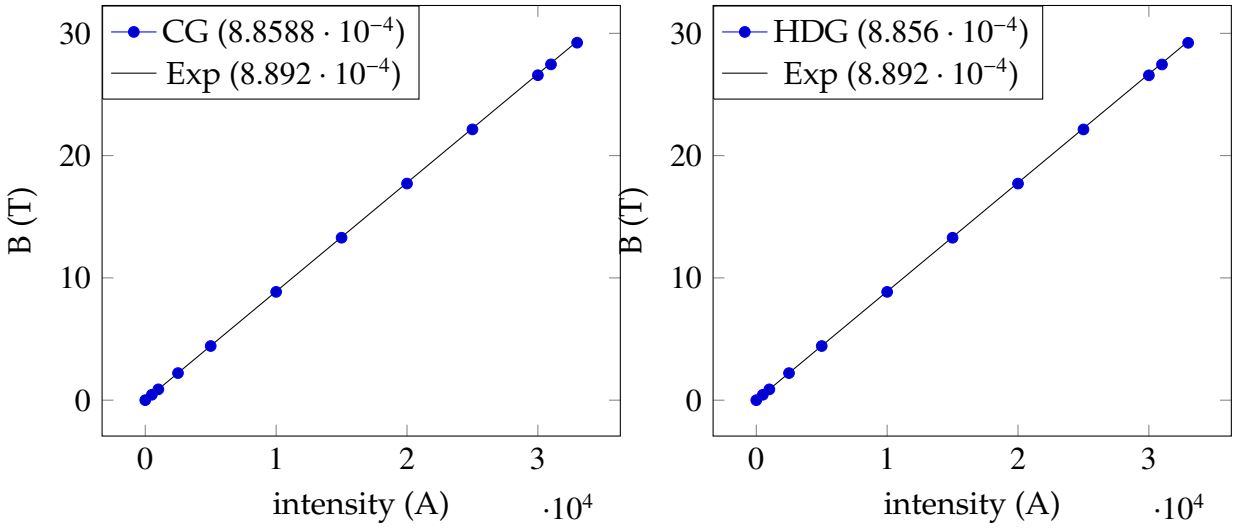
Les résultats concernant la résistance pour le premier et le dernier couple d'hélices sont présentés dans la figure B.8.

On peut voir que pour le premier couple, les valeurs expérimentales et simulées concordent correctement. Par contre le dernier couple ne correspond pas. Cela peut être dû à la taille de ces hélices qui sont bien plus grandes que les premières et mettent donc plus de temps à atteindre un plateau.

B.4.2 Optimisation géométrique

Pour certaines applications, les utilisateurs sont intéressés par un champ magnétique le plus homogène possible à l'intérieur de la zone d'expérience. L'homogénéité est définie comme $h = \frac{\max_{\mathbf{x}\in\Omega_{mgn}} \mathbf{B}_z(\mathbf{x})}{\min_{\mathbf{x}\in\Omega_{mgn}} \mathbf{B}_z(\mathbf{x})} - 1$ et est de l'ordre de 10^{-3} dans 1 cm^3 tandis que les utilisateurs cherchent à atteindre 10^{-6}.

Il est donc important pour le LNCMI d'avoir un champ le plus homogène possible dans la région d'intérêt. D'après une idée de C. Trophime et A. Janka, on cherche à



(a) CG (b) HDG

Figure B.7 – Field factor experimental vs numerical

modifier la forme des découpes de l'aimant en appliquant une transformation géométrique, et optimiser cette transformation pour avoir la meilleure homogénéité.

La forme de l'hélice est modifiée en utilisant une torsion angulaire, dont l'angle est contrôlé par une courbe de Bézier $\alpha(z)$. Les points de contrôle p_k de la courbe sont les paramètres de l'optimisation $\boldsymbol{\mu}$.

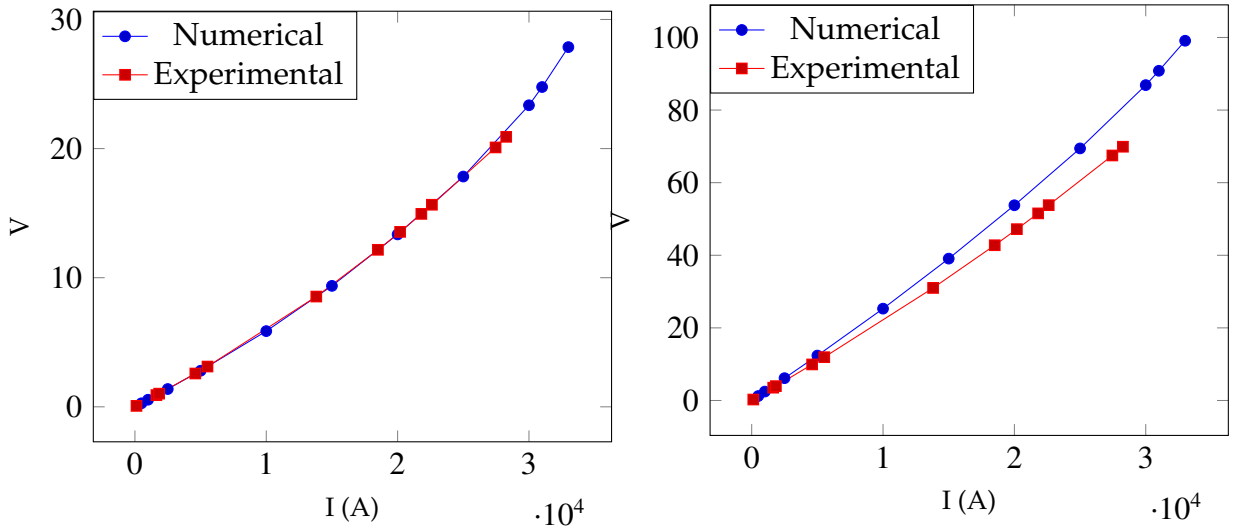
$$\tilde{\alpha}_{\boldsymbol{\mu}}(t) = \sum_{k=0}^{n} B_k^n(t) p_k = \sum_{k=0}^{n} C_n^k t^k (1-t)^{n-k} p_k \tag{B.1}$$

La transformation géométrique est alors écrite comme:

$$\phi_{\mu} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x\cos(\alpha_{\mu}(z)) + y\sin(\alpha_{\mu}(z)) \\ -x\sin(\alpha_{\mu}(z)) + y\cos(\alpha_{\mu}(z)) \\ z \end{pmatrix} \tag{B.2}$$

Nous allons nous concentrer sur un aimant à 4 hélices, avec une sphère en son centre où calculer le champ magnétique. Les deux hélices situées le plus à l'intérieur vont être modifiées pour maximiser l'homogénéité tandis que les deux autres serviront à fournir un champ de fond et ne seront pas optimisées.

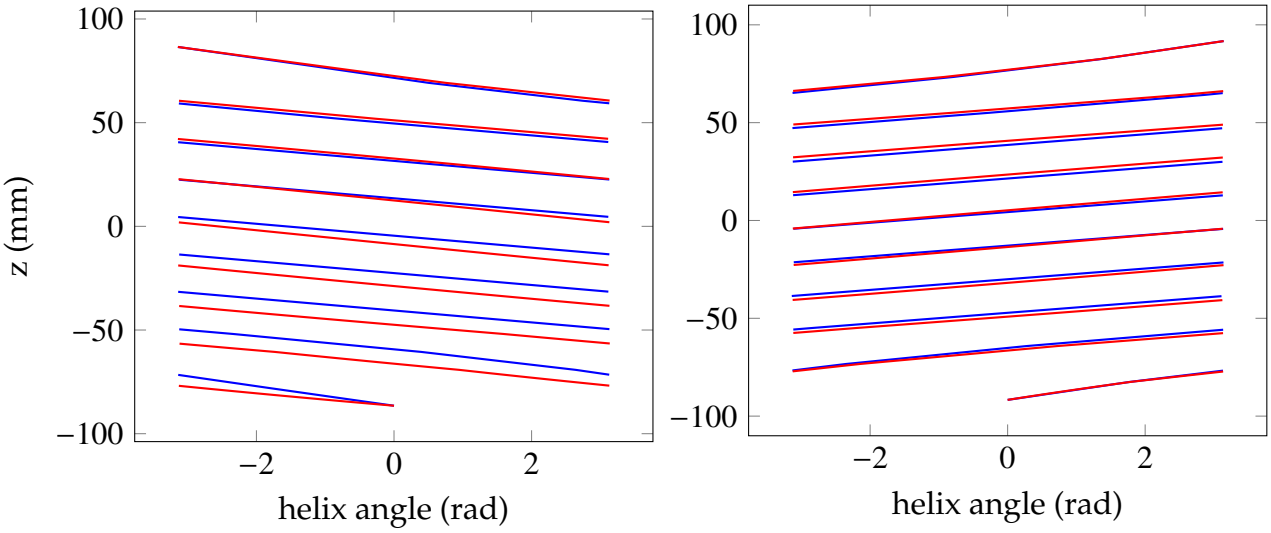
L'optimisation est effectuée à l'aide du logiciel NLopt [Johnson, Steven G.,]. La difficulté pour nous provient du fait que nous ne connaissons pas le gradient de la fonction objectif, et nous devons donc utiliser des algorithmes n'utilisant pas la dérivée qui sont moins efficaces.
La stratégie a été d'utiliser d'abord un algorithme global pour localiser de manière approximative le minimum, puis d'utiliser un algorithme local pour affiner cette location. En comptant les deux optimisations, le problème est résolu plus de 1200 fois, une résolution haute fidélité prenant plus de 120 secondes, et une résolution réduite prenant 10 secondes, cela représente plus de 36 heures de gagnées.



(a) CG (b) CG

Figure B.8 – Resistance of couple 1: experimental vs numerical

La figure 10.5 montre les différences entre les découpes initiales et optimales des hélices.



(a) Cuts of the first helix (b) Cuts of the second helix

Figure B.9 – Initial (blue) and optimal (red) cuts

Le tableau 10.6 résume les quantités initiales et optimales obtenues. On peut voir que la perte de l'intensité du champ magnétique est très limitée, moins de 0.001%, mais le minimum de l'intensité est beaucoup plus proche du maximum, entraînant une amélioration de l'homogénéité par un facteur 26.

	initial	optimal
$\min \lvert\mathbf{B}_z\rvert$	7.31521	7.33091
$\max \lvert\mathbf{B}_z\rvert$	7.33913	7.33181
$\frac{\max \lvert\mathbf{B}_z\rvert}{\min \lvert\mathbf{B}_z\rvert} - 1$	3.23698×10^{-3}	1.22054×10^{-4}

Figure B.10 – Initial and optimal homogeneity

B.5 Conclusion

Cette thèse a continué le travail déjà effectué dans le projet HifiMagnet par C. Daversin, dans le but de simplifier la maintenance des modèles existants et d'avoir des modèles plus précis et efficaces. Cet effort a été fait pour que les ingénieurs du LNCMI puissent utiliser ces méthodes dans leur travail pour construire de meilleurs aimants ou mieux comprendre leur fonctionnement.

La première partie de cette thèse a consisté à réécrire les modèles pour utiliser les dernières fonctionnalités de `Feel++` et de simplifier leurs configurations.
Nous avons par la suite implémenté la méthode HDG dans `Feel++` et créé des toolboxes pour les problèmes thermoélectrique et élastique. Cette méthode permet de contrôler les flux directement, comme la densité de courant, le champ magnétique ou le stresse, et ainsi avoir de meilleures propriétés physiques, comme la conservation du courant. En particulier, nous avons développé les conditions aux bords intégrales, nous permettant d'être le plus proche possible des conditions expérimentales du LNCMI. Pour faire nos calculs en parallèle et donc utiliser des géométries réelles, nous avons dû utiliser un partitionnement spécial ainsi que la condensation statique pour diminuer le coût de calcul.

Ensuite, nous voulions améliorer l'efficacité de nos simulations en utilisant des méthodes de réduction d'ordre. Nous avons choisi la méthode des bases réduites, qui nous offre la possibilité d'avoir une stratégie en ligne/hors ligne. Nous avons créé le modèle réduit pour le problème thermoélectrique et implémenté la réduction d'ordre pour Biot-Savart. Nous avons aussi aidé à l'implémentation de la version discrète de EIM dans `Feel++` qui nous permet de gérer simplement des paramètres géométriques complexes. Pour réduire encore le coût de calcul de Biot-Savart, nous avons implémenté la méthode de quadrature empirique. Toutes ces méthodes ont été combinées pour avoir un calcul en temps réel du champ magnétique dans un aimant à hauts champs.

Enfin, nous avons utilisé les méthodes à notre disposition dans deux applications qui peuvent être utilisées par le LNCMI pour améliorer les qualités des aimants produits ou faciliter leur utilisation. Nous avons confronté nos modèles aux données expérimentales du LNCMI, recherchant à identifier les paramètres de refroidissement correspondant aux expériences. Nous avons trouvé que les corrélations utilisées pour les coefficients d'échanges thermiques sont importantes, menant à des erreurs de moins de 4% pour CG et HDG dans certains cas. Nous avons aussi reproduit le facteur de champ et la résistance de l'aimant, utilisés pour contrôler l'intensité nécessaire et détecter les anomalies. L'utilisation de la réduction d'ordre pour encore optimiser ces

paramètres a été compliquée par la taille des problèmes et le nombre de paramètres, mais des stratégies alternatives sont implémentées pour résoudre ce problème.
La seconde application consiste à optimiser les découpes de l'aimant pour obtenir une meilleure homogénéité. Cela implique l'utilisation de paramètres géométriques, menant à l'exploitation de toutes les méthodes présentées (RB, DEIM, EQM, SER) pour réduire le coût de l'optimisation. Les résultats sont prometteurs et demandent d'utiliser cette méthode avec plus de paramètres pour obtenir une amélioration significative de l'homogénéité.

Pour la suite du projet HifiMagnet, de nouvelles méthodes et améliorations vont être développées pour aider le LNCMI à l'aide des outils de simulations.

D'abord, d'un point de vue mathématique, on pourrait ajouter le modèle HDG pour les équations de Maxwell, permettant le calcul de toutes les physiques avec HDG. HDG pourrait aussi être utilisé avec les bases réduites, donnant la possibilité d'avoir le même contrôle sur les flux que dans le modèle haute fidélité. De même, les problèmes magnétostatique et d'élasticité pourraient être réduits par la méthode RB, ouvrant la possibilité à d'autres applications nécessitant des calculs en temps réel.

La modélisation du problème peut aussi être améliorée, d'abord en utilisant un modèle dynamique pour voir l'évolution dans le temps des différentes quantités d'intérêt. Un stage est en cours sur ce sujet. Ensuite, nous pourrions modéliser complètement le refroidissement de l'aimant, en résolvant les équations de Navier-Stokes pour l'écoulement de l'eau. Pour être plus précis, nous devrions aussi utiliser une carte ALE pour prendre en compte la déformation de l'aimant.

Enfin, les méthodes développées durant le projet HifiMagnet devraient être plus intégrées pour les ingénieurs du LNCMI, avec l'utilisation du portail MSO4SC pour lancer des simulations dans le cloud depuis une station de travail. Les méthodes de réduction de modèle devraient aussi être accessibles à travers la bibliothèque MOR_DICUS, un projet FUI qui a pour but d'industrialiser de telles méthodes.
Et bien sûr, le projet devrait continuer à développer d'autres applications d'intérêt pour le LNCMI.

Bibliography

[Johnson, Steven G.,] Johnson, Steven G. The nlopt nonlinear-optimization package. `http://github.com/stevengj/nlopt`.

[Amestoy et al., 2000] Amestoy, P., Duff, I., and L'Excellent, J.-Y. (2000). Multifrontal parallel distributed symmetric and unsymmetric solvers. *Computer Methods in Applied Mechanics and Engineering*, 184(2):501 – 520.

[Arnold and Brezzi, 1985] Arnold, D. and Brezzi, F. (1985). Mixed and nonconforming finite element methods: implementation, postprocessing and error estimates. *Math. Modeling and Numer. Anal.*, 19(1):7–32.

[Arnold et al., 2002] Arnold, D. N., Brezzi, F., Cockburn, B., and Marini, L. D. (2002). Unified analysis of discontinuous galerkin methods for elliptic problems. *SIAM Journal on Numerical Analysis*, 39(5):1749–1779.

[Assous and Michaeli, 2013] Assous, F. and Michaeli, M. (2013). A numerical method for handling boundary and transmission conditions. *Mathematical and Computer Modelling*, 58(1):393 – 402. Financial IT & Security and 2010 International Symposium on Computational Electronics.

[Babuška, 1971] Babuška, I. (1971). Error-bounds for finite element method. *Numer. Math.*, 16(4):322–333.

[Baggag et al., 1999] Baggag, A., Atkins, H., and Keyes, D. (1999). Parallel implementation of the discontinuous galerkin method. Technical report.

[Balay et al., 2018] Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., May, D. A., McInnes, L. C., Mills, R. T., Munson, T., Rupp, K., Sanan, P., Smith, B. F., Zampini, S., Zhang, H., and Zhang, H. (2018). PETSc Web page. `http://www.mcs.anl.gov/petsc`.

[Balmès, 1996] Balmès, E. (1996). Parametric families of reduced finite element models. theory and applications. *Mechanical Systems and Signal Processing*, 10(4):381–394.

[Barrault et al., 2004] Barrault, M., Maday, Y., Nguyen, N. C., and Patera, A. T. (2004). An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667 – 672.

[Barrenechea et al., 2018] Barrenechea, G. R., Dolean, V., Nataf, F., and Tournier, P.-H. (2018). Hybrid discontinuous Galerkin discretisation and domain decomposition preconditioners for the Stokes problem. *Computational Methods in Applied Mathematics*.

[Bebendorf and Ostrowski, 2009] Bebendorf, M. and Ostrowski, J. (2009). Parallel hierarchical matrix preconditioners for the curl-curl operator. *Journal of Computational Mathematics - J COMPUT MATH*, 27:624–641.

[Berger and Colella, 1989] Berger, M. and Colella, P. (1989). Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82(1):64 – 84.

[Bey et al., 1996] Bey, K. S., Oden, J. T., and Patra, A. (1996). A parallel hp-adaptive discontinuous galerkin method for hyperbolic conservation laws. *Applied Numerical Mathematics*, 20(4):321 – 336. Adaptive mesh refinement methods for CFD applications.

[Boffi et al., 2013] Boffi, D., Brezzi, F., and Fortin, M. (2013). *Mixed Finite Element Methods and Applications*, volume 44. Springer Series in Computational Mathematics.

[Brezzi et al., 1985] Brezzi, F., Douglas, J., and Marini, L. D. (1985). Two families of mixed finite elements for second order elliptic problems. *Numerische Mathematik*, 47:217–235.

[Buffa, Annalisa et al., 2012] Buffa, Annalisa, Maday, Yvon, Patera, Anthony T., Prud´homme, Christophe, and Turinici, Gabriel (2012). A priori convergence of the greedy algorithm for the parametrized reduced basis method. *ESAIM: M2AN*, 46(3):595–603.

[Canuto et al., 2006] Canuto, C., Hussaini, M., Quarteroni, A., and A. Zang, T. (2006). *Spectral Methods: Fundamentals in Single Domains*, volume 23.

[Chaturantabut and Sorensen, 2010] Chaturantabut, S. and Sorensen, D. (2010). Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764.

[Chen et al., 2017] Chen, H., Qiu, W., Shi, K., and Solano, M. (2017). A superconvergent hdg method for the maxwell equations. *Journal of Scientific Computing*, 70(3):1010–1029.

[Chinesta et al., 2011] Chinesta, F., Ladeveze, P., and Cueto, E. (2011). A short review on model order reduction based on proper generalized decomposition. *Archives of Computational Methods in Engineering*, 18(4):395.

[Ciarlet, 1978] Ciarlet, P. (1978). The finite element method for elliptic problems, vol. 40 of classics in applied mathematics, society for industrial and applied mathematics (siam), philadelphia, pa, 2002. reprint of the 1978 original. *Reprint of the 1978 original*, 1(1):2–4.

[Cockburn, 2010] Cockburn, B. (2010). The hybridizable Discontinuous Galerkin method. In *Proceedings of the International Congress of Mathematicians*.

[Cockburn et al., 2008] Cockburn, B., Dong, B., and Guzmán, J. (2008). A superconvergent LDG-hybridizable galerkin method for second-order elliptic problems. *Math. Comp.*, 77(264):1887–1916.

[Cockburn et al., 2009a] Cockburn, B., Dong, B., Guzmán, J., Restelli, M., and Sacco, R. (2009a). A hybridizable Discontinuous Galerkin method for steady-state convection-diffusion-reaction problems. *SIAM Journal on Scientific Computing*, 31(5):3827–3846.

[Cockburn et al., 2009b] Cockburn, B., Gopalakrishnan, J., and Lazarov, R. (2009b). Unified hybridization of discontinuous galerkin, mixed, and continuous galerkin methods for second order elliptic problems. *SIAM Journal on Numerical Analysis*, 47(2):1319–1365.

[Cockburn et al., 2009c] Cockburn, B., Guzmán, J., and Wang, H. (2009c). Superconvergent discontinuous galerkin methods for second-order elliptic problems. *Math. Comput.*, 78:1–24.

[Cockburn et al., 1990] Cockburn, B., Hou, S., and Shu, C.-W. (1990). The runge-kutta local projection discontinuous galerkin finite element method for conservation laws. iv: The multidimensional case. *Mathematics of Computation*, 54(190):545–581.

[Cockburn et al., 2011] Cockburn, B., Karniadakis, G. E., and Shu, C.-W. (2011). *Discontinuous Galerkin Methods: Theory, Computation and Applications*. Springer Publishing Company, Incorporated, 1st edition.

[Cockburn et al., 1989] Cockburn, B., Lin, S.-Y., and Shu, C.-W. (1989). Tvb runge-kutta local projection discontinuous galerkin finite element method for conservation laws iii: One-dimensional systems. *Journal of Computational Physics*, 84(1):90 – 113.

[Cockburn et al., 2012] Cockburn, B., Qiu, W., and Shi, K. (2012). Conditions for superconvergence of hdg methods for second-order elliptic problems. *Mathematics of Computation*, 81:1327–1353.

[Cockburn and Shu, 1989] Cockburn, B. and Shu, C.-W. (1989). Tvb runge-kutta local projection discontinuous galerkin finite element method for conservation laws ii: General framework. *Mathematics of Computation*, 52(186):411–435.

[Cockburn and Shu, 1991] Cockburn, B. and Shu, C.-W. (1991). The runge-kutta local projection p^1-discontinuous-galerkin finite element method for scalar conservation laws. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 25(3):337–361.

[Cockburn and Shu, 1998] Cockburn, B. and Shu, C.-W. (1998). The runge–kutta discontinuous galerkin method for conservation laws v: Multidimensional systems. *Journal of Computational Physics*, 141(2):199 – 224.

[Colburn, 1933] Colburn, A. (1933). A method of correlating forced convection heat transfer data and a comparison with fluid friction. *Trans. AIChE*, 29:174–210.

[Courant, 1943] Courant, R. (1943). Variational methods for the solution of problems of equilibrium and vibrations. *Bull. Amer. Math. Soc.*, 49:1–23.

[Daversin et al., 2016a] Daversin, C., Hild, R., Prud'Homme, C., and Trophime, C. (2016a). Full 3d non-linear multi-physics model for high field polyhelix magnets. *10th International Symposium on Electric and Magnetic Fields, Lyon, France*.

[Daversin and Prud'Homme, 2015] Daversin, C. and Prud'Homme, C. (2015). Simultaneous Empirical Interpolation and Reduced Basis method for non-linear problems. *Comptes Rendus Mathématique*.

[Daversin et al., 2016b] Daversin, C., Prudhomme, C., and Trophime, C. (2016b). Full three-dimensional multiphysics model of high-field polyhelices magnets. *IEEE transactions on applied superconductivity*, 26(4).

[Daversin, C. et al., 2013] Daversin, C., Veys, S., Trophime, C., and Prud´homme, C. (2013). A reduced basis framework: Application to large scale non-linear multi-physics problems. *ESAIM: Proc.*, 43:225–254.

[Daversin Catty, 2016] Daversin Catty, C. (2016). *Reduced basis method applied to large non-linear multi-physics problems. Application to high field magnets design*. Theses, IRMA (UMR 7501).

[Debray et al., 2012] Debray, F., Dumas, J., Trophime, C., and Vidal, N. (2012). Dc high field magnets at the lncmi. *IEEE transactions on applied superconductivity*, 22(3).

[Debray et al., 2002] Debray, F., Jongbloets, H., Joss, W., Martinez, G., Mossang, E., Petmezakis, P., Picoche, J., Plante, A., Rub, P., Sala, P., and Wyder, P. (2002). The grenoble high magnetic field laboratory as a user facility. *IEEE transactions on applied superconductivity,*, 12(1).

[DeVore et al., 2012] DeVore, R., Petrova, G., and Wojtaszczyk, P. (2012). Greedy algorithms for reduced bases in banach spaces. *Constructive Approximation*, 37.

[Dittus and Boelter, 1985] Dittus, F. and Boelter, L. (1985). Heat transfer in automobile radiators of the tubular type. *International Communications in Heat and Mass Transfer*, 12(1):3 – 22.

[Dumitru, 2013] Dumitru, C. (2013). Numerical problems in 3d magnetostatic fem analysis. In *Advances in Automatic Control, Modelling & Simulation*. Proceeding of the 15th International Conference on Automatic Control, Modelling & Simulation.

[Feynman et al., 2011] Feynman, R., Leighton, R., and Sands, M. (2011). *The Feynman Lectures on Physics, Vol. II: The New Millennium Edition: Mainly Electromagnetism and Matter*. Feynman Lectures on Physics. Basic Books.

[Fink and Rheinboldt, 1983] Fink, J. and Rheinboldt, W. (1983). On the error behavior of the reduced basis technique for nonlinear finite element approximations. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 63(1):21–28.

[Freund and Stenberg, 1995] Freund, J. and Stenberg, R. (1995). On weakly imposed boundary conditions for second order problems. pages 327–336.

[Geuzaine and Remacle, 2009] Geuzaine, C. and Remacle, J. F. (2009). Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*.

[GLPK, 2020] GLPK (2020). Gnu linear programming kit. `http://www.gnu.org/software/glpk/glpk.html`.

[Greif and Schötzau, 2007] Greif, C. and Schötzau, D. (2007). Preconditioners for the discretized time-harmonic maxwell equations in mixed form. *Numerical Linear Algebra with Applications*, 14:281–297.

[Grepl, Martin A. et al., 2007] Grepl, Martin A., Maday, Yvon, Nguyen, Ngoc C., and Patera, Anthony T. (2007). Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. *ESAIM: M2AN*, 41(3):575–605.

[Grisvard, 1985] Grisvard, P. (1985). *Elliptic problems in non smooth domains*. Number 24 in Monographs and studies in Mathematics. Pitman.

[Guidoboni et al., 2020] Guidoboni, G., Hild, R., Prada, D., Prud'homme, C., Sacco, R., Sala, L., and Szopos, M. (2020). An implementation of hdg methods with feel++. application to problems with integral boundary conditions. to appear.

[Hesthaven et al., 2015] Hesthaven, J. S., Rozza, G., and Stamm, B. (2015). *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. SpringerBriefs in Mathematics. Springer International Publishing.

[Hiptmair and Xu, 2007] Hiptmair, R. and Xu, J. (2007). Nodal auxiliary space preconditioning in h(curl) and h(div) spaces. *SIAM Journal on Numerical Analysis*, 45:2483–2509.

[Hrennikoff, 1941] Hrennikoff, A. (1941). Solution of Problems of Elasticity by the Framework Method. *Journal of Applied Mechanics*, 8(4):0–0.

[Huynh et al., 2007] Huynh, D., Rozza, G., Sen, S., and Patera, A. (2007). A successive constraint linear optimization method for lower bounds of parametric coercivity and inf–sup stability constants. *Comptes Rendus Mathematique*, 345(8):473 – 478.

[Jackson, 1999] Jackson, J. D. (1999). *Classical electrodynamics*. Wiley, New York, NY, 3rd ed. edition.

[JONES et al.,] JONES, D. R., PERTTUNEN, C. D., and STUCKMAN, B. E. *JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS*.

[Kabaria et al., 2014] Kabaria, H., J. Lew, A., and Cockburn, B. (2014). A hybridizable discontinuous galerkin formulation for non-linear elasticity. 283.

[Kerschen et al., 2005] Kerschen, G., Golinval, J.-c., VAKAKIS, A. F., and BERGMAN, L. A. (2005). The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: An overview. *Nonlinear Dynamics*, 41(1):147–169.

[Kirby et al., 2012] Kirby, R., Sherwin, S., and Cockburn, B. (2012). To cg or to hdg: A comparative study. *Journal of Scientific Computing*, 51(1):183–212.

[Kolmogoroff, 1936] Kolmogoroff, A. (1936). Uber die beste annaherung von funktionen einer gegebenen funktionenklasse. *Annals of Mathematics*, 37:107.

[Kratz and Wyder, 2002] Kratz, R. and Wyder, P. (2002). *Principles of pulsed magnet design*. Engineering materials. Springer, Berlin.

[Lax and Milgram, 1954] Lax, P. D. and Milgram, A. N. (1954). Parabolic equations. In *Contributions to the theory of partial differential equations*, Annals of Mathematics Studies, no. 33, pages 167–190. Princeton University Press, Princeton, N. J.

[Lu et al., 2015] Lu, P., Chen, H., and Qiu, W. (2015). An absolutely stable *hp*-HDG method for the time-harmonic Maxwell equations with high wave number. *ArXiv e-prints*.

[Machiels et al., 2000] Machiels, L., Maday, Y., Oliveira, I. B., Patera, A. T., and Rovas, D. V. (2000). Output bounds for reduced-basis approximations of symmetric positive definite eigenvalue problems. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 331(2):153 – 158.

[Maday et al., 2008] Maday, Y., Nguyen, N., T. Patera, A., and Pau, G. S. H. (2008). A general multipurpose interpolation procedure: The magic points. *Communications on Pure and Applied Analysis*, 8.

[Mises, 1913] Mises, R. v. (1913). Mechanik der festen körper im plastisch- deformablen zustand. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, 1913:582–592.

[Montgomery, 1969] Montgomery, D. (1969). *Solenoid magnet design: the magnetic and mechanical aspects of resistive and superconducting systems*. Solenoid magnet design. Wiley-Interscience.

[Nédélec, 1980] Nédélec, J. (1980). Mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik*, 35:315–341.

[Nédélec, 1986] Nédélec, J. (1986). A new family of mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik*, 50:57–81.

[Ngoc Cuong et al., 2005] Ngoc Cuong, N., Veroy, K., and Patera, A. T. (2005). *Certified Real-Time Solution of Parametrized Partial Differential Equations*, pages 1529–1564. Springer Netherlands, Dordrecht.

[Nguyen et al., 2010] Nguyen, N., Peraire, J., and Cockburn, B. (2010). A hybridizable discontinuous galerkin method for stokes flow. 199:582–597.

[Nguyen et al., 2011] Nguyen, N., Peraire, J., and Cockburn, B. (2011). An implicit high-order hybridizable discontinuous galerkin method for the incompressible navier-stokes equations. *J. Comput. Phys.*, 230(4):1147–1170.

[Nicolaides, 1982] Nicolaides, R. (1982). Existence, uniqueness and approximation for generalized saddle point problems. *SIAM Jour. on Numer. Anal.*, 19(2):349–357.

[Nitsche, 1971] Nitsche, J. (1971). Über ein variationsprinzip zur lösung von dirichlet-problemen bei verwendung von teilräumen, die keinen randbedingungen unter-

worfen sind. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 36(1):9–15.

[Noor, 1981] Noor, A. K. (1981). Recent advances in reduction methods for nonlinear problems. *Computers & Structures*, 13(1):31–44.

[Patera and Rozza, 2007] Patera, A. and Rozza, G. (2007). Reduced basis approximation and a posteriori error estimation for parametrized partial differential equations. MIT Pappalardo Graduate Monographs in Mechanical Engineering. Copyright MIT 2006-2007.

[Peterson, 1989] Peterson, J. S. (1989). The reduced basis method for incompressible viscous flow calculations. *SIAM Journal on Scientific and Statistical Computing*, 10(4):777–786.

[Porsching and Lee, 1987] Porsching, T. A. and Lee, M. L. (1987). The reduced basis method for initial value problems. *SIAM Journal on Numerical Analysis*, 24(6):1277–1287.

[Powell, 1994] Powell, M. J. D. (1994). *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*, pages 51–67. Springer Netherlands, Dordrecht.

[Prud'Homme et al., 2012] Prud'Homme, C., Chabannes, V., Doyeux, V., Ismail, M., Samake, A., and Pena, G. (2012). Feel++: A Computational Framework for Galerkin Methods and Advanced Numerical Methods.

[Prud'homme and Patera, 2004] Prud'homme, C. and Patera, A. (2004). Reduced-basis output bounds for approximately parameterized elliptic coercive partial differential equations. *Computing and Visualization in Science*, 6(2-3):147–162.

[Prud'Homme et al., 2001] Prud'Homme, C., Rovas, D. V., Veroy, K., Machiels, L., Maday, Y., Patera, A. T., and Turinici, G. (2001). Reliable Real-Time Solution of Parametrized Partial Differential Equations: Reduced-Basis Output Bound Methods. *Journal of Fluids Engineering*, 124(1):70–80.

[Qiu et al., 2013] Qiu, W., Shen, J., and Shi, K. (2013). An HDG method for linear elasticity with strong symmetric stresses. *ArXiv e-prints*.

[Quarteroni et al., 2011] Quarteroni, A., Rozza, G., and Manzoni, A. (2011). Certified reduced basis approximation for parametrized partial differential equations and applications. *Journal of Mathematics in Industry*, 1(1):1–49.

[Raviart and Thomas, 1977] Raviart, P.-A. and Thomas, J. M. (1977). Primal hybrid finite element methods for 2nd order elliptic equations. *Mathematics of computation*, 31(138):391–413.

[Reed and Hill, 1973] Reed, W. and Hill, T. (1973). Triangular mesh methods for the neutron transport equation.

[Remacle et al., 2003] Remacle, J.-F., E. Flaherty, J., and Shephard, M. (2003). An adaptive discontinuous galerkin technique with an orthogonal basis applied to compressible flow problems. *SIAM review*, 45:53–72.

[Rheinboldt, 1992] Rheinboldt, W. C. (1992). On the theory and error estimation of the reduced basis method for multi-parameter problems. Technical report, DTIC Document.

[Ribes et al., 2017] Ribes, A., Bruneton, A., and Geay, A. (2017). Salome: an open-source simulation platform integrating paraview.

[Rozza et al., 2007] Rozza, G., Huynh, D., and Patera, A. (2007). Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *Archives of Computational Methods in Engineering*, 15(3):1–47.

[Silberberg et al., 2009] Silberberg, Y., Lahini, Y., Bromberg, Y., Small, E., and Morandotti, R. (2009). Universal correlations in a nonlinear periodic 1d system. *Physical Review Letters*, 102(23).

[Slaughter and Petrolito, 2002] Slaughter, W. and Petrolito, J. (2002). The linearized theory of elasticity. *Applied Mechanics Reviews - APPL MECH REV*, 55.

[Solin et al., 2003] Solin, P., Segeth, K., and Dolezel, I. (2003). *Higher-Order Finite Element Methods*. Studies in Advanced Mathematics. Taylor & Francis.

[Strang and Fix, 1973] Strang, W. and Fix, G. (1973). *An analysis of the finite element method*. Prentice-Hall series in automatic computation. Prentice-Hall.

[Thomée, 2006] Thomée, V. (2006). *Galerkin Finite Element Methods for Parabolic Problems (Springer Series in Computational Mathematics)*. Springer-Verlag, Berlin, Heidelberg.

[Toth and Bole, 2018] Toth, J. and Bole, S. T. (2018). Design, construction, and first testing of a 41.5 t all-resistive magnet at the nhmfl in tallahassee. *IEEE Transactions on Applied Superconductivity*, 28(3):1–4.

[Tresca, 1864] Tresca, H. (1864). *Mémoire sur l'écoulement des corps solides soumis à de fortes pressions*. Gauthier-Villars.

[Trophime et al., 2002] Trophime, C., Egorov, K., Debray, F., Joss, W., and Aubert, G. (2002). Magnet calculations at the grenoble high magnetic field laboratory. *IEEE transactions on applied superconductivity,*, 12(1).

[Vallaghé et al., 2011] Vallaghé, S., Fouquembergh, M., Le Hyaric, A., and Prud'Homme, C. (2011). A successive constraint method with minimal offline constraints for lower bounds of parametric coercivity constant. working paper or preprint.

[Veroy et al., 2003] Veroy, K., Prud'homme, C., and Patera, A. (2003). Reduced-basis approximation of the viscous Burgers equation: Rigorous *a posteriori* error bounds. *C. R. Acad. Sci. Paris, Série I*, 337(9):619–624.

[Veroy, Karen et al., 2002] Veroy, Karen, Rovas, Dimitrios V., and Patera, Anthony T. (2002). A posteriori error estimation for reduced-basis approximation of parametrized elliptic coercive partial differential equations: "convex inverse" bound conditioners. *ESAIM: COCV*, 8:1007–1028.

[Veys, 2014] Veys, S. (2014). *A computational reduced basis framework : applications to nonlinears multiphysics problems*. Theses, Université de Grenoble.

[Wahl, 2018] Wahl, J.-B. (2018). *The Reduced basis method applied to aerothermal simulations*. Theses, Université de Strasbourg.

[Wilson, 1987] Wilson, M. (1987). *Superconducting Magnets*. Monographs on Cryogenics. Clarendon Press.

[Yano and Patera, 2019] Yano, M. and Patera, A. T. (2019). An lp empirical quadrature procedure for reduced basis treatment of parametrized nonlinear pdes. *Computer Methods in Applied Mechanics and Engineering*, 344:1104 – 1123.

Résumé: Nous présentons dans cette thèse notre travail sur le contrôle et l'optimisation d'aimants à hauts champs. Les physiques impliquées sont présentées et leur discrétisation est détaillée. Elles consistent en un problème thermoelectrique non linéaire, un problème magnétostatique et un problème d'élasticité linéaire. La méthode de Galerkin Discontinu Hybrid (HDG) est utilisée pour approcher au mieux les champs d'intérêt, tels que la densité de courant, le champ magnétique ou le tenseur des contraintes. Nous avons développé et implémenté les Conditions Intégrales aux Bords (IBC) pour pouvoir imposer l'intensité de courant directement au lieu d'utiliser la différence de potentiel. Pour résoudre notre problème en temps réel, nous avons utilisé la méthode des Bases Réduites (RB), combinée avec la Méthode d'Interpolation Empirique (EIM), sa version discrète, la méthode Simultanée EIM et RB (SER) et la Méthode de Quadrature Empirique (EQM). Finalement, nous avons appliqué ces méthodes à deux applications d'intérêt pour le LNCMI, l'identification des paramètres de refroidissement basé sur des données expérimentales, et l'optimisation des découpes de l'aimant pour améliorer son homogénéité.
Mots-clés: Réduction d'ordre de modèle, Aimants à haut champs, Méthode des Bases Réduites, Méthode d'Interpolation Empirique, Méthode Eléments Finis, Galerkin Discontinu Hybrid, Calcul Haute Performance, `Feel++`

Summary: We present in this thesis our work on the control and optimization of high field magnets. The physics involved in the operation of the magnet are presented, and their discretization is detailed. It consists of a non-linear thermoelectric problem, a magnetostatic problem and a linear elasticity problem. The Hybrid Discontinuous Galerkin (HDG) method is used in order to better approximate the fields of interests, such as the current density, the magnetic field or the stress. We developed and implemented the Integral Boundary Condition (IBC) to be able to impose the current intensity directly instead of using the difference of potential. To solve our problem in real time, we used the Reduce Basis method (RB), combined with the Empirical Interpolation Method (EIM), its discrete version, the Simultaneous EIM and RB method and the Empirical Quadrature Method (EQM). Finally, we applied our methods to two applications of interest for the LNCMI, the identification of cooling parameters based on experimental data, and the optimization of the cuttings of the magnets to improve its homogeneity.
Keywords: Model Order Reduction, High Field Magnets, Reduced Basis Method, Empirical Interpolation Method, Finite Elements Method, Hybrid Discontinuous Galerkin, High Performance Computing, `Feel++`

INSTITUT DE RECHERCHE MATHÉMATIQUE AVANCÉE
UMR 7501
Université de Strasbourg
CNRS
IRMA, UMR 7501
7 Rue René Descartes
F-67000 STRASBOURG

Tél. 03 68 85 01 29

irma.math.unistra.fr
irma@math.unistra.fr

IRMA 2020/011
https://tel.archives-ouvertes.fr/tel-03025312