The Reduced Basis Method Applied to Aerothermal Simulations

Jean-Baptiste Wahl

Thèse

INSTITUT DE RECHERCHE MATHÉMATIQUE AVANCÉE

UMR 7501

Strasbourg

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

Jean-Baptiste Wahl

The Reduced Basis Method Applied to Aerothermal Simulations

Soutenue le 13 Septembre 2018
devant la commission d'examen

Martin Grepl, rapporteur
Anthony Nouy, rapporteur
Fabian Mangeant, examinateur
Robert Mosé, examinateur
Christophe Prud'homme, directeur de thèse
Yannick Hoarau, co-directeur de thèse

www-irma.u-strasbg.fr

cnrs dépasser les frontières

IRMA Institut de Recherche Mathématique Avancée

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

Jean-Baptiste WAHL

soutenue le 13 septembre 2018

pour obtenir le grade de Docteur de l'Université de Strasbourg

Spécialité : Mathématiques Appliquées

The Reduced Basis Method Applied to Aerothermal Simulations

THÈSE DIRIGÉE PAR :

Christophe PRUD'HOMME	Directeur de thèse, Université de Strasbourg
Yannick HOARAU	Co-directeur de thèse, Université de Strasbourg

RAPPORTEURS :

Martin GREPL	RWTH Aachen University
Anthony NOUY	École Centrale Nantes

EXAMINATEURS :

Fabien MANGEANT	Renault
Robert MOSÉ	Université de Strasbourg

Life is like a box of chocolates.
You never know what you're gonna get.

FORREST'S MOTHER

À Toi, Mam'

Acknowledgment

Je commencerai naturellement par remercier mon directeur de thèse Christophe Prud'homme. Tu m'as fait découvrir l'analyse numérique en première année de master et tu as su me communiquer ta passion pour cette discipline. Tu m'as ensuite très vite fait confiance pour travailler avec toi et ton équipe. Merci de m'avoir donné l'opportunité de travailler sur cette thèse. J'ai énormément appris à tes côtés et je t'en serai toujours reconnaissant.

Merci à Yannick Hoarau, mon co-directeur de thèse, pour son soutien et sa patience. Le travail sur la turbulence fut plus douloureux que prévu mais tu as toujours été de bon conseil et présent quand j'en ai eu besoin.

Merci à mes rapporteurs Anthony Nouy et Martin Grepl qui ont accepté de relire et d'apprécier mon manuscrit. Merci à Fabien Mangeant et Robert Mosé d'avoir accepté de faire parti de mon jury et de contribuer à l'évaluation de mon travail.

Je remercie également le laboratoire de l'IRMA et l'Université de Strasbourg pour m'avoir accueilli pendant ces années de thèse. Merci à tous mes collègues, et particulièrement à mes co-bureau, pour les échanges et les moments partagés.

Merci à toute l'équipe Feel++ pour m'avoir embarqué dans cette belle aventure . Merci à Stéphane et Cécile pour m'avoir mis le pied à l'étrier avec les bases réduites et pour m'avoir aidé à rentrer dans le code, sans vous j'y serai probablement encore perdu. Merci à Lorenzo pour sa bonne humeur au travail et pendant le voyage au US (on remet ça bientôt !). Merci à Guillaume pour les pauses café interminables à refaire le monde, à regarder les canards ou à râler sur tout et n'importe quoi. Merci à Céline pour son soutien indéfectible et pour la grimpe. Un énorme merci à Vincent C. pour son aide précieuse et sa bonne humeur; rares sont les développements que j'aurais pu réaliser sans ton aide ! Un autre énorme merci à Romain pour les

échanges sérieux et moins sérieux, pour les barres de rire, pour les bières, pour m'avoir supporté pendant ces années et pendant tous nos déplacements ensemble. Enfin, un merci tout particulier à Vincent H., le meilleurs nounours qu'on puisse trouver quand on bloque sur un problème !

Merci à ma famille et mes amis qui m'ont aidé et supporté dans cette aventure pendant toutes ces années. Un énorme merci à mes parents qui n'ont jamais perdu espoir ! Vous m'avez toujours soutenu dans mes projets et fais confiance pour réussir. Votre foi dans mes capacités m'a fait avancé pendant toutes ces années. Cette thèse est aussi pour vous et surtout pour toi Maman.

Enfin, le plus gros des merci pour mon Pou et ma Biquette. Cette thèse c'est aussi la votre. Je l'ai écrite, mais on a l'a vécu et souffert ensemble. C'est en vous que j'ai trouvé la force d'aller jusqu'au bout.

Contents

Notations

Acronyms

FE	:	Finite Element
SUPG	:	Streamline Upwind/Petrov-Galerkin
GLS	:	Galerkin Least Square
RANS	:	Reynolds Averaged Navier-Stokes
DNS	:	Direct Numerical Simulation
SA	:	Spalart-Allmaras
RB	:	Reduced Basis
CRB	:	Certified Reduced Basis
EIM	:	Empirical Interpolation Method
SER	:	Simultaneous EIM and RB

Functional Analysis

Ω	:	Regular Bounded Domain	$\Omega \subset \mathbb{R}^d, d = 1, 2, 3$
$\mathcal{L}^p(\Omega)$	:	Lebesgue Space on Ω	section 1.1
$H^k(\Omega)$	:	Sobolev Space on Ω	section 1.2
γ_a	:	Continuity Constant of a	
α_a	:	Coercivity Constant of a	
β_{Br}	:	Brezzi inf-sup Constant	

β_{Ba}	:	Babuška inf-sup Constant	
∇u	:	Gradient of scalar field u	$\nabla u = (\frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2})$
Δu	:	Laplacian of a scalar field u	$\Delta u = \frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2}$
$\boldsymbol{H}u$	:	Hessian matrix of a scalar field u	$\boldsymbol{H}u = \begin{pmatrix} \frac{\partial^2 u}{\partial x_1^2} & \frac{\partial^2 u}{\partial x_1 x_2} \\ \frac{\partial^2 u}{\partial x_2 x_1} & \frac{\partial^2 u}{\partial x_2^2} \end{pmatrix}$
$\nabla \cdot \boldsymbol{u}$	:	Divergence of a vector field $\boldsymbol{u}$	$\nabla \cdot u = \frac{\partial \boldsymbol{u}_1}{\partial x_1} + \frac{\partial \boldsymbol{u}_2}{\partial x_2}$
$\nabla \boldsymbol{u}$	:	Gradient of vector field $\boldsymbol{u}$	$\nabla u = \begin{pmatrix} \frac{\partial \boldsymbol{u}_1}{\partial x_1} & \frac{\partial \boldsymbol{u}_1}{\partial x_2} \\ \frac{\partial \boldsymbol{u}_2}{\partial x_1} & \frac{\partial \boldsymbol{u}_2}{\partial x_2} \end{pmatrix}$
$\Delta \boldsymbol{u}$	:	Laplacian of a vector field $\boldsymbol{u}$	$\Delta u = \begin{pmatrix} \frac{\partial^2 \boldsymbol{u}_1}{\partial x_1^2} + \frac{\partial^2 \boldsymbol{u}_1}{\partial x_2^2} \\ \frac{\partial^2 \boldsymbol{u}_2}{\partial x_1^2} + \frac{\partial^2 \boldsymbol{u}_2}{\partial x_2^2} \end{pmatrix}$
$\boldsymbol{H}\boldsymbol{u}$	:	Hessian rank-3 tensor of a vector field $\boldsymbol{u}$	$(\boldsymbol{H}\boldsymbol{u})_k = \begin{pmatrix} \frac{\partial^2 \boldsymbol{u}_k}{\partial x_1^2} & \frac{\partial^2 \boldsymbol{u}_k}{\partial x_1 x_2} \\ \frac{\partial^2 \boldsymbol{u}_k}{\partial x_2 x_1} & \frac{\partial^2 \boldsymbol{u}_k}{\partial x_2^2} \end{pmatrix}$

Finite Element Method

$\mathcal{T}_h$	:	Mesh on Ω of characteristic size h	
$\boldsymbol{n}$	:	Normal Vector to the Current Face	
$\mathcal{N}$	:	FE space dimension	
$\mathbb{L}^k_{ch}(\Omega)$	:	Continuous Lagrange FE space on Ω	section 2.2
$\mathbb{TH}^k_{ch}(\Omega)$	:	Taylor-Hood FE space on Ω	$\mathbb{TH}^k_{ch} = [\mathbb{L}^{k+1}_{ch}]^d \times \mathbb{L}^k_{ch}$

Aerothermal Simulation

ρ	:	Fluid Density	$\mathrm{kg.m^{-3}}$
μ	:	Fluid Dynamic Viscosity	$\mathrm{Pa.s}$
ν	:	Fluid Kinematic Viscosity	$\mathrm{m^2.s^{-1}}$
κ	:	Fluid Thermal Diffusivity	$\mathrm{m^2.s^{-1}}$
$\boldsymbol{T}$	:	Cauchy Stress Tensor	$\boldsymbol{T}(\boldsymbol{u}, p) = 2\boldsymbol{\mathcal{T}}(\boldsymbol{u}) - \boldsymbol{I}p$
$\boldsymbol{\mathcal{T}}$	:	Viscous Stress Tensor	$\boldsymbol{\mathcal{T}}(\boldsymbol{u}) = \mu \boldsymbol{S}(\boldsymbol{u})$
$\boldsymbol{S}$	:	Strain Tensor	$\boldsymbol{S}(\boldsymbol{u}) = \frac{1}{2}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^\intercal)$
V/V_h	:	Velocity function space	
Q/Q_h	:	Pressure function space	
X/X_h	:	Temperature function space	
$\boldsymbol{u}$	:	Velocity Vector Field	
T	:	Temperature Scalar Field	
p	:	Pressure Scalar Field	

Turbulence Modelization

$\boldsymbol{R}$	:	Reynolds Stress Tensor
k	:	Kinetic Energy
μ_t	:	Eddy Viscosity

Reduced Basis Method

$\mathcal{D}$	:	Parameter Space	$\mathcal{D} \in \mathbb{R}^p$, $p \in \mathbb{N}^*$
Ξ	:	Super sampling for RBM offline part	
X_N	:	Reduced Basis Space of size N	
S_N	:	Sample of parameters on which X_N is built	

Introduction

ANR Project CHORUS

This thesis has been funded by the French National Research Agency (ANR) and the project CHORUS (Common Horizon of Open Research in Uncertainty for Simulation) started in 2013. This project aimed to support the research activities and the software development in the domain of the uncertainty quantification. The work was split into three main topics: the academic research, the software development and the benchmarking on industrial test-cases. All the implementation was made under the open source license to benefit to a large community.

The quantification of uncertainty is the capacity to measure the effect of the uncertainty in numerical models and simulations. In the industry, this subject is particularly important during the early phases of design and certifications. Although the topic is well defined and mastered in the academic community, the application to industrial problems raises some scalability issues. The majority of the methods are based on many query techniques which imply to perform a considerable number of parametric simulations to collect and analyze the data. This kind of procedure leads to scalability issue when we are working on large dimension cases. To tackle that scalability issue there exist two main solutions: the exploitation of the growing HPC (High-Performance Computing) capabilities and the development of efficient and accurate model order reduction methods. Those two topics are highly interconnected and have been widely studied in the context of this ANR project.

The project gathered different industrial and academic research teams to work on these subjects. Our goal was to improve the robustness, the scalability and the reliability of different reduction modeling techniques used for uncertainty quantification:

- the low rank and tensor reduction
- the PGD
- the Reduced Basis
- the Multi fidelity kriging
- the Bayesian calibration

In parallel of the research efforts, industrials proposed different concrete applications. The idea was to confront the academic work to the resolution of complex test-cases.

Airbus Use Case

Airbus Group was one of the leading company in the project. They proposed different subjects of research and test-cases. In particular, they were interested in the development of a reliable reduced model for aerothermal simulations. This research topic has a double objective. First, it makes conceivable the use of numerical simulations in the design and certification phases. The actual full order aerothermal models require huge computation capabilities. With that constraint, it is not practicable to perform a large number of simulations, necessary for the uncertainty quantification or the design optimization. In a second time, Airbus Group was interested in the interface between their environment conditioning system (ECS) and a reduced aerothermal model. This ECS is a numerical model, based on the resolution of multiple ordinary differential equations (ODE). Among other things, it control the pressure and the quality of the air in the aircraft. The integration of a reduced high fidelity 3D model in the ECS would highly increase its possibilities. However, it requires real-time embedded simulations which are impossible with a full order model but achievable with a reduced one.

Airbus proposed to work on an aerothermal flow problem in an airplane. The geometry is decomposed into two sub-domains: the cabin on the top with passengers and the bay on the bottom with various electric equipment. A cooling system controls the temperature in both areas, see figure 1

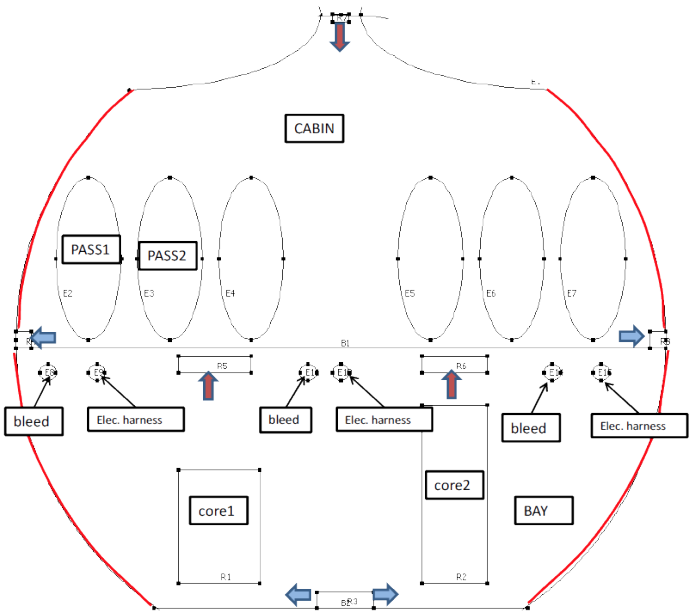


Figure 1 – Description of the Cabin Use Case Proposed by Airbus Group

The development of a reliable reduced order model for this test case was the starting point of this thesis.

Modeling Aerothermal Flows

Before thinking about the reduction of the model, we had to implement a reliable *truth* approximation. That kind of simulations implies to model the fluid flow and the heat transfers in a complex coupled system. The temperature of the fluid is subject to two phenomena: the usual diffusion and the convection through the effect of the velocity of the fluid. These heat transfers can be modeled by a classic advection-diffusion equation. In addition, we have to determine the fluid behavior, governed by the laws of the fluid dynamics.

The Navier-Stokes equations are the key to computational fluid dynamics (CFD). This non-linear system provides a mathematical model for the velocity and pressure field of a Newtonian fluid. The original equations are designed for compressible fluids. In our case, we will only consider flow with reasonably low velocity, and we can use the incompressibility assumption to simplify the equations. The effect of the temperature of the fluid will be model using the Boussinesq approximation to express the density of the fluid as a function of its temperature. Even with a

simplified version of the Navier-Stokes equations, there exists no known analytical solutions to this problem. Thus the numerical resolution is the only actual way to perform CFD[79]. Many approaches exist for the numerical treatment of these equations: finite volume, finite element, finite difference, spectral element, boundary element,... All those methods are based on a discretization of the spatial domain and a resolution of the equations projected on the discrete space.

In the context of this thesis, we chose to work with the finite element method (FEM). This technique is particularly adapted for the resolution of algebraic equations on complex domains. It can be performed on non-structured discretizations and provide accurate approximations with robust convergence theories [17]. We chose this method for its reliability and its convenient formulation. We were also motivated in our choice by the use of the reduced basis method which was initialy designed for the finite element formulation[80].

However, in our case, the physical parameters of the air imply almost negligible diffusion compared to the convection effects. Using the FEM for the resolution of that kind of convection dominated problems, usually produces non-physical oscillations. Thus, the finite element community developed various solutions to tackle this annoying instability issue. In this thesis, we chose to focus on the streamline diffusion methods [46, 7], which allows stabilizing the problem by adding numerical diffusivity in the direction of the convection field. These techniques were widely studied to find the most appropriate formulation for our model.

Those stabilization methods allow recovering a numerically stable solution. Nevertheless, when the Reynolds number is growing, these methods do not take care of the energy dissipated in the smallest scales. To capture these effects, the naive solution would be to use ridiculously thin discretization[57]. This solution is called Direct Numerical Simulation and is not achievable for industrial cases. The alternative is the use of turbulence models to compute an additional turbulent viscosity, modeling the loss of energy in the smallest scales. There exist two main families of turbulence models. The large eddy simulation (LES) is used for transient simulations. The Reynolds average Navier-Stokes (RANS) is adapted for stationary problems, and that is why decide to implement two RANS models: the Spalart-Allmaras and the $k - \omega$ SST [41].

The cited methods allow performing fluid and aerothermal simulations, including for high Reynolds or turbulent flows. The next step is now to build a reduced version of this model.

Model Order Reduction

The model order reduction methods are particularly adapted in the context of parametrized Partial Differential Equations (PDE). The physic and the mesh are always the same, but the parameters of the equations may vary (boundary

conditions, geometrical parametrization, source terms, ...). It is interesting to develop a reduced model when it becomes necessary to solve a parametrized PDE multiple times, with different sets of parameters. The reduced models are usually built from an expensive learning procedure to gather information on the model. Then, it exploits the invariant structures of the problem to compute a new reduced order approximation.

We chose to focus our research on the Reduced Basis Method (RBM), one of the principal model order reduction methods, which presents some interesting advantages [80, 103, 81, 83, 88]:

- Rigorous a posterior error estimators
- An efficient Offline/Online decomposition.

The idea of the reduced basis method is to build a small (relatively to the finite element dimension) dimension space. Then the resolution of the PDE in this low dimension space is very fast and produce accurate approximation as long as the reduced space is rich enough. There exist different algorithms to build this reduced space and the methods are usually problem-dependant. In the context of aerothermal simulations, the constructions of an efficient reduced model also require some advanced techniques to deal with non-linearity: the Empirical Interpolation Method (EIM)[3, 65, 23, 37] will be an essential tool for our reduced basis framework.

Feel++: Finite Element Embedded Library

Feel++ is a finite element embedded library in C++. It offers a very convenient language designed for the resolution of PDE, relying on a Domain Specific Language (DSEL). The goal of Feel++ is to provide a language close to the mathematical one to solve complex PDE. The idea is to provide to scientists a framework in which they can express in a language close to the mathematics the strategy they propose for solving complex systems of PDE and generate a high performance code. To this goal, Feel++ uses the last standards of C++ and meta-programming (through the template and boost meta-programming library) to have a maximum of generality. The library also offers seamless parallel computing. This High Performance Computing capability is essential for the treatment of concrete industrial problems.

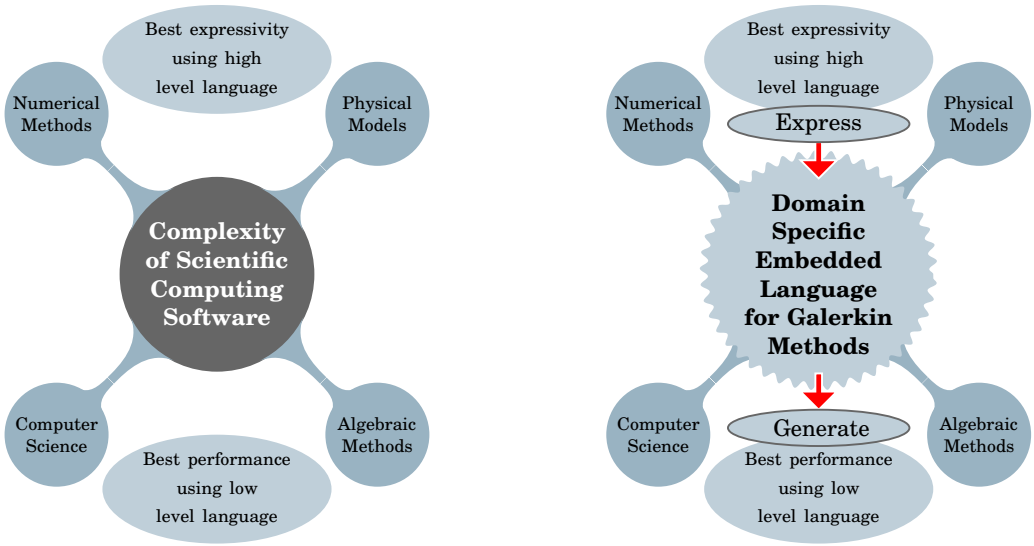


Figure 2 – The DSEL offered by Feel++ provides high level language to break the complexity of scientific computing software while keeping the performances of a low level language.

Feel++ is an open-source library developed under the LGPL and GPL License. It is used in numerous scientific projects funded by different institutions such as the French National Research Agency or the European Commission.

During this thesis, all the applications have been developed using Feel++. We also widely contributed to the reduced basis framework of the library.

Plan

Just after this introduction, the chapter 1 introduces useful mathematical notions.

Then, the manuscript is organized into three parts.

The part I is dedicated to the aerothermal simulation using the finite element method.

In chapter 2 we introduce the governing equations, we present our discretization method and the numerical strategy implemented for the resolution of the non-linear system. The last section of the chapter is dedicated to the validation of our implementation with numerical results on different benchmarks.

In chapter 3, we detail the stabilization method studied during this thesis. We remind the different formulation of the SDM such as the Streamline Upwind/Petrov-Galerkin (SUPG) or the Galerkin Least Square (GLS). Then we discuss the design of the stabilization parameter which significantly impact the efficiency of the method. As a complement of the SDM, we also compare the different discontinuity capturing method, essential for high order stabilization. The efficiency of the methods introduced in this chapter will be discussed with various numerical applications.

In chapter 4, we propose a very brief review of the existing turbulent models,

and we focus on the RANS method. We detail two models in particular: the Spalart-Allmaras and k-ω SST. The implementation of these models is validated by numerical results on the well-known NACA0012 benchmark proposed on the NASA turbulence modeling resource website.

The part II presents our work on the model order reduction. In chapter 5, we remind the principle of the RBM for coercive and saddle-point elliptic problems. We give an example of an application with a reduced model of Stokes flow in a parametrized geometry.

In chapter 6, we present the Empirical Interpolation Method (EIM) and a new implementation of this method developed during this thesis. This new development allows constructing an affine approximation for discrete operators. We also detail the interface of this version of EIM with the Simultaneous EIM and RB (SER) algorithm allowing a decrease of the computational cost. The implementation and the efficiency of the method are discussed with the resolution of a standard benchmark in the last section.

In chapter 7, we finally propose our solution for the reduction of an aerothermal model. We detail the treatment of the quadratic non-linearity, the reduction of the stabilization operators and the construction of the reduced spaces. The method is illustrated with two applications. The first one is a simulation of the heat transfers in the neighboring of an electronic component, refreshed by a cooling system. The second one is a simplified version of the use-case proposed by Airbus Group; we proposed a reduced model of the aerothermal flow into the airplane cabin.

The last part III is an overview of the implementation work realized during this thesis. In chapter 8, we present the design and the different features implemented in our aerothermal library. These tools and algorithms are now progressively integrated into the Feel++ toolboxes.

In chapter 9, we give an overview of the developments in Feel++ RB framework. The main contributions are the implementation of the reduced basis by block for multi-physics problem and the reduced basis for aerothermal problems.

In chapter 10, we finally detail the implementation of the EIM for discrete operators. This development has been entirely realized during this thesis and involved interesting methods of parallel and metaprogramming.

Publications

in preparation	*Review and Implementation of Streamline Diffusion Methods on Anisotropic Meshes, Application to Aerothermal Simulations* with C. Prud'homme
in preparation	*Implementation of RANS models in Feel++ library* with C. Prud'homme, Y. Hoarau, V. Chabannes

in preparation	*Simultaneous EIM and RB Construction for Non-linear Operators, Application to Aerothermal Problems* with C. Prud'homme, V. Chabannes

Oral Presentations

2018	**MoRePaS**, Nantes, France *High Reynolds Aerothermal Simulations and Reduced Basis*
2017	**Feel++ Users Days**, Strasbourg, France *Aerothermal Simulation and Model Order Reduction, Using the Open-Source Framework Feel++*
2018	**ANR CHORUS Workshop**, Paris, France *Model Order Reduction for Multi-physics Problems, Using the Open-Source Framework Feel++*
2017	**A3F**, Strasbourg, France *Aerothermal Simulation and Model Order Reduction, using the Open-Source Framework Feel++*
2015	**SimRace**, Paris, France *Solving Strategy for Large Scale Aerothermal Simulation using the Open-Source Framework Feel++*
2015	**Feel++ Users Days**, Strasbourg, France *Aerothermal Simulations : Towards Reduced Basis Applications*
2014	**Research Group on Model Order Reduction**, Porquerolles, France

Posters

2018	**CANUM**, Cap d'Agde, France Poster: *High Reynolds Aerothermal Simulations and Reduced Basis*
2015	**Journée Poster de l'école doctoral** , Strasbourg, France Poster: *La Méthode des Bases Réduites Appliquée à des Simulations d'Aerothermie*

CHAPTER 1

Preliminary Notions

Contents

This first chapter is dedicated to the definition of some mathematical notions and notations, relevant to the context of this document.

1 Function Spaces

In the following sections, and more generally in this thesis report, Ω is supposed to be a bounded domain of $\mathbb{R}^d$, $d = 1, 2, 3$, with a Lipschitz boundary $\partial\Omega = \Gamma$.

1.1 Lebesgue Space

Lebesgue spaces, $\mathcal{L}^p(\Omega)$, are spaces of functions on Ω for which the p-th power of its absolute value is Lebesgue integrable,

$$\mathcal{L}^p(\Omega) = \{v : \Omega \to \mathbb{R}, \int_\Omega |v|^p \, d\Omega \ < \infty\} \tag{1.1}$$

$\mathcal{L}^p(\Omega)$ is a Banach space with the usual p-norm $|| \cdot ||_p$,

$$||v||_p = (\int_\Omega |v|^p \, d\Omega \ )^{1/p}. \tag{1.2}$$

We can extend this notion to the particular case of $p = \infty$ using the essential supremum of the function v as ∞-norm,

$$\mathcal{L}^\infty(\Omega) = \{v : \Omega \to \mathbb{R}^n, ||v||_\infty < \infty\}, \qquad ||v||_\infty = \operatorname*{ess\,sup}_{\boldsymbol{x}\in\Omega} |v(\boldsymbol{x})|, \tag{1.3}$$

where the essential supremum of a function is defined as the supremum of the function *almost everywhere*, i.e. except on a set of measure zero. This notion is an adaptation of the usual global supremum to measure theory.

In the following report, we essentially use the common $\mathcal{L}^2(\Omega)$ space of square-integrable functions,

$$\mathcal{L}^2(\Omega) = \{v : \Omega \to \mathbb{R}^n, ||v||_2 < \infty\}, \quad ||v||_2 = \sqrt{\int_\Omega |v|^2 \, d\Omega} \, . \tag{1.4}$$

$\mathcal{L}^2$ is particularly interesting since he has an Hilbert structure, with its usual scalar product

$$(u, v)_0 = \int_\Omega u \cdot v \, d\Omega \, . \tag{1.5}$$

1.2 Sobolev Space

A Sobolev space $W^{k,p}(\Omega)$ is a subspace of $\mathcal{L}^p(\Omega)$ such that for any $v \in W^{k,p}(\Omega)$ all the weak derivatives of v, up to the order k, are in $\mathcal{L}^p(\Omega)$,

$$W^{k,p}(\Omega) = \{v \in \mathcal{L}^p(\Omega), D^\alpha v \in \mathcal{L}^p(\Omega), |\alpha| \leq k\}, \tag{1.6}$$

where $D^\alpha v$ is the generalised weak derivative operator, defined for any multi-index $\alpha = (\alpha_1, ..., \alpha_d)$, such that $\alpha_i \geq 0$, $1 \leq i \leq d$, and for any $\boldsymbol{x} = (x_1, .., x_d) \in \Omega$ as

$$D^\alpha = \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} ... \partial x_d^{\alpha_d}}, \quad |\alpha| = \sum_{i=1}^{d} \alpha_i. \tag{1.7}$$

Sobolev spaces have Banach structure with the norm

$$\begin{aligned} \|v\|_{k,p} &= \left(\sum_{|\alpha| \leq k} \|D^\alpha v\|_p^p \right)^{1/p}, \qquad 1 \leq p \leq +\infty \\ \|v\|_{k,\infty} &= \max_{|\alpha|<k} \|D^\alpha v\|_\infty. \end{aligned} \tag{1.8}$$

In the context of parametrized differential equations, the Sobolev spaces with $p = 2$ are particularly interesting since they have an Hilbert structure, with the scalar product

$$(u,v)_k = \int_\Omega u \cdot v \, d\Omega + \sum_{0<|\alpha|\leq k} \int_\Omega D^\alpha u \cdot D^\alpha v \, d\Omega = (u,v)_0 + \sum_{0<|\alpha|\leq k} \int_\Omega D^\alpha u \cdot D^\alpha v \, d\Omega \,, \tag{1.9}$$

and the associated norm

$$||v||_k = \sqrt{(v,v)_k}. \tag{1.10}$$

In this case, we use the notation $H^k(\Omega) = W^{k,2}(\Omega)$,

$$H^k(\Omega) = \{v \in \mathcal{L}^2(\Omega), D^\alpha v \in \mathcal{L}^2(\Omega), |\alpha| \leq k\}. \tag{1.11}$$

1.3 Dual Space

Let X be an Hilbert space with scalar product $(\cdot,\cdot)_X$ and associated norm $||\cdot||_X$. The dual spaces X' of X is the set of continuous linear forms on X,

$$X' = \{l : X \to \mathbb{R}, l \text{ continuous linear}\}. \tag{1.12}$$

For any $l \in X'$, we define its dual norm by

$$||l||_{X'} = \sup_{v \in X} \frac{|l(v)|}{||v||_X} \tag{1.13}$$

The Riesz representation theorem establishes that X and X' are isometrically isomorphic. A direct consequence of this theorem is the existence, for any $l \in X'$, of a Riesz representation $\hat{l} \in X$ such that

$$(\hat{l}, v)_X = l(v), \quad \forall v \in X. \tag{1.14}$$

This Riesz representation is very convenient to evaluate the dual norm of a linear form l, using the isometrical relation between X and X',

$$||\hat{l}||_X = ||l||_{X'} = \sup_{v \in X} \frac{|l(v)|}{||v||_X}. \tag{1.15}$$

This relation is widely used in the reduced basis methodology, see chapter 5.

2 The Finite Element Method

2.1 Variational Formulation

In a weak formulation, an equation is no longer required to hold absolutely as in its strong formulation. Instead, the weak formulation requires a solution in the sense of distributions and might not be defined on a set of measure zero. This formulation is crucial in the context of functional analysis and especially for treatment of PDEs.

In the present report, we assume that all studied PDE admit a weak formulation, written as find $u \in X$ such that

$$a(u, v) = f(v), \quad \forall v \in X, \tag{1.16}$$

where X is an Hilbert space on Ω, $a : X \times X \to \mathbb{R}$ is a bilinear form and $f : X \to \mathbb{R}$ is a linear form. u is called the trial function and lives in the trial space X while v are the test functions living in the test space X.

We now remind two fundamental theorems that ensure the well-posedness of the examined problems. In the rest of this report, when we describe a problem and its weak formulation, we assume that it fulfills the conditions of one of these theorems - depending on the nature of the problem.

Theorem 1 (Lax-Milgram)**.**
We consider the problem (1.16) *and we furthermore assume that*

(a) the linear form l is continuous,

$$\gamma_l = \sup_{v \in X} \frac{|l(v)|}{||v||_X} < \infty \tag{1.17}$$

(b) the bilinear form a is continuous,

$$\gamma_a = \sup_{u \in X} \sup_{v \in X} \frac{|a(u, v)|}{||u||_X ||v||_X} < \infty \tag{1.18}$$

(c) the bilinear form a is coercive,

$$\alpha_a = \inf_{v \in X} \frac{a(v, v)}{||v||_X^2} > 0. \tag{1.19}$$

Under these assumptions, there exists an unique solution $u \in X$ to the weak formulation (1.16)*.*

Proof. See [60].

□

Remark 1. *The proof of this theorem comes from the calculus of variations; that is why the term of* variational *formulation is often used to refer to the weak formulation.*

Theorem 2 (Brezzi)**.**
In the context of fluid simulation we will consider saddle point problems expressed under the variational formulation: find $(\boldsymbol{u}, p) \in V \times Q$ *such that*

$$\begin{cases} a(\boldsymbol{u}, \boldsymbol{v}) + b(\boldsymbol{v}, p) = f(\boldsymbol{v}), & \forall \boldsymbol{v} \in V, \\ b(\boldsymbol{u}, q) = g(q), & \forall q \in Q, \end{cases} \tag{1.20}$$

where V *and* Q *are two Hilbert spaces with associated scalar products* $(\cdot,\cdot)_V$, $(\cdot,\cdot)_Q$ *and norms* $||\cdot||_V$, $||\cdot||_Q$. Z *is the composite space* $Z = V \times Q$ *with inner product* $(\cdot,\cdot)_Z = (\cdot,\cdot)_V + (\cdot,\cdot)_Q$ *and norm* $||\cdot||_Z = \sqrt{(\cdot,\cdot)_Z}$. $a(\cdot,\cdot) : V \times V \to \mathbb{R}$, $b(\cdot,\cdot) : V \times Q \to \mathbb{R}$ *and* $f(\cdot) : V \to \mathbb{R}$, $g(\cdot) : Q \to \mathbb{R}$ *are bilinear and linear bounded forms respectively.*

We furthermore assume that :

(a) a *and* b *are continuous*

$$\gamma_a = \operatorname*{supsup}_{\boldsymbol{u}\in V\, \boldsymbol{v}\in V} \frac{|a(\boldsymbol{u}, \boldsymbol{v})|}{||\boldsymbol{u}||_V ||\boldsymbol{v}||_V} < \infty \tag{1.21}$$

$$\gamma_b = \operatorname*{supsup}_{\boldsymbol{v}\in V\, q\in Q} \frac{|b(\boldsymbol{v}, q)|}{||\boldsymbol{v}||_V ||q||_Q} < \infty, \tag{1.22}$$

(b) a *is coercive on* V

$$\alpha_a = \inf_{\boldsymbol{v}\in V} \frac{a(\boldsymbol{v}, \boldsymbol{v})}{||\boldsymbol{v}||_Q^2} > 0, \tag{1.23}$$

(c) b *satisfies the inf-sup condition (or Ladyzhenskaya–Babuška–Brezzi condition), expressed either using the Brezzi definition,*

$$\beta_{Br} = \inf_{q\in Q} \sup_{\boldsymbol{v}\in V} \frac{b(\boldsymbol{v}, q)}{||\boldsymbol{v}||_V ||q||_Q} > 0, \tag{1.24}$$

or the Babuška definition,

$$\beta_{Ba} = \inf_{(\boldsymbol{u},p)\in Z} \sup_{(\boldsymbol{v},q)\in Z} \frac{a(\boldsymbol{u}, \boldsymbol{v}) + b(\boldsymbol{v}, p) + b(\boldsymbol{u}, q)}{||(\boldsymbol{u}, p)||_Z ||(\boldsymbol{v}, q)||_Z} > 0. \tag{1.25}$$

Under these assumptions, there exists a unique couple of solutions $(\boldsymbol{u}, p) \in Z$ *to the problem* (1.20)*.*

Proof. See [5]. □

2.2 Finite Element Definition

Using the formalism introduced in [17], a finite element is defined by a tuple (K, P, Σ). K is a compact, connected and non-empty geometrical element with a Lipschitz boundary - typically a simplex or a hypercube. P is a finite dimensional function space on K. Σ is a set of N_l linear forms $\{\sigma_i : P \to \mathbb{R}\}_{i=1}^{N_l}$ such that

$$\begin{aligned} P &\to \mathbb{R}^{N_l} \\ p &\to (\sigma_1(p), ..., \sigma_{N_l}(p)) \end{aligned} \tag{1.26}$$

is bijective. Those linear forms are called the degrees of freedom (DoF) of the finite element.

We find many kinds of finite element in the literature: Lagrange, Hermite, Raviart-Thomas, Crouzeix-Raviart, Nedelec,... Lagrange finite element is probably the most common type and will be used in this work. For more details on the finite elements available in the Feel++ library, we refer the reader to [15, 24].

Lagrange Finite Element

We denote by $\mathbb{P}^k(K)$ the set of polynomial functions on K of maximum degree k. We introduce the interpolation points $\{t_i\}_{i=1}^{N_l}$ defined on the element K. The linear forms in Σ are then defined as

$$\begin{aligned} \sigma_i : \mathbb{P}^k(K) &\to \mathbb{R} \\ p &\to p(t_i) \end{aligned} \tag{1.27}$$

A Lagrange finite element of order k is defined by the tuple $(K, \mathbb{P}^k(K), \Sigma)$ and the local basis functions $\{\phi_i\}_{i=1}^{N_l}$ are set such that

$$\sigma_i(\phi_j) = \delta_{ij}, \quad \forall 1 \leq i, j \leq N_l, \tag{1.28}$$

where δ_{ij} is the Kronecker symbol.

Lagrange Finite Element Space

We introduce now a conforming partition of the domain Ω

$$\mathcal{T}_h = \{K_m\}_{m=1}^{N_e} \tag{1.29}$$

where all the elements K_i are of same nature - simplex or hypercubes. For any Hilbert space $X(\Omega)$ on Ω, we can define its discrete representation on the mesh $\mathcal{T}_h$, as the continuous Lagrange finite element space of order k

$$X_h(\Omega) \equiv \mathbb{L}^k_{ch}(\Omega) = \{v \in X(\Omega) \cap C^0(\Omega), v|_{K_m} \in \mathbb{P}^k(K_m)\}. \tag{1.30}$$

The subscript $_c$ stands for the continuous property which is actually not mandatory, depending on the projection method. Hereafter we will only consider the Galerkin projection with continuous Lagrange space $\mathbb{L}^k_{ch}$. The subscript $_h$ stands for the characteristic length of the mesh $\mathcal{T}_h$. We usually denote by $\mathcal{N}$ the dimension of the FE space $X_h(\Omega)$ and use the convenient notation $X_h \equiv X_h(\Omega)$ when it is not ambiguous.

The FE space X_h is an Hilbert space with scalar product $(\cdot,\cdot)_X$ and associated norm $||\cdot||_X$ inherited from X.

Remark 2. *It was convenient in the previous definition to make the assumption $\Omega_h = \Omega$, with $\Omega_h = \cup_{K\in\mathcal{T}_h} K$. In practice, we only have an approximation $\Omega_h \approx \Omega$ and the FE element space X_h is not defined on Ω but only on the approximated domain Ω_h.*

2.3 Galerkin Projection

We use the Galerkin method as internal approximation method. In order to numerically solve the problem (1.16), we introduce a FE subspace $X_h \subset X$. The exact solution u of (1.16) will be approximated by the discrete solution $u_h \in X_h$ such that

$$a(u_h, v_h) = f(v_h), \quad \forall v_h \in X_h. \tag{1.31}$$

Since the solution $u_h \in X_h = \text{span}\{\phi_i, 1 \le i \le \mathcal{N}\}$, we have,

$$u_h = \sum_{j=1}^{\mathcal{N}} u_h^j \phi_j. \tag{1.32}$$

The variational formulation can then be rewritten as find $(u_h^1, ...u_h^{\mathcal{N}})$ such that

$$\sum_{i=1}^{\mathcal{N}} u_h^j a(\phi_j, \phi_i) = f(\phi_i), \quad \forall 1 \le i \le \mathcal{N}. \tag{1.33}$$

And finally, introducing the matrix $\boldsymbol{A}_h$, $(\boldsymbol{A}_h)_{ij} = a(\phi_j, \phi_i)$ and the vector $\boldsymbol{F}_h$, $(\boldsymbol{F}_h)_i = f(\phi_i)$, the Galerkin projection u_h of the continuous solution u is the solution of the linear system

$$\boldsymbol{A}_h\, u_h = \boldsymbol{F}_h \tag{1.34}$$

Well-Posedness of the Discrete Problem

Assuming that the problem (1.16) is well-posed in the sense of the Lax-Milgram theorem 1, we have the following relations:

$$\gamma_l \ge \gamma_{lh} = \sup_{v_h \in X_h} \frac{|l(v_h)|}{||v_h||_X}, \tag{1.35}$$

$$\gamma_a \geq \gamma_{ah} = \sup_{u_h \in X_h} \sup_{v_h \in X_h} \frac{|a(u_h, v_h)|}{||u_h||_X ||v_H||_X}, \tag{1.36}$$

$$\alpha_a \leq \alpha_{ah} = \inf_{v_h \in X_h} \frac{a(v_h, v_h)}{||v_h||_X^2}. \tag{1.37}$$

The properties of continuity and coercivity are then inherited from the continuous problem, and it guarantees the well-posedness of discrete problem (1.34), [60].

In the case of saddle point problem, the property of continuity is also inherited from the continuous problem. However, the inf-sup condition is not automatically satisfied for any FE space $V_h \times Q_h$. A very convenient solution to satisfy this necessary condition is to choose specific finite element space. For instance, in the context of this thesis, we use Taylor-Hood spaces of degree k,

$$\mathbb{TH}_{ch}^k = [\mathbb{L}_{ch}^{k+1}]^d \times \mathbb{L}_{ch}^k. \tag{1.38}$$

This discretization ensures the inf-sup condition to be satisfied [5], and then the well-posedness of the discrete problem is ensured by the Brezzi theorem 2.

Convergence of the Method

The finite element methodology provides bounds on the error between the exact solution u and the FE approximation u_h, [17]. This error depends on the following parameters:

- the polynomial order k of the FE space,
- the nature of the considered norm: usually the H^m-norm $||\cdot||_m$, as defined in section 1.2. The case $m = 0$ stands for the $\mathcal{L}^2$-norm,
- the characteristic length h of the mesh $\mathcal{T}_h$,
- the regularity of the solution.

Considering these elements, we have the existence of a constant $C \in \mathbb{R}, C > 0$ such that

$$||u - u_h||_m \leq C h^{k+1-m}. \tag{1.39}$$

This bound ensures the convergence of the FE solution u_h when the characteristic length of the mesh decrease,

$$u_h \underset{h \to 0}{\to} u. \tag{1.40}$$

Part I

Aerothermal Simulations

CHAPTER 2

Modeling of an Aerothermal Problem

Contents

In this first chapter we describe the full resolution process, from the continuous level to the computation of a discrete solution. We start with the formulation of the governing equations arising from different physical laws and approximations. In the second section, we detail the discretization of the continuous problem, using the FEM. And finally, we present our resolution strategy for the non-linear system, in section3. The last section of this chapter is dedicated to numerical results and especially to the validation of our implementation.

1 The Navier Stokes Equations and the Boussinesq Approximation

In the context of this thesis, the fluid is always supposed Newtonian and incompressible. Besides, we use the Boussinesq approximation for modeling the buoyancy force. With these hypotheses, it only remains three quantities of interest in the aerothermal system: the velocity ($\boldsymbol{u}$), the pressure (p) and the temperature (T).

In this first section, we briefly remind the laws of conservation involved in the establishment of the Navier-Stokes equations [79]. These equations mainly derive from the principles of continuity of mass, momentum, and energy. Those physical rules combined with the Boussinesq approximation lead to a coupled system between the three quantities, $\boldsymbol{u}$ p, and T.

Derivation of continuity equation

This equation arises from the fundamental law of Newton mechanics that states the conservation of mass (M) in an arbitrary control volume (V_m) which leads to

$$0 = \frac{dM}{dt} = \frac{d}{dt}\int_{V_m} \rho \, dV = \int_{V_m} \frac{\partial \rho}{\partial t} \, dV + \int_{S_m} \rho \boldsymbol{u} \cdot \boldsymbol{n} \, dS = \int_{V_m} (\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u})) \, dV, \quad (2.1)$$

where S_m is the boundary surface of the volume V_m, $\boldsymbol{n}$ is the outward normal vector to the surface and ρ is the density of the fluid. Since the volume V_m is arbitrary chosen, the integrand has to be zero and leads to the continuity equation,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0. \quad (2.2)$$

In the particular case of incompressible fluids, the density (ρ) is considered constant, this leads to the simplified continuity equation,

$$\boxed{\nabla \cdot \boldsymbol{u} = 0.} \quad (2.3)$$

Derivation of Momentum Equation

One way to obtain this second equation is the application of the second Newton's law to a fluid volume element. It leads to the first form of the momentum equation,

$$\rho(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}) = \boldsymbol{s}, \quad (2.4)$$

where $\boldsymbol{s}$ is the momentum source. This term is composed of two kinds of forces: the surface forces act only on the surface of a control volume whereas body forces act on the whole control volume. Thus, by splitting $\boldsymbol{s}$, we obtain the Cauchy equation

$$\rho(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}) = \nabla \cdot \boldsymbol{T} + \boldsymbol{f} \quad (2.5)$$

where $\boldsymbol{f}$ represents the sum of different body forces applied on the fluid and $\boldsymbol{T}$ is the Cauchy stress tensor, used to express the surface forces acting on the fluid. This equation can again be rewritten by introducing the viscous stress tensor $\boldsymbol{\mathcal{T}}$ and the pressure field p,

$$\rho(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}) = -\nabla p + \nabla \cdot \boldsymbol{\mathcal{T}} + \boldsymbol{f}. \tag{2.6}$$

A fluid is defined as Newtonian [4] when the following assumptions are verified:

- The viscous stress tensor is a linear function of the strain rate tensor, $\boldsymbol{\mathcal{T}} = 2\mu \boldsymbol{S}$, where $\boldsymbol{S}(\boldsymbol{u}) = \frac{1}{2}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^{\intercal})$,
- The fluid is isotropic,
- $\nabla \cdot \boldsymbol{\mathcal{T}} = 0$ at rest.

And by adding the incompressible hypothesis (2.3), we obtain

$$\boxed{\rho(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}) = -\nabla p + 2\nabla \cdot (\mu \boldsymbol{S}(\boldsymbol{u})) + \boldsymbol{f}.} \tag{2.7}$$

Henceforth, the only body force considered is the gravity: $\boldsymbol{f} = \rho \boldsymbol{g}$, where $\boldsymbol{g}$ is the gravitational field.

The Boussinesq Approximation

The Boussinesq approximation states that density differences are sufficiently small to be neglected, except where they appear in terms multiplied by $\boldsymbol{g}$. So we can assume that $\rho = \rho_0$ except in the buoyancy force which is the only force acting on the fluid (in the considered model).

The Boussinesq approximation also links the density variation to the temperature field with the relation

$$\rho - \rho_0 = -\rho_0 \beta (T - T_0), \tag{2.8}$$

where ρ_0 and T_0 are reference density and temperature and β is the coefficient of thermal expansion. We can then rewrite the buoyancy force

$$\boldsymbol{f} = \rho \boldsymbol{g} = \rho_0 \boldsymbol{g} - \rho_0 \beta (T - T_0) \boldsymbol{g}. \tag{2.9}$$

This yields to a new momentum equation

$$\rho(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}) = -\nabla p + 2\nabla \cdot (\mu \boldsymbol{S}(\boldsymbol{u})) + \rho_0 \boldsymbol{g} - \rho_0 \beta (T - T_0) \boldsymbol{g}. \tag{2.10}$$

To avoid the evaluation of the term $\rho_0 \boldsymbol{g}$, we use a pressure shift and we define the new pressure $p^* = p + \rho_0 \boldsymbol{g} h$, where h is the elevation. From here, we will just ignore the * superscript and denote by p the shifted pressure.

The Boussinesq approximation also gives a simplified energy equation [98], by considering the different hypothesis exposed and the fact that the heat capacity is constant,

$$\boxed{\frac{\partial T}{\partial t} + \boldsymbol{u} \cdot \nabla T = \nabla \cdot (\kappa \nabla T) + \frac{J}{\rho C_p}} \tag{2.11}$$

where κ is the thermal diffusivity, C_p is the heat capacity of the fluid and J is the rate per unit volume of internal heat production. This last quantity is always supposed to be zero in the present work.

Strong Formulation

We obtain a non-linear system of three coupled equations

$$\begin{cases} \rho(\dfrac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}) + \nabla p - 2\nabla \cdot (\mu \boldsymbol{S}(\boldsymbol{u})) = -\rho\beta(T - T_0)\boldsymbol{g}, \\ \nabla \cdot \boldsymbol{u} = 0, \\ \dfrac{\partial T}{\partial t} + \boldsymbol{u} \cdot \nabla T - \nabla \cdot (\kappa \nabla T) = 0. \end{cases} \tag{2.12}$$

This system is completed with boundary conditions and initial values for the different fields of interest. These closure conditions will be detailed for each studied model and test case but we can already enumerate some usual boundary conditions and notations.

We consider the problem (2.12) on a domain $\Omega \in \mathbb{R}^d$, $d = 2, 3$, with a sufficiently smooth boundary, $\partial\Omega = \Gamma$. This boundary is partitioned (in two different ways) to settle the different boundary conditions

$$\Gamma = \Gamma_D^F \cup \Gamma_N^F \cup \Gamma_S = \Gamma_D^T \cup \Gamma_N^T \cup \Gamma_R^T, \tag{2.13}$$

where the F and T superscript stand for fluid or temperature and the D, N, S and R subscript stand for Dirichlet, Neumann, Slip and Robin conditions. On these subsets of Γ, the boundary conditions can be described as

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{u}_D(\boldsymbol{x}, t), && \text{on } \Gamma_D^F, && \text{Dirichlet condition on Fluid} \\ \boldsymbol{T}(\boldsymbol{u}, p)\boldsymbol{n} &= \boldsymbol{0}, && \text{on } \Gamma_N^F, && \text{Neumann condition on Fluid} \\ \boldsymbol{u} \cdot \boldsymbol{n} = 0,\ & (\boldsymbol{T}(\boldsymbol{u}, p)\boldsymbol{n})_{\boldsymbol{\tau}} = \boldsymbol{u}_S && \text{on } \Gamma_S^F, && \text{Slip condition on Fluid} \\ T &= T_D(\boldsymbol{x}, t) && \text{on } \Gamma_D^T, && \text{Dirichlet condition on Temperature} \\ \kappa \nabla T \cdot \boldsymbol{n} &= \phi_\Gamma(\boldsymbol{x}, t) && \text{on } \Gamma_N^T, && \text{Neumann condition on Temperature} \\ \kappa \nabla T \cdot \boldsymbol{n} &= \kappa_\Gamma(T_R(\boldsymbol{x}, t) - T) && \text{on } \Gamma_R^T, && \text{Robin condition on Temperature} \end{aligned} \tag{2.14}$$

where $\boldsymbol{u}_D$, $\boldsymbol{u}_S$, T_D, F_Γ and κ_Γ are at least $\mathcal{L}^2$ functions on Γ.

2 Finite Element Approach

We now detail the application of the finite element method (FEM) to the aerothermal system (2.12). We are mainly interested in the steady state of the model (when it exists) and then the stationary system reads

$$\begin{cases} \rho \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p - 2\nabla \cdot (\mu \boldsymbol{S}(\boldsymbol{u})) = -\rho\beta(T - T_0)\boldsymbol{g}, \\ \nabla \cdot \boldsymbol{u} = 0, \\ \boldsymbol{u} \cdot \nabla T - \nabla \cdot (\kappa \nabla T) = 0, \\ +\text{Boundary Conditions 2.14.} \end{cases} \tag{2.15}$$

2.1 Variational Formulation

We start by introducing the Hilbert spaces V, Q and X with the scalar products $(\cdot,\cdot)_V$, $(\cdot,\cdot)_Q$, $(\cdot,\cdot)_X$ and associated norms $||\cdot||_V$, $||\cdot||_Q$, $||\cdot||_X$, respectively. We also define the composite space $Z = V \times Q$ with the scalar product $(\cdot,\cdot)_Z = (\cdot,\cdot)_V + (\cdot,\cdot)_Q$ and associated norm $||\cdot||_Z = \sqrt{||\cdot||_V + ||\cdot||_Q}$.

We apply the scalar product to each row of the system (2.15) with its respective test function $\boldsymbol{v} \in V$, $q \in Q$ or $S \in X$. Then, by using standard integration by part formulas, we obtain the variational formulation, where the weak solution $(\boldsymbol{u}, p, T) \in V \times Q \times X$ is such that

$$\begin{aligned} \int_\Omega \rho(\boldsymbol{u} \cdot \nabla \boldsymbol{u}) \cdot \boldsymbol{v}\, d\Omega \; + \int_\Omega 2\mu \boldsymbol{S}(\boldsymbol{u}) : \boldsymbol{S}(\boldsymbol{v})\, d\Omega \; - \int_\Gamma \boldsymbol{T}(\boldsymbol{u}, p)\boldsymbol{n} \cdot \boldsymbol{v}\, d\Gamma \\ - \int_\Omega p \nabla \cdot \boldsymbol{v}\, d\Omega \; - \alpha \int_\Omega (\nabla \cdot \boldsymbol{u}) q\, d\Omega \; = - \int_\Omega \rho\beta(T - T_0)\boldsymbol{g} \cdot \boldsymbol{v}\, d\Omega\,, \\ \int_\Omega (\boldsymbol{u} \cdot \nabla T) S\, d\Omega \; + \int_\Omega \kappa \nabla T \cdot \nabla S\, d\Omega \; - \int_\Gamma \kappa \nabla T \boldsymbol{n} \cdot S\, d\Gamma \; = 0, \\ \forall (\boldsymbol{v}, q, S) \in V \times Q \times X, \end{aligned} \tag{2.16}$$

where $\alpha = \pm 1$ determines when we use symmetric or asymmetric formulation.

The terms $\int_\Gamma \boldsymbol{T}(\boldsymbol{u}, p)\boldsymbol{n} \cdot \boldsymbol{v}\, d\Gamma$ and $\int_\Gamma \kappa \nabla T \cdot \boldsymbol{n} S\, d\Gamma$ allow to apply the boundary conditions 2.14. The test spaces V and X are supposed to verify $\boldsymbol{v}|_{\Gamma_D^F} = 0, \forall \boldsymbol{v} \in V$ and $T|_{\Gamma_D^T} = 0, \forall T \in X$. Thus the terms $\int_\Gamma \boldsymbol{T}(\boldsymbol{u}, p)\boldsymbol{n} \cdot \boldsymbol{v}\, d\Gamma$ and $\int_\Gamma \kappa \nabla T \cdot \boldsymbol{n} S\, d\Gamma$ are zero on Γ_D^F and Γ_D^T, respectively, and the Dirichlet boundary conditions will be imposed strongly as described later. The Neumann boundary conditions on fluid also set the term $\int_\Gamma \boldsymbol{T}(\boldsymbol{u}, p)\boldsymbol{n} \cdot \boldsymbol{v}\, d\Gamma$ to zero since the Cauchy stress tensor is supposed to be null. The treatment of the slip boundary condition was not trivial and will be detailed in a further paragraph. It remains the Robin and Neumann boundary conditions on the temperature which leads to

$$\int_\Gamma \kappa \nabla T \cdot \boldsymbol{n} S\, d\Gamma \; = \int_{\Gamma_N^T} \phi_\Gamma S\, d\Gamma \; + \int_{\Gamma_R^T} \kappa_\Gamma (T_R - T) S\, d\Gamma\,. \tag{2.17}$$

2.2 Discretization

In order to solve the problem, (2.16) we introduce the discrete spaces

$$V_h \times Q_h = \mathbb{TH}^k_{ch} \subset V \times Q, \qquad X_h = \mathbb{L}^k_{ch} \subset X, \tag{2.18}$$

see section 2 for a description of these FE spaces. The resolution on these discrete spaces leads to the computation of the FE approximation $(\boldsymbol{u}_h, p_h, T_h) \in V_h \times Q_h \times X_h$ such that

$$\begin{aligned} c(\boldsymbol{u}_h, \boldsymbol{v}_h; \boldsymbol{u}_h) + d_u(\boldsymbol{u}_h, \boldsymbol{v}_h) + \alpha b(\boldsymbol{v}_h, p_h) + b(\boldsymbol{u}_h, q_h) &= f(\boldsymbol{v}_h; T_h), \\ a(T_h, S_h; \boldsymbol{u}_h) + d_T(T_h, S_h) &= g(S), \\ \forall (\boldsymbol{v}_h, q_h, S_h) \in V_h \times Q_h \times X_h, & \end{aligned} \tag{2.19}$$

where, $\forall (\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}) \in V^3$, $\forall q \in Q$, $\forall (T, S) \in X^2$,

$$\begin{aligned} c(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{w}) &= \int_\Omega \rho(\boldsymbol{w} \cdot \nabla \boldsymbol{u}) \cdot \boldsymbol{v} \, d\Omega \,, & d_u(\boldsymbol{u}, \boldsymbol{v}) &= \int_\Omega 2\mu \boldsymbol{S}(\boldsymbol{u}) : \boldsymbol{S}(\boldsymbol{v}) \, d\Omega \,, \\ b(\boldsymbol{u}, q) &= -\int_\Omega (\nabla \cdot \boldsymbol{u}) q \, d\Omega \,, & & \\ a(T, S; \boldsymbol{u}) &= \int_\Omega (\boldsymbol{u} \cdot \nabla T) S \, d\Omega \,, & d_T(T, S) &= \int_\Omega \kappa \nabla T \cdot \nabla S \, d\Omega \, + \int_{\Gamma^T_R} \kappa_\Gamma T S \, d\Gamma \\ f(\boldsymbol{v}; T) &= \int_\Omega \rho \beta (T - T_0) \boldsymbol{g} \cdot \boldsymbol{v} \, d\Omega \,, & g(S) &= \int_{\Gamma^T_N} \phi_\Gamma S \, d\Gamma \, + \int_{\Gamma^T_R} \kappa_\Gamma T_R S \, d\Gamma \end{aligned} \tag{2.20}$$

By introducing the basis $\mathcal{B}_{\boldsymbol{u}} = \{\zeta_i\}_{i=1}^{\mathcal{N}_{\boldsymbol{u}}}$, $\mathcal{B}_p = \{\eta_i\}_{i=1}^{\mathcal{N}_p}$ and $\mathcal{B}_T = \{\xi_i\}_{i=1}^{\mathcal{N}_T}$ of the spaces V_h, Q_h and X_h respectively, we can rewrite system (2.20) in the matrix formulation: find $(\boldsymbol{u}_h, p_h, T_h) \in V_h \times Q_h \times X_h$ such that

$$\begin{bmatrix} \boldsymbol{D_u} + \boldsymbol{C}(\boldsymbol{u}_h) & \boldsymbol{B}^\intercal & 0 \\ \alpha \boldsymbol{B} & 0 & 0 \\ 0 & 0 & \boldsymbol{A}(\boldsymbol{u}_h) + \boldsymbol{D}_T \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_h \\ p_h \\ T_h \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}(T_h) \\ 0 \\ G \end{bmatrix}, \tag{2.21}$$

with, for any $1 \leq i, j \leq \mathcal{N}_{\boldsymbol{u}}$, $1 \leq m \leq \mathcal{N}_p$, $1 \leq k, l \leq \mathcal{N}_T$,

$$\begin{aligned} &(\boldsymbol{D_u})_{ij} = d_u(\zeta_j, \zeta_i), \qquad (\boldsymbol{C}(\boldsymbol{u}_h))_{ij} = c(\zeta_j, \zeta_i; \boldsymbol{u}_h), \qquad (\boldsymbol{F}(T_h))_i = f(\zeta_i; T_h), \\ &(\boldsymbol{B})_{mj} = b(\zeta_j, \eta_m) \\ &(\boldsymbol{D}_T)_{kl} = d_T(\xi_l, \xi_k), \qquad (\boldsymbol{A}(\boldsymbol{u}_h))_{kl} = a(\xi_l, \xi_k; \boldsymbol{u}_h), \qquad (\boldsymbol{G})_k = g(\xi_k). \end{aligned} \tag{2.22}$$

Dirichlet Boundary Conditions

As explained in the previous paragraph, we usually strongly impose the Dirichlet boundary conditions. We modify the algebraic structures after the assembly of the system (2.21) to force the value on the Dirichlet boundaries.

We introduce the application $\varphi_u : [1 : \mathcal{N}_{\boldsymbol{u}}] \to \Omega$ such that for any $i \in [1 : \mathcal{N}_{\boldsymbol{u}}]$, $\boldsymbol{x} = \varphi_u(i)$ is the point in Ω corresponding to the i-th DoF. We can now define $\mathcal{I}_{\Gamma_D} = \{i \in [1 : \mathcal{N}_{\boldsymbol{u}}], \varphi_u(i) \in \Gamma_D\}$. The entries of the matrix $\boldsymbol{M} = \boldsymbol{D_u} + \boldsymbol{C}(\boldsymbol{u}_h)$ and the entries of the vector $\boldsymbol{F}(T_h)$ are then modified such that

$$\begin{aligned} (\boldsymbol{M})_{ij} &= 0, & \forall i, j \in \mathcal{I}_{\Gamma_D}, i \neq j \\ (\boldsymbol{M})_{ii} &= 1, & \forall i \in \mathcal{I}_{\Gamma_D}, \\ (\boldsymbol{F})_i &= \boldsymbol{u}_\Gamma(\varphi_u(i)), & \forall i \in \mathcal{I}_{\Gamma_D}. \end{aligned} \tag{2.23}$$

We apply the same process in order to impose the value T_Γ of temperature field on the boundary Γ_D^T.

Nevertheless, we also implemented a weak formulation to compare the results of the two methods. The weak formulation does not set any constraint on the function spaces, and use Nitsche's penalization methods to impose the Dirichlet conditions through the variational formulation. When imposing the Dirichlet conditions weakly, we add the following terms to the variational formulation:

$$-\int_{\Gamma_D^F} \boldsymbol{T}(\boldsymbol{u}_h, p_h)\boldsymbol{n} \cdot \boldsymbol{v}_h + (\boldsymbol{u}_h - \boldsymbol{u}_D) \cdot \boldsymbol{T}(\boldsymbol{v}_h, q_h)\boldsymbol{n} + \gamma_h^F (\boldsymbol{u}_h - \boldsymbol{u}_D) \cdot \boldsymbol{v}_h \, d\Gamma \; = 0, \tag{2.24}$$

$$-\int_{\Gamma_D^T} \kappa \nabla T_h \cdot \boldsymbol{n} S_h + \kappa (T_h - T_D) \nabla S_h \cdot \boldsymbol{n} + \gamma_h^T (T_h - T_D) S_h \, d\Gamma \; = 0, \tag{2.25}$$

where $\gamma_h^T > 0, \gamma_h^F > 0$ are suitable constants defined by elements.

Slip Boundary Conditions

The slip boundary conditions are mainly required for the wall functions of the turbulence models, see chapter 4. We can strongly impose this kind of condition when the concerned boundary is parallel to the axis. In this particular case, we can edit the algebraic structures as we do for the Dirichlet conditions. However, in a general context - on curved boundaries - the application of slip conditions is not trivial.

Kuzmin *et al.* proposed in [59] an algebraic version but this technique requires an additional linear resolution and it was not appropriate to our solving strategy. As an alternative solution, we initially implemented a penalization formulation method following the design presented in [100]. This technique is close to the Nitsche's approach and is very convenient since it only adds extra terms in the variational formulation. Unfortunately, the convergence rate of this method is very low, and we preferred looking for alternative solutions. We nevertheless detail this formulation, and we present our convergence study for Stokes problem in the appendix 2.

After this unfruitful effort, we found two options. Dione proposes the first one in his thesis work [27]. In this review of the penalization method, he suggested some modifications to boost the convergence rate. Those adjustments mainly aim to

improve the quality of the geometric approximation either with a continuation of the normal-to-the-face vector field or with high-order elements on the curved boundaries. This approach was interesting, but we finally chose to implement the alternative method proposed in [100] using Lagrange multipliers. The formulation is a bit more constraining as it requires introducing a new FE space for the Lagrange multiplier, but it significantly improves the accuracy of the results.

For any point of the boundary $\partial\Omega$, we note $\boldsymbol{n}$ the outgoing unit normal vector to $\partial\Omega$. We also introduce, for any vector $\boldsymbol{a}$, its normal and tangential components respectively denoted by $(\boldsymbol{a})_{\boldsymbol{n}}$ and $(\boldsymbol{a})_{\boldsymbol{\tau}}$.

$$\begin{aligned}(\boldsymbol{a})_{\boldsymbol{n}} &= (\boldsymbol{n}\cdot\boldsymbol{a})\boldsymbol{n}\\ (\boldsymbol{a})_{\boldsymbol{\tau}} &= \boldsymbol{a} - (\boldsymbol{a})_{\boldsymbol{n}}\end{aligned} \tag{2.26}$$

We define the Lagrange multiplier space Λ_h, which is a $\mathbb{P}^0$ discontinuous space on the boundary Γ_S. The fluid part of the variational formulation (2.19) is completed into: find $(\boldsymbol{u}_h, p_h, \xi_h) \in V_h \times Q_h \times \Lambda_h$ such that

$$\mathcal{B}_{\gamma,h}((\boldsymbol{u}_h, p_h, \xi_h), (\boldsymbol{v}_h, q_h, \lambda_h)) = \mathcal{L}_{\gamma,h}(\boldsymbol{v}_h, q_h, \lambda_h), \quad \forall (\boldsymbol{v}_h, q_h, \lambda_h) \in V_h \times Q_h \times \Lambda_h, \tag{2.27}$$

where

$$\begin{aligned}\mathcal{B}_{\gamma,h}((\boldsymbol{u}_h, p_h, \xi_h), (\boldsymbol{v}_h, q_h, \lambda_h)) = &\int_\Omega (\boldsymbol{u}_h\cdot\nabla\boldsymbol{u}_h)\cdot\boldsymbol{v}_h \, d\Omega \; + \int_\Omega 2\mu\boldsymbol{S}(\boldsymbol{u}_h):\boldsymbol{S}(\boldsymbol{v}_h)\, d\Omega\\ &- \int_\Omega p_h\nabla\cdot\boldsymbol{v}_h \, d\Omega \; - \alpha\int_\Omega (\nabla\cdot\boldsymbol{u}_h) q_h \, d\Omega\\ &+ \int_\Gamma \xi_h\boldsymbol{v}_h\cdot\boldsymbol{n} \, d\Gamma \; + \int_\Gamma \boldsymbol{u}_h\cdot\boldsymbol{n}\lambda_h \, d\Gamma\\ &- \gamma\int_\Gamma h_f(\xi_h + \boldsymbol{n}\boldsymbol{T}(\boldsymbol{u}_h, p_h)\boldsymbol{n})(\lambda_h + \epsilon\boldsymbol{n}\boldsymbol{T}(\boldsymbol{v}_h, q_h)\boldsymbol{n})\, d\Gamma \;,\end{aligned} \tag{2.28}$$

and

$$\mathcal{L}_{\gamma,h}(\boldsymbol{v}, q, \lambda) = \int_\Omega \boldsymbol{f}\cdot\boldsymbol{v}_h \, d\Omega \; + \int_\Gamma \boldsymbol{g}_{\boldsymbol{\tau}}\cdot(\boldsymbol{v}_h)_{\boldsymbol{\tau}} \; d\Gamma \tag{2.29}$$

where γ is a positive constant and $\epsilon = 1, -1$ and h_f is the characteristic length of the current face. The values of γ and ϵ will be discussed in the further numerical tests in the section 4.2.

3 Solution Strategy

The Galerkin projection on the FE spaces defined in the previous section requires the resolution of the non-linear system (2.21). The actual resolution of such systems might be non-trivial for different reasons:

- The numerical instability of the model: this issue is prevalent when using the FEM to solve convection dominated problems. We usually overcome this problem by adding stabilization terms in the discrete formulation. The chapter 3 is dedicated to such techniques.

- The choice of the resolution scheme: there are many possibilities to treat the non-linearity of the system. We present and discuss some of these options in the present section.

- The preconditioning of the system: this point might not be an issue when the size of the problem remains reasonable (typically in 2D). In that case, a direct resolution is usually achievable using, for instance, an LU decomposition. However, in 3D we need proper preconditioning strategies to succeed solving large linear systems. This issue is directly related to the choice of the resolution scheme.

During this thesis work, we mainly considered two iterative methods for the resolution of non-linear problems: the Picard algorithm (or fixed point algorithm) and the Newton algorithm. The Newton method theoretically presents an attractive quadratic convergence rate. However, the algorithm is very sensitive to the initial guess, and the convergence radius decrease very fast for high Reynolds number. This point rapidly became an issue, and we had to find new solutions. We implemented the Picard algorithm as an alternative. This method has slow convergence rate compared to the Newton algorithm however, it is usually more robust and less sensitive to the initial guess. The Picard algorithm offered a new option but was still unsuccessful to solve aerothermal problems for high Reynolds numbers.

At this point, we decided to implement a transient version of our model. Our initial motivation was to understand the evolution of the flow when the Reynolds number was growing. But we finally considered the transient resolution as a way to reach the stationary state. We tested different time integration algorithms. The use of the standard backward differentiation formulas, present in Feel++, produced a considerable number of iterations to reach the stationary state. Then we started looking at adaptive time-step, and we implemented the algorithm proposed by Kay *et al.*in [54, 38]. We give implementation details and a review of the method in the appendix 1. Unfortunately, as illustrated in our numerical results, the technique was not entirely stable. We finally dropped the idea of using transient resolution, and we start studying the pseudo time-step methodology.

In the current section, we briefly remind the mentioned iterative methods and the different resolution schemes we envisaged. Then we will detail the pseudo-transient continuation and discuss its advantages. We dedicate the last part to the different preconditioning strategies.

3.1 Non-Linear Iterative Methods

Picard Algorithm

We consider the generic non linear discrete problem: find $\boldsymbol{w} \in \mathbb{R}^{\mathcal{N}}$ such that

$$\boldsymbol{M}(\boldsymbol{w})\boldsymbol{w} = \boldsymbol{F}(\boldsymbol{w}), \tag{2.30}$$

where $\boldsymbol{M} : \mathbb{R}^{\mathcal{N}} \to \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ and $\boldsymbol{F} : \mathbb{R}^{\mathcal{N}} \to \mathbb{R}^{\mathcal{N}}$. This problem can be solved using Picard iterations. From an initial guess $\boldsymbol{w}^0$, we build a sequence $(\boldsymbol{w}^n)$, hopefully convergent such that $\boldsymbol{w}^n \to \boldsymbol{w}$. For any $n \geq 0$, assuming that $\boldsymbol{w}^n$ is known, $\boldsymbol{w}^{n+1}$ is computed by solving the linear system

$$\boldsymbol{M}(\boldsymbol{w}^n)\boldsymbol{w}^{n+1} = \boldsymbol{R}(\boldsymbol{w}^n). \tag{2.31}$$

The Picard algorithm presents a wider radius of convergence than the Newton algorithm. However, this method has potentially very slow convergence rate and is very sensitive to local minima.

Newton Algorithm

We consider the generic non-linear discrete system: find $\boldsymbol{w} \in \mathbb{R}^{\mathcal{N}}$ such that

$$\mathcal{F}(\boldsymbol{w}) = 0 \tag{2.32}$$

where $\mathcal{F} : \mathbb{R}^{\mathcal{N}} \to \mathbb{R}^{\mathcal{N}}$ is supposed to be differentiable. Here again we iteratively build a sequence $(\boldsymbol{w}^n)$ such that $\boldsymbol{w}^n \to \boldsymbol{w}$, from an initial guess $\boldsymbol{w}^0$. For any $n \geq 0$, assuming that $\boldsymbol{w}^n$ is known, we set $\boldsymbol{w}^{n+1} = \boldsymbol{w}^n + \delta^n$, with δ^n solution of the linear system

$$\mathcal{F}'(\boldsymbol{w}^n)\delta^n = -\mathcal{F}(\boldsymbol{w}^n), \tag{2.33}$$

where $\mathcal{F}'$ is the Jacobian matrix of $\mathcal{F}$. This method as a theoretical quadratic convergence, but is very sensitive to the choice of the initial guess $\boldsymbol{w}^0$. Local minima of the function $\mathcal{F}$ might also be an issue.

Resolutions Schemes

These two algorithms work with the same idea: splitting the non-linear resolution in multiple linear resolutions. This technique might be very costly if the algorithm is not able to quickly converge. In both cases, the choice of an appropriate initial guess is crucial since it affects both the capability to converge and the number of required iterations.

Once we dispose of these iterative solvers, we have to decide if we want to treat the system (2.21) in a monolithic way or by splitting the problem. The monolithic

resolution consists in the direct application of an iterative solver to the non-linear system

$$\begin{bmatrix} \boldsymbol{D_u} + \boldsymbol{C}(\boldsymbol{u}_h) & \boldsymbol{B}^\intercal & 0 \\ \alpha \boldsymbol{B} & 0 & 0 \\ 0 & 0 & \boldsymbol{A}(\boldsymbol{u}_h) + \boldsymbol{D}_T \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_h \\ p_h \\ T_h \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}(T_h) \\ 0 \\ \boldsymbol{G} \end{bmatrix}. \tag{2.34}$$

The decoupled resolution consists in fixed point iterations between the resolution of the non-linear system (using our favorite iterative method)

$$\begin{bmatrix} \boldsymbol{D_u} + \boldsymbol{C}(\boldsymbol{u}_h) & \boldsymbol{B}^\intercal \\ \alpha \boldsymbol{B} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_h \\ p_h \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}(T_h) \\ 0 \end{bmatrix} \tag{2.35}$$

and the linear one

$$\left[\boldsymbol{A}(\boldsymbol{u}_h) + \boldsymbol{D}_T\right] \left[T_h\right] = \left[\boldsymbol{G}\right], \tag{2.36}$$

The different options were implemented and tested. Each configuration has its advantages and will be chosen following the needs. For instance, the decoupled version is particularly efficient for transient resolution. In this particular case, we can use an Oseen linearization for the fluid system to run quick simulations. On the other hand, this decoupled scheme is hardly practicable for steady resolution since we have to manage the non-linearity between $\boldsymbol{u}$ and p and then between the fluid and T. When we finally decided to only work with the steady state directly, we abandoned the decoupled resolution.

Continuation

The remaining issue was the choice of the initial guess. Our framework was still unable to reach the solution for more dynamic problems (smaller viscosity and diffusivity). To improve the convergence of our solver, we envisaged two solutions: a physical continuation and a transient continuation. We finally implemented a mixed solution in which both kind of continuation can be used.

The physical continuation iteratively solves the system with evolving parameters. Each new resolution uses the previous solution as an initial guess. In practice, the continuation is made on the viscosity μ and the thermal diffusivity κ. For a continuation of N steps, we will successively solve

$$\mathcal{F}(\boldsymbol{u}^k, p^k, T^k; \mu^k, \kappa^k) = 0 \tag{2.37}$$

where $\mu^k = \varepsilon(k)\mu$ and $\kappa = \varepsilon(k)\kappa$, $\varepsilon(N) = 1$. There are many possibilities for the design of the continuation function ε. We chose a log-equidistributed partitioning of the viscosity range with

$$\varepsilon(k) = \frac{\mu^{k/N}}{\mu}. \tag{2.38}$$

This technique is very efficient and particularly stable for high Reynolds simulations. However, we usually need a well tuned continuation to guaranty the convergence.

To remedy this issue, we implemented the pseudo-transient continuation which remarkably improves the convergence of the Newton algorithm. This technique allows to remove the physical continuation or, at least, to only consider few steps.

3.2 Pseudo Transient Continuation

The pseudo-transient continuation [55, 19] is a modification of the Newton algorithm to improve the convergence capabilities of the method. The continuation is made by imposing local transient states to the system, to progressively reach the steady state.

We consider a Differential Algebraic Equation

$$\mathcal{F}(\boldsymbol{w}) = 0 \tag{2.39}$$

resulting of a FEM discretization. In a generic formulation, we suppose $\boldsymbol{w} \in \mathcal{N}$ and $\mathcal{F} : \mathbb{R}^{\mathcal{N}} \to \mathbb{R}^{\mathcal{N}}$ differentiable. The non-linear system $\mathcal{F}(\boldsymbol{w}) = 0$ has to be solved using an iterative method of successive linear resolutions. For such techniques, the choice of the initial guess is crucial. For instance, for the Navier-Stokes system, for a low Reynolds number, a common initialization is to start by resolving the associated stokes problem. For larger Reynolds numbers, the convergence radius of iterative methods becomes smaller, and we have to choose a better initial guess.

The pseudo-transient continuation (Ψtc) method is particularly adapted for problems deriving from a transient dynamic. From the generic stationary equation (2.39), we can define the transient problem associated

$$\boldsymbol{w}' + \mathcal{F}(\boldsymbol{w}) = 0, \boldsymbol{w}(0) \quad = \boldsymbol{w}_0, \tag{2.40}$$

where $\boldsymbol{w}' = \frac{\partial \boldsymbol{w}}{\partial t}$. Then, it is natural to seek a root of $\mathcal{F}$ as the limit of $\boldsymbol{w}(t)$

$$\boldsymbol{w} = \lim_{t \to \infty} \boldsymbol{w}(t). \tag{2.41}$$

When we are not interested in the transient states, we can reach this limit faster by considering non-physic intermediary states of the problem. This idea is the main ingredient of the pseudo-transient continuation.

In [55], Kelley *et al.*propose a continuation algorithm that integrates iteratively

$$\boldsymbol{V}\boldsymbol{w}' = \mathcal{F}(\boldsymbol{w}), \quad \boldsymbol{w}(0) = \boldsymbol{w}_0 \tag{2.42}$$

by a variable time step δ_n. This time step is supposed to grow as the residual $\mathcal{F}(\boldsymbol{w}_n)$ approaches zero. Here $\boldsymbol{V}$ is a nonsingular matrix used to improve the scaling of the problem and usually build to balance the local CFL number through the domain.

The convergence results for Ψtc in [55] are extended in [19] for systems in which it is more convenient to not parabolize all the equations. For instance, in

: Newton Algorithm

Choose ε;
Set $\boldsymbol{w} = \boldsymbol{w}_0$;
repeat
  Solve $\mathcal{F}'(\boldsymbol{w})s = -\mathcal{F}(\boldsymbol{w})$;
  Set $\boldsymbol{w} = \boldsymbol{w} + s$;
  Evaluate $\mathcal{F}(\boldsymbol{w})$;
until $||\mathcal{F}(\boldsymbol{w})|| < \varepsilon$;

: Ψtc Algorithm

Choose ε;
Set $\boldsymbol{w} = \boldsymbol{w}_0$ and $\delta = \delta_0$;
repeat
  Solve $(\frac{1}{\delta}D + \mathcal{F}'(\boldsymbol{w}))s = -\mathcal{F}(\boldsymbol{w})$;
  Set $\boldsymbol{w} = \boldsymbol{w} + s$;
  Evaluate $\mathcal{F}(\boldsymbol{w})$;
  Update δ;
until $||\mathcal{F}(\boldsymbol{w})|| < \varepsilon$;

Figure 2.1 – Comparison of Ψtc and Newton algorithms

the incompressible Navier-Stokes problem, it is entirely natural only to evolve the velocity through pseudo-time stepping. It results in the equations,

$$\boldsymbol{D}\boldsymbol{w}' = -\mathcal{F}(\boldsymbol{w}), \tag{2.43}$$

with

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{V} & \\ & 0 \end{bmatrix} \tag{2.44}$$

with $\boldsymbol{V}$ a non-singular matrix. The design of this matrix will be discussed in a further paragraph.

The procedure of Ψtc is very close to the usual Newton method, but we add the matrix $\frac{1}{\delta_n}\boldsymbol{D}$ to the Jacobian matrix, $\mathcal{F}'$, of the function $\mathcal{F}$. The pseudo transient iteration then reads

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - (1/\delta_n \boldsymbol{D} + \mathcal{F}'(\boldsymbol{w}_n))^{-1}\mathcal{F}(\boldsymbol{w}_n). \tag{2.45}$$

As a comparison we remind the Newton algorithm and present the Ψtc algorithm in figure 2.1.

Choice of parameter δ_n

An updating procedure for the pseudo time step δ is proposed in [55] using the Switched Evolution Relaxation. This method increases the time step in inverse proportion of the residual reduction.

$$\delta_{n+1} = \delta_n \frac{||\mathcal{F}(\boldsymbol{w}_{n-1})||}{||\mathcal{F}(\boldsymbol{w}_n)||} = \delta_0 \frac{||\mathcal{F}(\boldsymbol{w}_0)||}{||\mathcal{F}(\boldsymbol{w}_n)||} \tag{2.46}$$

In order to control the evolution of this pseudo time step, a bounding function ϕ is often added

$$\delta_{n+1} = \phi(\delta_n \frac{||\mathcal{F}(\boldsymbol{w}_{n-1})||}{||\mathcal{F}(\boldsymbol{w}_n)||}), \tag{2.47}$$

with

$$\phi(\xi) = \min(\xi, \delta_{\max}), \tag{2.48}$$

where $\delta_{\max}$ is a custom threshold parameter. When this bounding function is used, the system switches into a steady state resolution after some iterations. The initial value δ_0 is another custom parameter and has to be chosen depending of the dynamic of the problem.

Another version of the Switched Evolution Relaxation proposes not to consider the norm of the residual $||\mathcal{F}(\boldsymbol{w})||$ but the norm of the increment $||\boldsymbol{w}_n - \boldsymbol{w}_{n-1}||$. However, this method is more prone to convergence failures and requires a uniformly well conditioned Jacobian. The update of the time step then reads

$$\delta_{n+1} = \phi(\delta_n ||\boldsymbol{w}_n - \boldsymbol{w}_{n-1}||). \tag{2.49}$$

This alternative was not studied in this thesis work since the first version gives satisfying results.

The Scaling Matrix

The construction of the scaling matrix $\boldsymbol{V}$ is made to balance the local CFL number,

$$C_K = \Delta t \sum_{i=1}^{d} \frac{u_i}{h_i} \tag{2.50}$$

where Δt is the time step, u_i is the component of the velocity field h_i is the characteristic length of the cell in the i-eme direction of the element K. Hence it naturally appears to choose the scaling factor per-element

$$\omega_K = \frac{||\boldsymbol{u}||_2}{h} \tag{2.51}$$

where $||\boldsymbol{u}||_2$ is the usual norm-2 of the convection field on the center of the element and h is the characteristic length of the element. The evaluation of h is discussed in chapter 3.

4 Validation

In this section, we present the results obtained with our aerothermal framework on different test-cases. The following problems are usually fundamental and aim to validate our implementation, see chapter 8. By default the configuration is the following:

- $\mathbb{TH}^1_{ch}$ FE space for the fluid part,

- $\mathbb{L}^2_{ch}$ FE space for the temperature,
- domain meshed with linear simplex,
- unstructured homogeneous mesh of characteristic size h,
- monolithic resolution with Newton algorithm and eventually Ψtc continuation,
- LU direct solver in 2D.

This default configuration may be edited for certain test-cases, it will be specified where appropriate.

The common quantities of interest are the following errors (usually normalized):

- $\mathcal{L}^2$-error on the velocity: $\|\boldsymbol{u} - \boldsymbol{u}_h\|_0$,
- H^1-error on the velocity: $\|\boldsymbol{u} - \boldsymbol{u}_h\|_1$,
- $\mathcal{L}^2$-error on the pressure (up to an additive constant): $\|p - p_h\|_0$,
- $\mathcal{L}^2$-error on the temperature: $\|T - T_h\|_0$.

4.1 Validation of the Fluid Solver

Fluid Convergence Study, Kovasznay Test Case

We run a convergence study on the well known case proposed by Kovasznay in [58]. The geometry of the problem is a 2D grid, $\Omega = [-0.5, 2] \times [-0.5, 1.5]$. The analytic solution is given by

$$\begin{aligned} u &= (1 - e^{\lambda x}\cos(2\pi y), \frac{\lambda}{2\pi} e^{\lambda x}\sin(2\pi y)) \\ p &= p_0 - e^{\lambda x} \end{aligned} \tag{2.52}$$

where $\lambda = \frac{\mathrm{Re}}{2} - \sqrt{\frac{\mathrm{Re}^2}{4} + 4\pi^2}$, see figure 2.2. For the present study we fixed $\mathrm{Re} = \frac{\rho}{mu} = 40$. We impose the analytic solution as strong Dirichlet condition on the boundaries of the domain, and we treat the non-linearity with Newton algorithm. We measured the three errors : $\|\boldsymbol{u}_h - \boldsymbol{u}_e\|_0$, $\|\boldsymbol{u}_h - \boldsymbol{u}_e\|_1$ and $\|p_h - p_e\|_0$. The evolution of these quantities with respect to the characteristic size h is presented in figure 2.3. The study is performed with or without stabilization method (SUPG1$_d$, see chapter 3), for different polynomial orders : $\mathbb{TH}^1_h$, $\mathbb{TH}^2_h$ and $\mathbb{TH}^3_h$. We retrieve the expected convergence rates for $\boldsymbol{u}$ and p. We also notice that the use of stabilization method does not influence the results. It proves the consistency of the method: the stabilization terms vanish when the characteristic length of the mesh goes to zero. Those results are identical with other configuration for the stabilization method.

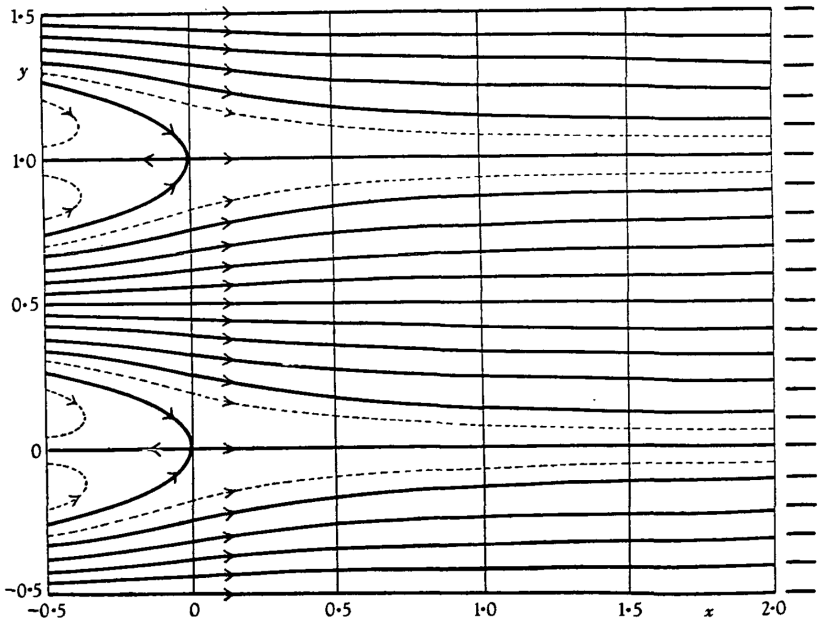
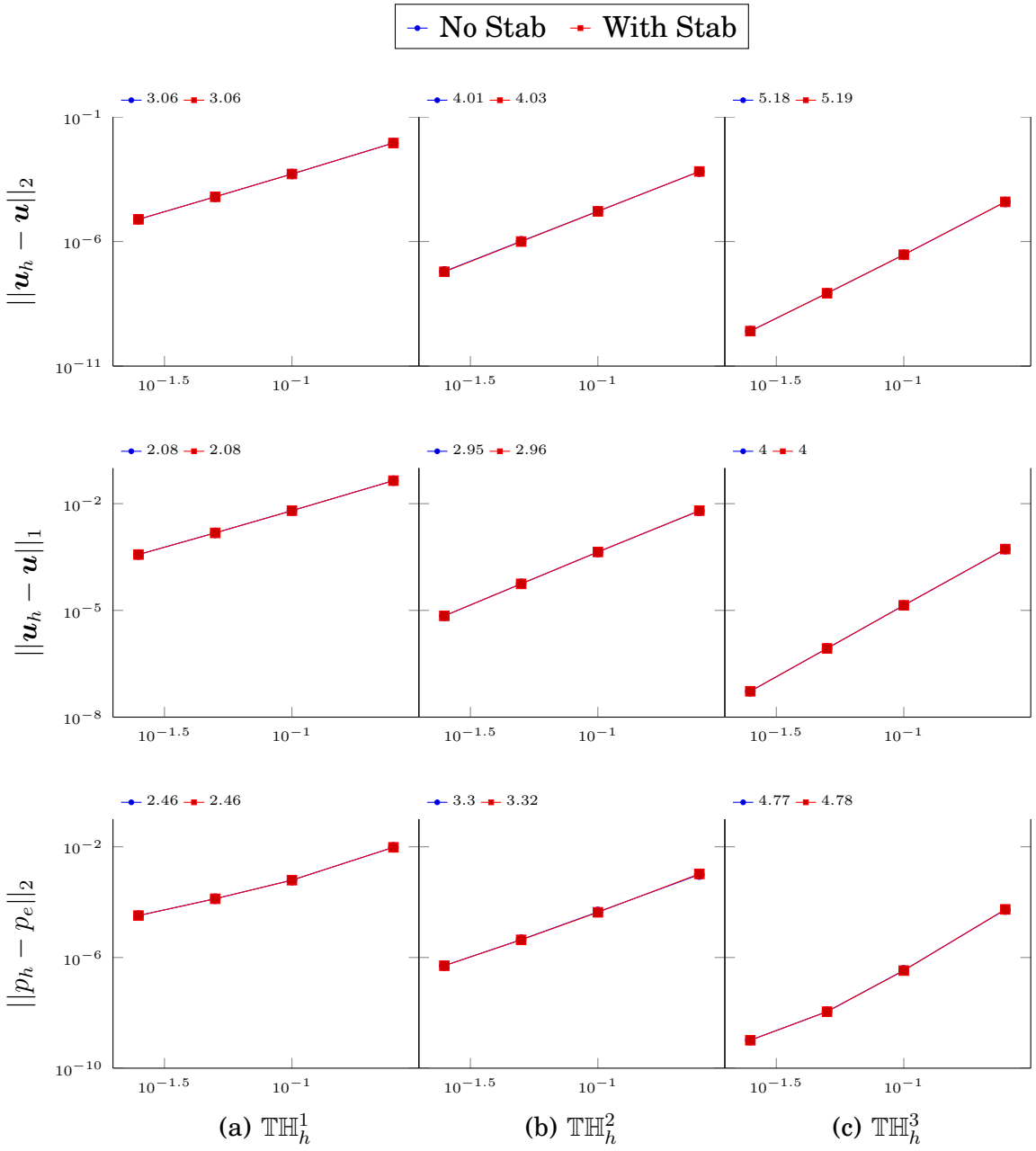


Figure 2.2 – Streamline of the analytic solution, capture from [58]



(a) $\mathbb{TH}_h^1$ (b) $\mathbb{TH}_h^2$ (c) $\mathbb{TH}_h^3$

Figure 2.3 – Kovasnzay test-case. Convergence study for different polynomial orders, with or without stabilization method. Log/Log scale.
Stab and No-Stab plots are perfectly stacked

Turek 2D Benchmark: Flow Around a Cylinder, Re=20

This benchmark is proposed on http://www.featflow.de. The geometry of the model is a pipe without a circular cylinder of radius $r = 0.05$, as described in figure 2.4. No-slip boundary conditions are imposed on upper and lower walls and

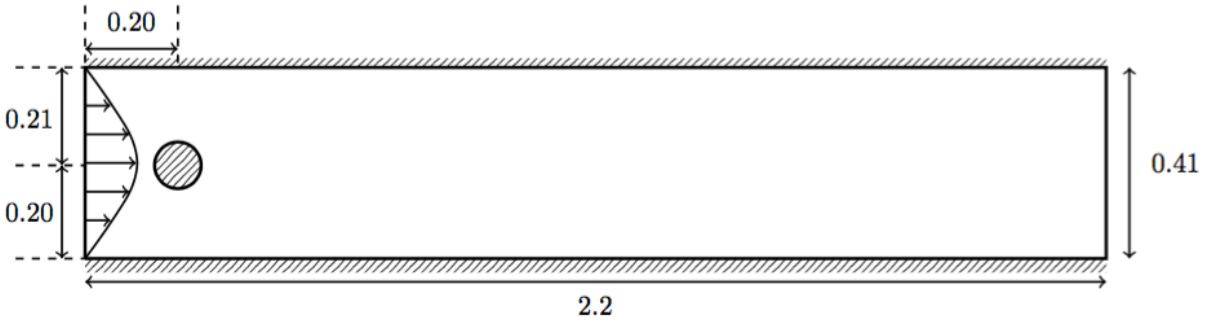


Figure 2.4 – Geometry of the Problem

on the boundary of the cylinder. The left edge is the input of the pipe where a Poiseuille profile is prescribed,

$$u(0, y) = (\frac{4U_y(0.41 - y)}{0.41^2}, 0) \tag{2.53}$$

with the maximum amplitude of the velocity $U_y = 0.3$. The right edge is the output of the pipe and with a do-nothing condition

$$\bar{\sigma} = 0. \tag{2.54}$$

The parameters of the fluid are set to $\rho = 1.0$ and $\mu = 0.001$. With this configuration, the Reynolds number obtained is $\mathrm{Re} = 20$ and the system turns into a stationary state.

In this benchmark, we are interested in three quantities. The lift and drag coefficients are respectively defined by

$$C_D = \frac{2}{\bar{U}^2 L} F_D, \qquad C_L = \frac{2}{\bar{U}^2 L} F_L \tag{2.55}$$

with $\bar{U} = 0.2$ the mean velocity at the input, $L = 0.1$ the characteristic length of the flow configuration and (F_D, F_L) such that

$$(F_D, F_L) = \int_{\Gamma_C} \bar{\sigma} \boldsymbol{n} d\Gamma \tag{2.56}$$

where Γ_C is the boundary of the cylinder and $\boldsymbol{n}$ is the outer normal vector of the circle. The last quantitity of interst is the pressure difference $\Delta_p = p(a_1) - p(a_2)$ between the points $a_1 = (0.15, 0.2)$ and $a_2 = (0.25, 0.2)$. We will compare our results with the reference values: $C_D = 5.57953523384$, $C_L = 0.010618948146$ and $\Delta_p = 0.11752016697$.

We tested the framework with or without stabilization method (SUPG1$_d$,see chapter 3). The results obtained with our fluid solver are coherent with the reference values, see table 2.5.

	No Stab		With Stab	
C_D	5.5769	[0.047]	5.5771	[0.043]
C_L	0.010629	[0.100]	0.010630	[0.110]
Δ_p	0.11751	[0.008]	0.11751	[0.007]

Figure 2.5 – Turek benchmark: flow around a cylinder, $\mathrm{Re} = 20$. Numerical results for the drag coefficient, lift coefficient and pressure difference. In bracket: relative error in %, compared with the reference values.

4.2 Validation of the Slip Boundary Condition Implementation

We present, in this subsection, a convergence study for two problems with slip conditions: one with a straight boundary and the second one with a curved boundary. The principle will be identical for both test-cases: we solve the Navier-Stokes equations on a closed domain, where a manufactured function is imposed on the boundary, using slip conditions. We can then compare the numerical solutions with the exact solutions and we evaluate the quantities: $\|\boldsymbol{u}_h - \boldsymbol{u}_e\|_0$, $\|\boldsymbol{u}_h - \boldsymbol{u}_e\|_1$ and $\|p_h - p_e\|_0$. For each problem, we display the evolution of these errors as a function of the characteristic size of the mesh. To determine the best configuration, we performed simulations for different values of the constants α, γ and ϵ. We remind the role of these constants:

- $\alpha = \pm 1$ determine the treatment of the pressure/velocity term in the Navier-Stokes weak formulation (symmetric or skew-symmetric), see (2.16),
- γ is a positive penalty constant used in the stabilized formulation for slip boundary conditions, see (2.28)
- $\epsilon = \pm 1$ determine the form of the stabilization terms in this formulation for slip boundary conditions: symmetric or skew-symmetric, see (2.28).

We also performed one simulation with strongly-imposed Dirichlet conditions to compare the convergence rates.

Slip Conditions on a Square

The domain is a square $\Omega = [-1, 1] \times [-1, 1]$. The viscosity is supposed to be constant $\mu = 1$. The manufactured solution is

$$\begin{aligned} \boldsymbol{u} &= (2y(1-x^2), -2x(1-y^2)), \\ p &= x^2 + y^2. \end{aligned} \tag{2.57}$$

With this solution we can evaluate the slip condition on the boundary

$$\begin{aligned} \boldsymbol{u} \cdot \boldsymbol{n} &= 0, \\ (\boldsymbol{T}(\boldsymbol{u}, p)\boldsymbol{n})_{\boldsymbol{\tau}} &= (2y(1-x^2), -2x(1-y^2)). \end{aligned} \tag{2.58}$$

The source term is chosen to fit with this solution. The discrete solution is computed for different values of $h = 0.2, 0.1, 0.05, 0.025$. The method only converges when $\epsilon = \alpha$, so we presented results for $(\alpha, \epsilon) = (1, 1)$ and $(\alpha, \epsilon) = (-1, -1)$. We also tested different values for $\gamma = 10^{-4}, 10^{-2}, 1, 10^2, 10^4$. The convergence study is summarized on figure 2.6.

Slip Conditions on a Ring

The second test-case is made on a ring, defined by $1 \leq x^2 + y^2 \leq 4$. The exact solution

$$\begin{aligned} \boldsymbol{u} &= (-y\sqrt{x^2+y^2}, x\sqrt{x^2+y2}), \\ p &= x^2 + y^2, \end{aligned} \tag{2.59}$$

is imposed with a Dirichlet condition on the internal boundary ($x^2 + y^2 = 1$) and with the slip condition

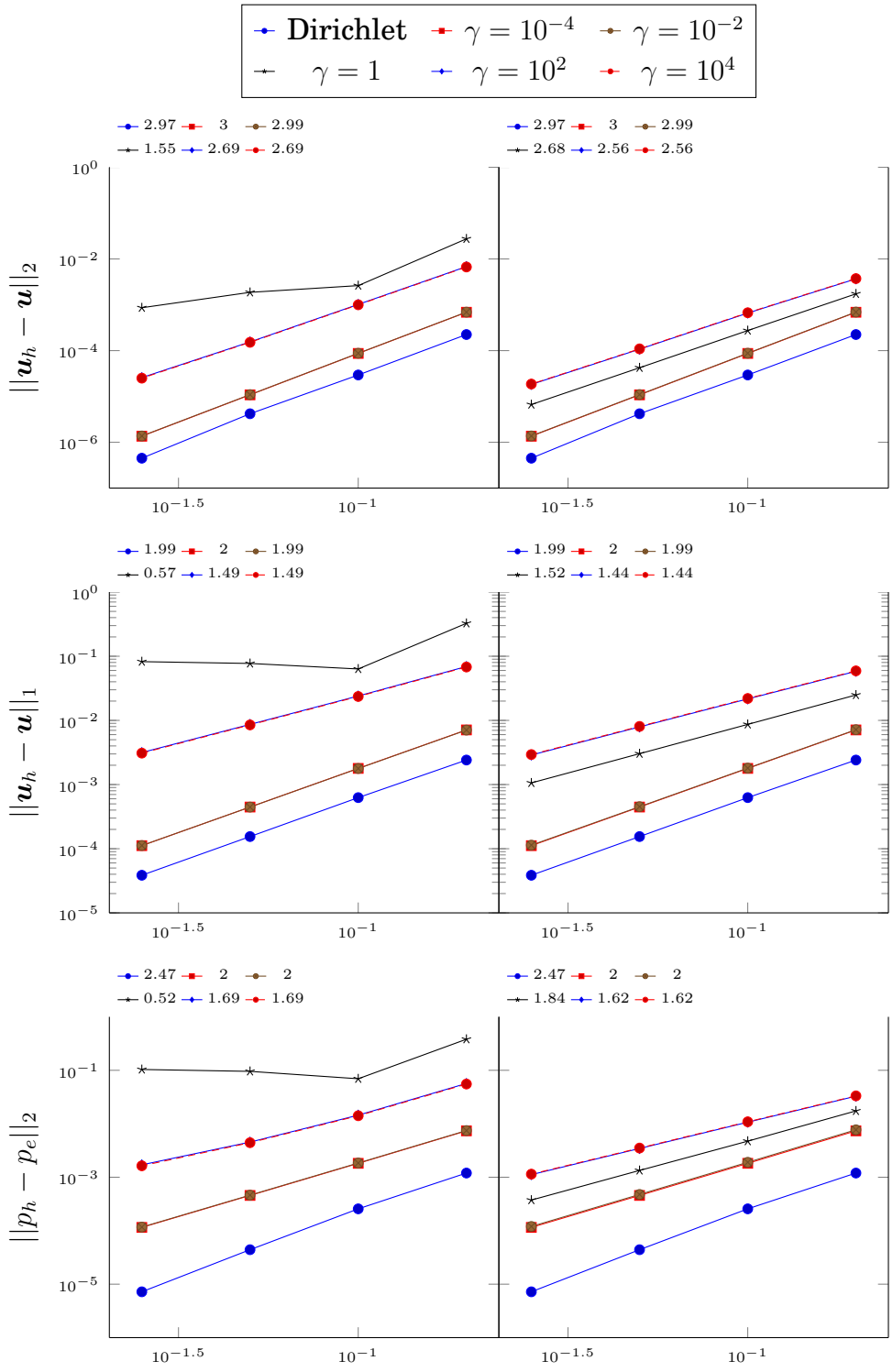
$$\begin{aligned} \boldsymbol{u} \cdot \boldsymbol{n} &= 0, \\ (\boldsymbol{T}(\boldsymbol{u}, p)\boldsymbol{n})_{\boldsymbol{\tau}} &= (-2y, 2x), \end{aligned} \tag{2.60}$$

on the external boundary ($x^2 + y^2 = 4$). Again the source term f is designed to obtain the exact solution in the domain. The results for this test-case can be seen on figure 2.7.

Considering the results of the figures 2.6 and 2.7, we observe that:

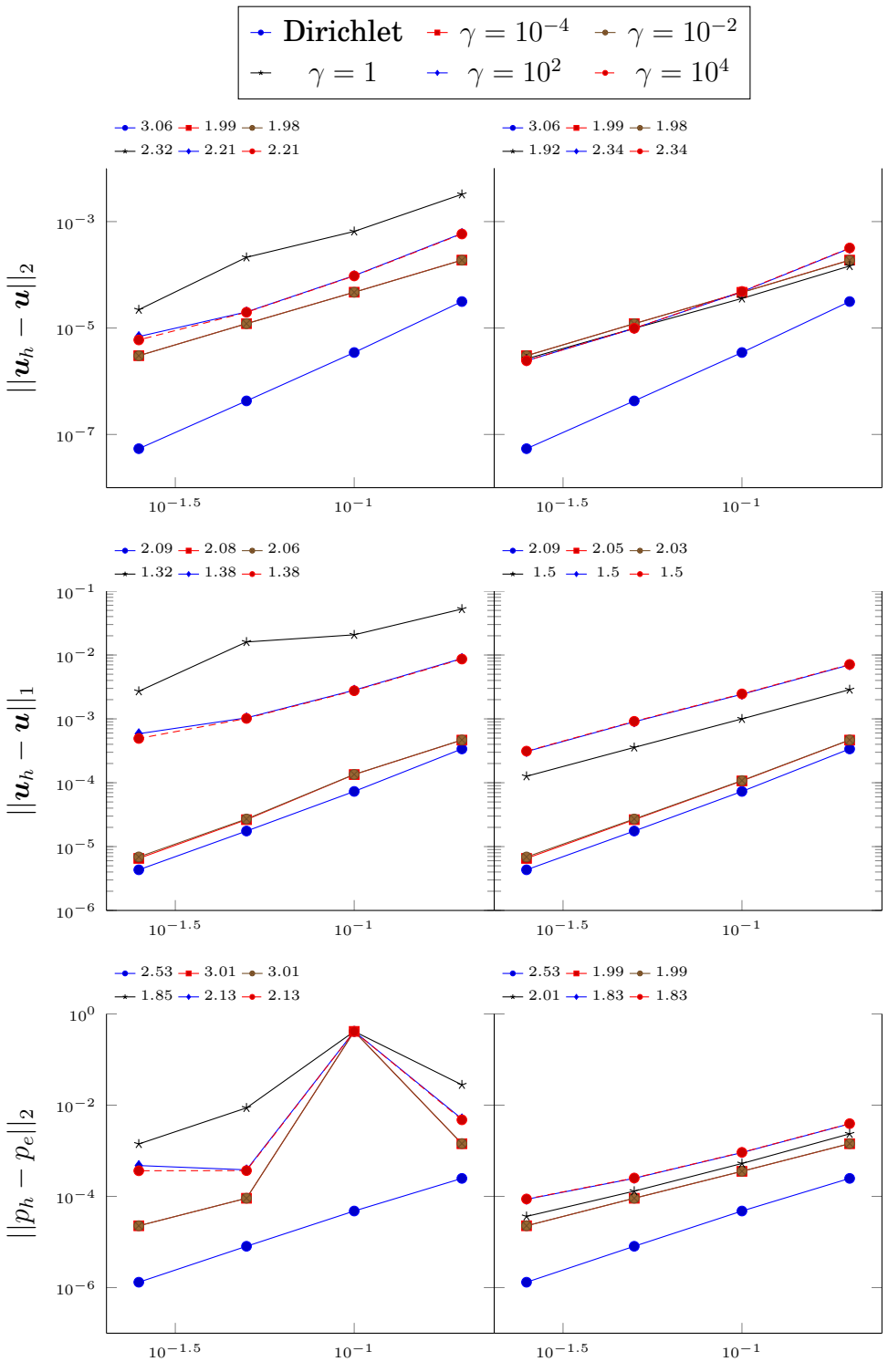
- as announced in [100], the version with $(\alpha, \epsilon) = (-1, -1)$ is more robust, and seems to be unconditionally stable,
- the convergence is globally better for small values of γ,
- we almost retrieve the theoretical convergence rate, except for the $\mathcal{L}^2$ error on $\boldsymbol{u}$, on the ring,
- with a suited configuration, the method is as efficient for curved or straight boundary.

From those observations, we can choose our favorite setup for later simulations with slip boundary conditions: $\gamma = 10^{-4}$, $\alpha = -1$ and $\epsilon = -1$. These values will be used by default. For that reason, the parameter α will not appear in the variational formulation anymore: we always use the anti-symmetric formulation.



(a) $\alpha = 1, \delta = 1$ (b) $\alpha = -1, \delta = -1$

Figure 2.6 – Slip conditions applied on a squared geometry.
Convergence study of the errors between exact and FE solutions as function of the characteristic length h. Log/Log scale. Linear regression and slope displayed for different values of α, γ and δ.



(a) $\alpha = 1, \delta = 1$ (b) $\alpha = -1, \delta = -1$

Figure 2.7 – Slip conditions applied on a ring geometry.
Convergence study of the errors between exact and FE solutions as function of the characteristic length h. Log/Log scale. Linear regression and slope displayed for different values of α, γ and δ.

4.3 Validation of the Aerothermal Solver

The Buoyancy Driven Cavity

We test here our aerothermal framework on the closed insulated heated cavity benchmark. The domain is a square $\Omega = [0,1] \times [0,1]$, the left and right walls have fixed temperature, respectively to $T = 1$ and $T = 0$. The bottom and top walls are supposed to be adiabatic. We also impose a no-slip boundary condition, $\boldsymbol{u} = (0,0)$, on the whole boundary. This setup is summarized in figure 2.8. We solved the aerothermal problem with different parameters. For this benchmark, the aerothermal system is usually written in a dimensionless formalism and the regime can be characterized using the Rayleigh number,

$$\mathrm{Ra} = \frac{\boldsymbol{g}\beta}{\nu\kappa}\Theta L^3, \tag{2.61}$$

where ν, $\boldsymbol{g}$, β and κ are the physical parameters defined in previously, Θ is the characteristic temperature difference and L is the characteristic length of the problem.

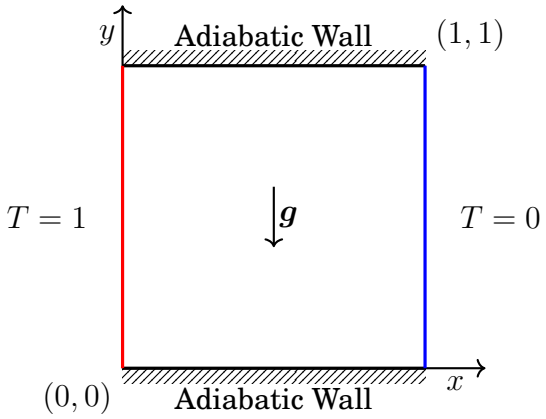


Figure 2.8 – Buoyancy Driven Cavity Setup

We performed simulations for $10^3 \leq \mathrm{Ra} \leq 10^8$, and we measured the following quantities:

- the maximum vertical velocity at the mid-height (value and location), table 2.1,
- the maximum horizontal velocity at the mid-width (value and location), table 2.2,
- the maximum Nusselt number on the hot wall (value and location), table 2.3,
- the averaged Nusselt number on the hot wall, table 2.4.

Ra	Present	Ref.[106]	Ref.[25]	Ref.[67]	Ref.[69]
10^3	3.697 (0.178)	3.686 (0.188)	3.679 (0.179)	3.73 (0.182)	3.696 (0.179)
10^4	19.63 (0.118)	19.79 (0.120)	19.51 (0.120)	19.90 (0.124)	19.61 (0.119)
10^5	68.59 (0.068)	70.63 (0.072)	68.22 (0.066)	70.0 (0.068)	68.69 (0.067)
10^6	221.1 (0.038)	227.1 (0.040)	216.8 (0.039)	228.0 (0.039)	220.8 (0.038)
10^7	697.6 (0.022)	714.5 (0.022)	-	698.0 (0.024)	703.3 (0.022)
10^8	2206 (0.12)	2259 (0.012)	-	-	2223 (0.013)

Table 2.1 – Buoyancy Driven Cavity. Maximum vertical velocity ($\boldsymbol{u}_y$) at the mid-height ($y = 0.5$). Value (x position) for different Rayleigh numbers

Ra	Present	Ref.[106]	Ref.[25]	Ref.[67]	Ref.[69]
10^3	3.649 (0.812)	3.489 (0.813)	3.634 (0.813)	3.680 (0.817)	3.649 (0.813)
10^4	16.18 (0.822)	16.12 (0.815)	16.20 (0.823)	16.10 (0.817)	16.18 (0.824)
10^5	34.73 (0.852)	33.39 (0.835)	34.81 (0.855)	34.00 (0.857)	34.77 (0.854)
10^6	64.82 (0.848)	65.40 (0.860)	65.33 (0.851)	65.40 (0.875)	64.69 (0.846)
10^7	148.6 (0.878)	143.6 (0.922)	-	139.7 (0.919)	145.3 (0.885)
10^8	321 (0.927)	296.71 (0.930)	-	-	283.7 (0.946)

Table 2.2 – Buoyancy Driven Cavity. Maximum horizontal velocity ($\boldsymbol{u}_x$) at the mid-width ($x = 0.5$). Value (y position) for different Rayleigh numbers

Our results are compared with reference values from [106], [25], [67] and [69]. We can reasonably conclude in the accuracy of our aerothermal framework. The measured values perfectly fit with the references, even for high Rayleigh numbers ($\mathrm{Ra} \geq 10^7$) when the resolution is definitely not trivial.

Ra	Present	Ref.[106]	Ref.[25]	Ref.[67]	Ref.[69]
10^3	1.506 (0.088)	1.501 (0.08)	1.50 (0.092)	1.47 (0.109)	1.506 (0.090)
10^4	3.531 (0.142)	3.579 (0.13)	3.53 (0.143)	3.47 (0.125)	3.530 (0.143)
10^5	7.721 (0.082)	7.945 (0.08)	7.71 (0.08)	7.71 (0.08)	7.708 (0.084)
10^6	17.56 (0.038)	17.86 (0.03)	17.92 (0.038)	17.46 (0.039)	17.53 (0.038)
10^7	39.92 (0.018)	38.6 (0.015)	-	30.46 (0.024)	41.02 (0.039)
10^8	93.63 (0.008)	91.16 (0.010)	-	-	91.21 (0.067)

Table 2.3 – Buoyancy Driven Cavity. Maximum Nusselt number on the hot wall ($x = 0$). Value (y position) for different Rayleigh numbers.

Ra	Present	Ref.[106]	Ref.[25]	Ref.[67]
10^3	1.118	1.117	1.12	1.074
10^4	2.245	2.254	2.243	2.084
10^5	4.522	4.598	4.52	4.3
10^6	8.835	8.976	8.8	8.743
10^7	16.63	16.66	-	13.99
10^8	31.12	31.49	-	-

Table 2.4 – Buoyancy Driven Cavity. Averaged Nusselt number on the hot wall ($x = 0$). Value for different Rayleigh numbers.

Conclusion

In this chapter, we detailed the different techniques implemented for the resolution of an aerothermal problem:

- formulation: incompressible Navier-Stokes equations coupled with an energy equation in the Boussinesq approximation,
- discretization: finite element method
- non linear resolution:
  - iterative method: Newton algorithm
  - continuation method on physical parameters
  - pseudo-transient continuation

In this framework, we developed an efficient tool for the resolution of aerothermal problems. We also proposed techniques to reach high Reynolds numerical solutions. And eventually, the numerical results confirmed the adequacy of our implementation.

However, in this chapter, we were only working on small academic problems, and we were free to refine the meshes when necessary. This refining solution is hardly practicable for industrial problems, and we have to consider new approaches. To avoid the Direct Numerical Simulation (DNS), we will introduce turbulence models in the chapter 4. But in a first instance, we have to deal with the instability issue of the Galerkin method. This delicate point is detailed in the next chapter.

CHAPTER 3

Stabilization Methods

Contents

The instability of the Galerkin method is a well-known issue of advection/convection dominated problems. When we started working on aerothermal simulations, the physical parameters were very smooth and not convection dominated. In these conditions, the stability of the solution was not an issue. Our straightforward recipe

was to refine the mesh when oscillations appeared. However, when we started looking at industrial configurations - with realistic physical parameters - it was not possible to refine meshes that much. We had to introduce new ingredients.

We naturally studied turbulence models and started working on the implementation. Nevertheless, it appeared very soon that the turbulence model would not be enough. We needed a specific feature to handle the numerical instability. We considered two kinds of method: the Continuous Interior Penalty (CIP) [11], and the Streamline Diffusion Methods (SDM) [46]. After some tests with the CIP method, we were not fully satisfied: the procedure was either not stable enough or too diffusive. We finally chose to focus our efforts on the SDM which are, by the way, the most common techniques to deal with turbulent flows.

The SDM were first introduced by Hughes and Brooks [46, 7], with the Streamline Upwind/Petrov-Galerkin (SUPG) method for advection dominated flows. Then, this technique has been extended into two new stable Petrov-Galerkin formulations: the Galerkin Least Square (GLS), Hughes *et al.* [43] and the Douglas-Wang (or SGS) for Stokes equations, Douglas and Wang [29]. In the following section, we propose a review of these different approaches for the advection-diffusion-reaction equation and the Navier-Stokes system.

Our implementation of those methods has been enriched progressively to face different issues:

- First SUPG implementation: efficient on homogeneous isotropic meshes. The design of the stabilization parameter was very basic and did not take into account the polynomial order of the finite element space.
- We implemented a new design for the stabilization parameter, and we introduced the weighting coefficient λ_K. The stabilization on anisotropic meshes (boundary layers) with very stretched cells remains inefficient.
- We expected some improvement with the implementation of the GLS and SGS formulations. The GLS helps in some configuration, but the SGS does not provide any significant upgrade. For that reason, we only propose numerical results for GLS and SUPG in this chapter.
- A significant enhancement came with the implementation of a new approach for the evaluation of the characteristic length. The new design is adapted to triangular cells and takes into account the direction of the advection field. Thus, the stabilization parameter became more efficient in the boundary layers, and we finally obtained stable solutions. However, the SDM still produces some under/overshoots in regions of sharp discontinuities. These small oscillations usually generate new kind of instability within the turbulence transport equations.
- The under/overshoots issue have been covered with the implementation of shock capturing (or discontinuity capturing) methods. With this new feature,

we can finally envisage the modeling of turbulent flows.

We give the details of those developments in the further sections. In the last section, we also propose a comparison of the different configurations with some relevant test cases. Hopefully, at the end of the chapter we will be able to determine the optimal stabilization technique for each kind of problems.

1 Streamline Diffusion Methods

Let X an Hilbert space on Ω, with scalar product $(\cdot,\cdot)_X$ and associated norm $\|\cdot\|_X$. We introduce a stationary advection-diffusion problem, where $u \in X$ is the solution of

$$\boldsymbol{a} \cdot \nabla u - \nabla \cdot (\kappa \nabla u) + \sigma u = f, \tag{3.1}$$

where $\boldsymbol{a} \in [\mathcal{L}^2(\Omega)]^d$ is the advection flow satisfying $\nabla \cdot \boldsymbol{a} = 0$ on Ω, $\kappa \in H^1(\Omega)$, $\kappa(\boldsymbol{x}) > 0$ is the diffusivity coefficient, $\sigma > 0 \in \mathcal{L}^2(\Omega)$ is a reaction coefficient and $f \in \mathcal{L}^2(\Omega)$ is source function.

To solve this problem, we introduce a suitable FE space $X_h \subset X$ and we write the variational formulation of equation (3.1), as find $u_h \in X_h$ such that

$$\int_\Omega \boldsymbol{a} \cdot \nabla u_h v_h \, d\Omega \; + \int_\Omega \kappa \nabla u_h \cdot \nabla v_h \, d\Omega \; + \int_\Omega \sigma u_h v_h \, d\Omega \; = \int_\Omega f v_h \, d\Omega \, , \quad \forall v_h \in X_h. \tag{3.2}$$

This discrete problem is called advection (or convection) dominated when the Peclet number is larger than 1,

$$\mathrm{Pe} = \frac{\|\boldsymbol{a}\|_\infty L}{2\kappa} > 1, \tag{3.3}$$

with L the characteristic length of the domain. This definition might be inappropriate when the nature of the flow $\boldsymbol{a}$ brutally changes in the domain or when the mesh is not homogeneous. For that reason we usually prefer the local Peclet number

$$\mathrm{Pe}_h = \frac{|\boldsymbol{a}|_p h}{2\kappa}, \tag{3.4}$$

where $|\cdot|_p$ denotes the p-norm and h is the characteristic length of the current element. Using the local Peclet number we can define a problem as advection dominated as soon as its max on the domain is larger than 1,

$$\max_{\boldsymbol{x} \in \Omega} \mathrm{Pe}_h(\boldsymbol{x}) > 1. \tag{3.5}$$

In the context of advection-dominated flow, the error bound on the Galerkin projection u_h does not guarantee a good approximation anymore. In practice the system becomes numerically unstable and non-physical oscillations appear. A trivial solution would be to refine the mesh to reduce the Peclet number, but this approach might be very costly and not achievable in practice. The alternative solution is to add a stabilization operator to take care of these numerical oscillations.

1.1 Stabilized Formulation

We first introduce the stabilized formulation, using the decomposition of the operators, as proposed in [43]. We denote by $\mathscr{L}$ the advection diffusion reaction operator, defined as

$$\mathscr{L}u = \boldsymbol{a} \cdot \nabla u - \nabla \cdot (\kappa \nabla u) + \sigma u, \forall u \in X \tag{3.6}$$

and we also define its symmetric and skew-symmetric contributions

$$\mathscr{L}_S u = -\nabla \cdot (\kappa \nabla u) + \sigma u, \qquad \mathscr{L}_{SS} = \boldsymbol{a} \cdot \nabla u, \quad \forall u \in X \tag{3.7}$$

respectively.

The stabilized formulation of equation (3.2) can be written as, find $u_h \in X_h$ such that

$$a(u_h, v_h) + a_K(u_h, v_h) = l(v_h) + l_K(v_h), \quad \forall v_h \in X_h, \tag{3.8}$$

where for any $(u, v) \in X^2$

$$a(u,v) = \int_\Omega \boldsymbol{a} \cdot \nabla u v \; d\Omega \; + \int_\Omega \kappa \nabla u \cdot \nabla v \; d\Omega \; + \int_\Omega \sigma u v \; d\Omega \; , \qquad l(v) = \int_\Omega f v \; d\Omega \; , \tag{3.9}$$

and the stabilization terms a_K and l_K are defined, for any $(u_h, v_h) \in X_h$, such that

$$a_K(u_h, v_h) = \sum_{K \in \mathcal{T}_h} \int_K \tau_h \mathscr{L} u_h (\mathscr{L}_{SS} v_h + \gamma \mathscr{L}_S v_h) \; d\Omega \; , \tag{3.10}$$

$$l_K(v_h) = \sum_{K \in \mathcal{T}_h} \int_K \tau_h f (\mathscr{L}_{SS} v_h + \gamma \mathscr{L}_S v_h) \; d\Omega \; . \tag{3.11}$$

The computation of the stabilization parameter τ_h is detailed in the following section. The constant γ determines the nature of the stabilization method:

- $\gamma = 0$: SUPG,
- $\gamma = 1$: GLS,
- $\gamma = -1$: DW.

Remark 3. *The stabilization terms* (3.10) *are consistent with respect to the strong formulation of the problems. They vanish if we plug the exact solution in the equations.*

1.2 The Navier-Stokes Equations

The stationary Navier-Stokes system

$$\begin{cases} \rho \boldsymbol{u} \nabla \boldsymbol{u} - \nabla \cdot (\mu \nabla \boldsymbol{u}) + \nabla p = g \\ \nabla \cdot \boldsymbol{u} = 0 \end{cases} \tag{3.12}$$

can be seen as a particular case of the generic advection diffusion reaction problem presented in former section by identifying $f = g - \nabla p$, $\boldsymbol{a} = \rho \boldsymbol{u}$, $\kappa = \mu$ and $\sigma = 0$. We can then extend the stabilization operator described previously and write the stabilized variational formulation for NS system, as find $(\boldsymbol{u}_h, p_h) \in V_h \times Q_h$ such that

$$\begin{aligned} \int_\Omega \rho \boldsymbol{u}_h \nabla \boldsymbol{u}_h \cdot \boldsymbol{v}_h \, d\Omega \; + \int_\Omega \mu \nabla \boldsymbol{u}_h : \nabla \boldsymbol{v}_h \, d\Omega \\ - \int_\Omega p_h \nabla \cdot \boldsymbol{v}_h \, d\Omega \; + \int_\Omega (\nabla \cdot \boldsymbol{u}_h) q_h \, d\Omega \; + a_K^{NS}(\boldsymbol{u}_h, \boldsymbol{v}_h; p_h) = \int_\Omega g \boldsymbol{v}_h \, d\Omega \; + l_K^{NS}(\boldsymbol{u}_h, \boldsymbol{v}_h) \end{aligned} \tag{3.13}$$

where

$$a_K^{NS}(\boldsymbol{u}_h, \boldsymbol{v}_h; p_h) = \sum_{K \in \mathcal{T}_h} \int_K \tau_h (\rho \boldsymbol{u}_h \nabla \boldsymbol{u}_h - \nabla \cdot (\mu \nabla \boldsymbol{u}_h) + \nabla p_h) \cdot (\rho \boldsymbol{u}_h \nabla \boldsymbol{v}_h - \gamma \nabla \cdot (\mu \nabla \boldsymbol{v}_h)) \, d\Omega \tag{3.14}$$

$$+ \sum_{K \in \mathcal{T}_h} \int_K \delta_h \nabla \cdot \boldsymbol{u}_h \nabla \cdot \boldsymbol{v}_h \, d\Omega \tag{3.15}$$

$$l_K^{NS}(\boldsymbol{u}_h, \boldsymbol{v}_h) = \sum_{K \in \mathcal{T}_h} \int_K \tau_h g \cdot (\rho \boldsymbol{u}_h \nabla \boldsymbol{v}_h - \gamma \nabla \cdot (\mu \nabla \boldsymbol{v}_h)) \, d\Omega \tag{3.16}$$

with γ and τ_h defined in previous paragraph. We choose here to keep the term l_K^{NS} on the right-hand side since the value of the convection field in that term is usually taken at the previous linear iteration, see section 3 for more details on the non-linear resolution.

Remark 4. *We add a stabilization term on the divergence. This term is controlled by a new parameter $\delta_h = 2|\boldsymbol{u}_h|_p^2 \tau_h$, where $|\cdot|_p$ is the p-norm as used in the definition of τ_h, see next section. The divergence stabilization term is not mandatory, but we observed a clear improvement of the results when using it. In further sections, we suppose that this term is present in the stabilized formulation by default.*

1.3 Convergence of the Method

We will not detail the error analysis of the SDM in this manuscript. The convergence of this formulation has been studied, for instance, in [43, 44, 53]. We remind here the global result: for sufficiently smooth u, there exists a constant C_u such that

$$\|u_h - u\|^2 \leq C_u h^{2l} \tag{3.17}$$

where

$$\begin{aligned} 2l &= 2k + 1, \quad \alpha \text{ large} \\ 2l &= 2k, \qquad \alpha \text{ small} \end{aligned} \tag{3.18}$$

with k the polynomial order of the finite element space and α is the weight local Peclet number introduced in (3.26), see section 2.

We reporter the reader to the cited references for a rigorous formulation of the hypothesis and a proof of this result.

2 Stabilization Parameter

The present stabilization methods are particularly sensitive to the choice of the parameter τ_h. Many different definitions of this parameter can be found in the literature. We present in this section a non-exhaustive list of suitable parameters determined from different mathematical error analysis [45, 43, 61, 62]. We first detail the design of the parameter for advection-diffusion problem and then an extension of this definition for problems with reaction term. In the last subsection, we present different approaches for the delicate computation of the characteristic length of an element.

The required ingredient for the stabilization parameter is the evaluation of the local Peclet number, Pe_h, presented in (3.4). For the Navier Stokes equation, the Peclet number is replaced by the Reynolds number but definition and roles are identical,

$$\mathrm{Re}_h = \frac{|\boldsymbol{u}|_p h}{2\nu}. \tag{3.19}$$

These quantities mainly depend on the characteristic size h and have real impacts on the efficiency of the method. For now, we do not specify the form of the length h. We will focus on this delicate detail in the subsection 2.2.

The value of the local Peclet/Reynolds number will determine the type of problem we are dealing with. Small Peclet/Reynolds numbers correspond to diffusive problems and are usually stable. Large Peclet/Reynolds numbers define convection dominated cases and require adapted stabilization. For the rest of the section, we use Peclet number and associated notations ($\boldsymbol{a}$, κ) in the formulas. The computation of stabilization parameter is equivalent for Navier Stokes equations, by replacing the corresponding quantities $\mathrm{Pe}_h \sim \mathrm{Re}_h$, $\boldsymbol{a} \sim \boldsymbol{u}$ and $\kappa \sim \nu$.

The stabilization parameter τ_h has to satisfy the asymptotic behaviors

$$\tau_h = O\left(\frac{h}{|\boldsymbol{a}|}\right), \quad \mathrm{Pe}_h \text{ large}, \qquad \tau_h = O\left(\frac{h^2}{\kappa}\right), \quad \mathrm{Pe}_h \text{ small} \tag{3.20}$$

In [8], Hughes *et al.* determine the optimal stabilization parameter, for 1-D problem with linear FE, as

$$\tau_h^0 = (\coth(\mathrm{Pe}_h) - \frac{1}{\mathrm{Pe}_h}). \tag{3.21}$$

This formula has been generalized for multi-dimensional problems but requires some modifications for higher FE polynomial order. An extension of this formula

was proposed in [45, 43], with the introduction of a function $\xi(\alpha)$ to locally switch between diffusive and convective problems. We can then define the stabilization parameter as

$$\tau_h^1 = \frac{h}{2|\boldsymbol{a}|_2}\xi(\alpha). \tag{3.22}$$

where α is a weighted version of Pe_h and $\xi(\alpha)$ has the properties described in figure 3.1, with $m \leq 2C$ and C is a constant satisfying the inverse estimate inequality

$$C \sum_{K \in \mathcal{T}_h} h^2 \|\Delta v\|_K^2 \leq \|\nabla v\|^2, \quad v \in X_h \tag{3.23}$$

where $\|\cdot\|_K$ is the $\mathcal{L}^2$-norm in the cell K. The practical evaluation of the constant

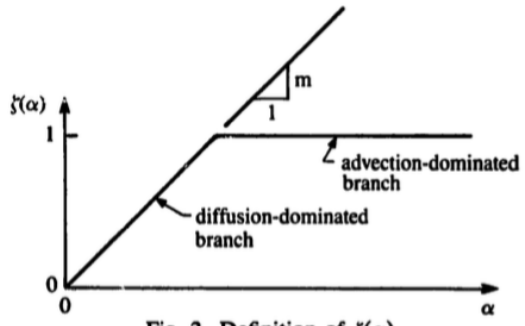


Fig. 2. Definition of $\zeta(\alpha)$.

Figure 3.1 – Definition of $\xi(\alpha)$, from [43]

C is explained in [48] : some basic values of C are given for particular shapes and polynomial degrees, for instance $C = 1/24$ for biquadratic rectangular elements. A general method is also proposed in order to evaluate C in any cases using the solution of the local eigen value problem

$$\lambda_K = \max_{v \in (\mathbb{P}^k(K) \setminus \mathbb{R})} \frac{\|\Delta \boldsymbol{v}\|_K^2}{\|\nabla \boldsymbol{v}\|_K^2} \tag{3.24}$$

we can then define

$$C = \frac{1}{\lambda_K h^2}. \tag{3.25}$$

With this definition, C is a constant independent of the shape of the cell as long as we use the definition of h given in [48], this definition is detailed in the subsection 2.2. We actually figured out that C has to be a constant on the whole mesh. This property is particularly important on anisotropic meshes. To recover this constant value we always use the h_h definition (3.45) to compute C. Then, we can use any definition for the characteristic length to compute the local Peclet number.

Hughes *et al.*proposed different formulas for $\xi(\alpha)$ first in [45] and [43] and finally in [61, 62]. This last version is more convenient to use and became the most common formulation, we recall here the main ingredients:

$$\xi(\alpha) = \begin{cases} \alpha, & 0 \leq \alpha \leq 1, \\ 1, & \alpha \geq 1, \end{cases} \tag{3.26}$$

$$\alpha = \frac{m|a|_p h}{2\kappa}, \tag{3.27}$$

$$|a|_p = \begin{cases} \left(\sum_{i=1}^{d} |a_i|^p\right)^{1/p}, & 1 \leq p < \infty, \\ \max\limits_{i=1,..,d} |a_i|, & p = \infty, \end{cases} \tag{3.28}$$

$$m = \min\{\frac{1}{3}, 2C\}. \tag{3.29}$$

with C defined as in (3.23) and then τ is defined as in (3.22).

Remark 5. *The interest of the weighted Peclet/Reynolds number α is to take into account the effect of the polynomial FE degree. Indeed, polynomial degree $k \geq 2$ has non-negligible effects on local Peclet/Reynolds number since it highly improves the quality of the discrete approximation.*

We will compare this definition of τ_h^1 with a formula proposed by Burda *et al.*in [10] using the approximation $h \approx \frac{1}{\sqrt{\lambda_K}}$ which also takes into account the influence of the polynomial order,

$$\tau_h^2 = \frac{\xi(\alpha)}{\sqrt{\lambda_K}|a|_p}, \tag{3.30}$$

$$\xi(\alpha) = \begin{cases} \alpha, & 0 \leq \alpha \leq 1, \\ 1, & \alpha \geq 1, \end{cases} \tag{3.31}$$

$$\alpha = \frac{|a|_p}{2\kappa\sqrt{\lambda_K}}, \tag{3.32}$$

$$\tag{3.33}$$

where λ_K is defined as in (3.24) and $|\cdot|_p$ is the usual p-norm defined in (3.26).

2.1 Extension to Advection Diffusion Reaction Problems

The parameter τ_h^1 defined in (3.22) is very efficient for advection dominated flows, but he does not take care of the potentially high reaction coefficient σ. It is not an an issue for the resolution of the aerothermal model presented in chapter I but we met some difficulties with the stabilization of the turbulence models equations, see chapter 4.

An extension of τ_h^1 for problem with reaction term is proposed in [33], with a new stabilization parameter τ_h^3 defined by

$$\tau_h^3 = \frac{h^2}{\sigma h^2 \xi(\alpha_\sigma) + \frac{4\kappa}{m}\xi(\alpha_\kappa)}, \tag{3.34}$$

$$\alpha_\sigma = \frac{2\kappa}{m\sigma h^2}, \tag{3.35}$$

$$\alpha_\kappa = \frac{m|\boldsymbol{a}|_p h}{2\kappa}, \tag{3.36}$$

$$\xi(\alpha) = \begin{cases} 1, & 0 \le \alpha \le 1, \\ \alpha, & \alpha \ge 1, \end{cases} \tag{3.37}$$

$$|\boldsymbol{a}|_p = \left(\sum_{i=1}^{d} |\boldsymbol{a}|^p \right)^{1/p}, \quad 1 \le p < \infty. \tag{3.38}$$

We did minor changes in the constants of τ_h^3 and α_κ in order to recover exactly the definition of τ_h^1 when $\sigma = 0$.

2.2 Choice of the characteristic length

The definition of the characteristic size of a cell is a key ingredient in the computation of a stabilization parameter. This choice is particularly crucial for anisotropic meshes with important aspect ratios. For isotropic meshes, most of the following solutions will produce acceptable results.

We want to compare these different options for highly anisotropic meshes, as used in the turbulent boundary layers. The same kind of study has been made by Mittal, [73] for the stabilization of equal-order-interpolation velocity-pressure for Navier-Stokes equations.

h_m **definition :** $h = h_m$ is the length of the smallest edge of the cell.

h_d **definition :** We use the projection of the current convection field $\boldsymbol{a}$ on the directions of the cell. This definition was introduced by Hughes in [42] and is particularly convenient for anisotropic meshes. We first define the local convection field $\hat{\boldsymbol{a}}$ on the reference element, $\hat{K}$:

$$\hat{\boldsymbol{a}} = \boldsymbol{J}^{-1}\boldsymbol{a} \tag{3.39}$$

where $\boldsymbol{J}$ is the Jacobian of the geometric mapping from the reference element to the current cell. We can then define

$$h = h_d = \frac{|\boldsymbol{a}|_p}{|\hat{\boldsymbol{a}}|_p}\hat{h}, \tag{3.40}$$

where $\hat{h}$ is the characteristic length of the reference element. This definition works particularly well on hypercubes, but we noticed some regularity issue when we applied this method on triangles. We finally figured out that these irregularities came from the shape of the reference cell. In Feel++, the reference simplex (in 2D) is a squared isosceles triangle. This shape is not symmetric by rotation and then the advection field $\hat{\boldsymbol{a}}$ would depend on the rotation between the current cell and the reference element. To overcome this problem, we decided to refer $\hat{\boldsymbol{a}}$ to a new reference triangle $\tilde{K}$ which is equilateral. An illustration of this procedure

Figure 3.2 – Geometric transformation back on a reference element $\tilde{K}$. Computation of the local field $\tilde{\boldsymbol{a}}$, used for the characteristic size h_d.

is proposed in figure 3.2. In our case the geometric mapping from $\hat{K}$ to $\tilde{K}$ can be written as

$$\tilde{X} = \boldsymbol{M}\hat{X} \tag{3.41}$$

with

$$\boldsymbol{M} = \begin{bmatrix} 1 & 1/2 \\ 0 & \sqrt{3}/2 \end{bmatrix}. \tag{3.42}$$

Then we can use the matrix $\boldsymbol{M}$ to report the field $\hat{\boldsymbol{a}}$ on $\tilde{K}$.

$$\tilde{\boldsymbol{a}} = \boldsymbol{M}\hat{\boldsymbol{a}} = \boldsymbol{M}\boldsymbol{J}^{-1}\boldsymbol{a} \tag{3.43}$$

and finally the characteristic length is computed as

$$h = h_d = \frac{|\boldsymbol{a}|_p}{|\tilde{\boldsymbol{a}}|_p}\tilde{h}. \tag{3.44}$$

h_h **definition :** Harari and Hughes proposed this definition in [48] and provided an interesting measure for anisotropic meshes. The idea is to compute the size of the cell using the area of the element and the average vertex-to-centroid distance. The formulation for triangle is

$$h_h = \frac{4S}{\sqrt{3\sum_{a=1}^{3}|x_i - x_c|^2}} \tag{3.45}$$

where S is the surface of the cell, x_i are the vertex of the triangle and x_c the centroid of the triangle. We have a similar definition for quadrangle elements

$$h_h = h_x h_y \sqrt{\frac{2}{h_x^2 + h_y^2}} \tag{3.46}$$

where h_x and h_y are the characteristic length of the quadrangle in both direction.

h_e **definition :** an almost optimal definition for anisotropic elements would be to compute the inscribed ellipse for each element and then to take as $h = h_e$ the radius of this ellipse in the direction of the advection field $\boldsymbol{a}$. Even if this approach is interesting in theory, it appears impracticable to implement and we did not take time to study this version.

3 Shock Capturing Method

The SDM are particularly efficient to stabilize convection-dominated problems, however these methods also generate significant overshoots and undershoots in regions of high gradient. An example of this non-physical oscillations can be seen in figure 3.3

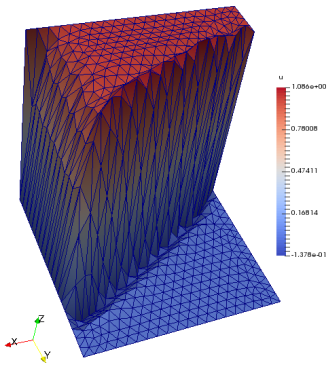


Figure 3.3 – Non-physical oscillations resulting from supg stabilization on the advection skew mesh test case 4.2

Different methods have been developed in the last decades to eliminate or at least diminish these oscillations. We particularly noted three families of methods. The non-linear upwinding techniques, introduced by Mizukami and Hughes [74], is very effective to remove the oscillations but is known to introduce too much numerical diffusion and then a loss of accuracy. Another solution is to add artificial diffusion. This diffusion can be either isotropic [45], orthogonal to the streamlines [52] or based on an edge stabilization [13]. In their implementation of a $k-\varepsilon$ model, Kuzmin et al. [59] proposed an alternative algebraic solution, by applying an iterative flux correction. Note that all these methods are non-linear.

In this work, we chose to focus on the artificial diffusion methods. These methods are convenient to implement once the SUPG/GLS framework is in place. Also, they succeed in being very effective with no excessive artificial diffusivity. We implemented different versions proposed in the literature, and we give some details in this section.

3.1 Isotropic Artificial Diffusion

Hughes *et al.*[45] first proposed to add a new term in the SUPG stabilization

$$(R_h(\boldsymbol{u}_h), \tau_h^{sc}\boldsymbol{a}^{\|} \cdot \nabla \boldsymbol{v}_h) \tag{3.47}$$

where $R_h(\boldsymbol{u}_h)$ is the discrete residual, $\boldsymbol{a}^{||}$ is the projection of the convection field $\boldsymbol{a}$ in the direction of $\nabla \boldsymbol{u}_h$ and τ_h^{sc} is a non negative stabilization parameter. Many definitions and improvements were developed after this first version, we remind hereafter the most common versions.

Galeão and do Carmo [28] proposed to replace the projected convection field $\boldsymbol{a}^{||}$ by the new vector field $z_h = R_h(\boldsymbol{u}_h)\frac{\nabla \boldsymbol{u}_h}{\|\nabla \boldsymbol{u}_h\|_2^2}$. The new additional term then reads

$$(R_h(\boldsymbol{u}_h), \tau_h^{sc} \boldsymbol{z}_h \cdot \nabla \boldsymbol{v}_h), \tag{3.48}$$

with the following definition for τ_h^{sc}

$$\tau_h^{sc} = \tau_h \max\left(0, \frac{\|\boldsymbol{a}\|_2}{\|\boldsymbol{z}_h\|_2} - 1\right), \tag{3.49}$$

where τ_h is the SUPG/GLS stabilization parameter as defined in section 2.

Almeida *et al.*[1] also use the formulation (3.48) but they introduce a new quantity ζ_h in their definition of τ_h^{sc} to reduce the artificial diffusion in regions where the numerical solution is smooth enough.

$$\tau_h^{sc} = \tau_h(\boldsymbol{a}) \max\left(0, \frac{\|\boldsymbol{a}\|_2}{\|\boldsymbol{z}_h\|_2} - \zeta_h\right), \quad \text{with} \quad \zeta_h = \max\left(1, \frac{\boldsymbol{a} \cdot \nabla \boldsymbol{u}_h}{R_h(\boldsymbol{u}_h)}\right). \tag{3.50}$$

3.2 Artificial Diffusion Added Orthogonally to Streamlines

This method, introduced by Johnson *et al.*[52] is very similar to the isotropic diffusion, but it is only applied in the direction orthogonal to Streamlines adding the term

$$(\tau_h^{sc} \bar{D} \nabla \boldsymbol{u}_h, \nabla \boldsymbol{v}_h) \tag{3.51}$$

where $\bar{D} = \mathbb{I} - \boldsymbol{a} \otimes \frac{\boldsymbol{a}}{\|\boldsymbol{a}\|_2^2}$ is the projection orthogonally to vector $\boldsymbol{a}$ and τ_h^{sc} is defined as $\tau_h^{sc} = \max\left(0, \|\boldsymbol{a}\|_2 h^{3/2} - \kappa\right)$.

Codina [18] gives a new definition for τ_h^{sc}, based on the validation of the discrete maximum principle for several problems,

$$\tau_h^{sc} = \frac{1}{2} \max\left(0, C - \frac{2\kappa}{\|\boldsymbol{a}^{\|}\|_2 h}\right) h \frac{R_h(\boldsymbol{u}_h)}{\nabla u_h}, \tag{3.52}$$

where C is a suitable positive constant (proposed constant are $C = 0.6$ for linear elements and $C = 0.35$ for quadratic elements).

Knobloch *et al.*proposed in [51] to replace $|\boldsymbol{a}^{||}|$ by the quantity $Q_K(\boldsymbol{u}_h) = \frac{\|R_h(\boldsymbol{u}_h)\|_2}{\|\nabla \boldsymbol{u}_h\|_2}$ which leads to the formulation

$$\tau_h^{sc} = \frac{1}{2} \max\left(0, C - \frac{2\kappa}{Q_K(\boldsymbol{u}_h) h}\right) h Q_K(\boldsymbol{u}_h). \tag{3.53}$$

In the same reference, Knobloch *et al.*also proposed a modified version of the shock capturing terms presented by Burman and Ern in [12] with

$$\tau_h^{sc} = \tau_h(\boldsymbol{a}) \frac{\|\boldsymbol{a}\|_2^2 \|R_h(\boldsymbol{u}_h)\|_2}{\|\boldsymbol{a}\|_2 \|\nabla \boldsymbol{u}_h\|_2 + \|R_h(\boldsymbol{u}_h)\|_2} \tag{3.54}$$

Considering all these methods and the results of section 4, we propose to use the shock capturing term (3.51) but with the new parameter

$$\tau_h^{sc} = \tau_h(\boldsymbol{a}) \max\left(0, \|\boldsymbol{a}\|_2^2 Q_K(\boldsymbol{u}_h) - Q_K^2(\boldsymbol{u}_h)\right). \tag{3.55}$$

We had no time to perform a proper convergence study for this parameter, but the numerical results presented in further sections are clearly encouraging.

4 Validation

We have implemented and compared the previously introduced methods on different well-known test cases. We proposed in this section a validation and a comparison of the different configurations. We tested the different stabilization methods : SUPG and GLS, with parameters 1 (3.26), 2 (3.30) and possibly 3 (3.34). We use the subscript $_m$, $_d$ or $_h$ for the different h formulations proposed in section 2.2. We compare those methods with standard Galerkin method (SGM) when it makes sense. All the simulations are made on triangular meshes with order 2 FE.

4.1 Comparison of the Stabilization Methods

Advection Skew the Mesh

We propose here a qualitative observation of the results computed with different configurations, on the well known advection skew the mesh example, [8]. The advection field is constant $\boldsymbol{a} = (\cos(\alpha), \sin(\alpha))$, the diffusivity coefficient is $\kappa = 10^{-8}$. There is no reaction and no source term. The test case consists in the propagation of a discontinuity in quasi pure advection setup. The domain is a square $\Omega = [0,1] \times [0,1]$ with the boundary conditions: $u(x,y) = 1$ on Γ_1 and $u(x,y) = 0$ on Γ_0, with $\Gamma_1 = \{(x,y), x = 0, 0 \leq y \leq 0.2\} \cup \{(x,y), y = 0\}$ and $\Gamma_0 = \partial\Omega \backslash \Gamma_1$. This setup is recapped on figure 3.4.

We compare the two methods (SUPG and GLS) for different values of α, $\alpha = \text{atan}(0.5)$ on figure 3.5, $\alpha = \text{atan}(1)$ on figure 3.6 and $\alpha = \text{atan}(2)$ on figure 3.7. Those quantitative results allow some observations:

- All the configurations succeed in stabilizing the model where the SGM does not converge. We note the apparition of non-physical undershoot and overshoot near the discontinuities.

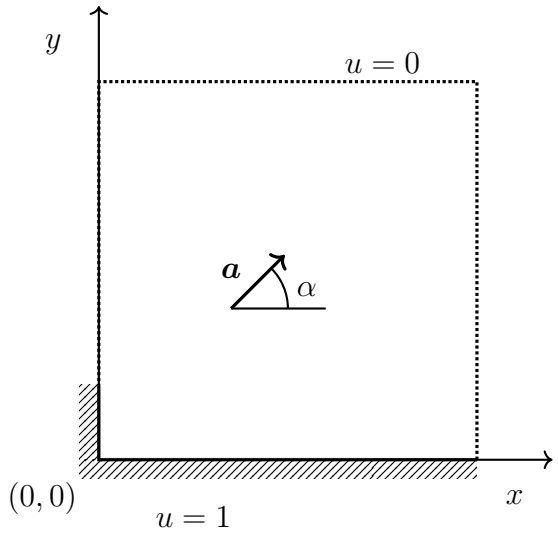


Figure 3.4 – Advection Skew the Mesh Setup

- The two methods (SUPG and GLS) give almost identical results and the stabilization parameter seems to have more influence than the method.
- The parameters τ_h^2 is more sensitive to discontinuities and boundaries.
- For τ_h^1, the choice of the characteristic length has only slight influence.

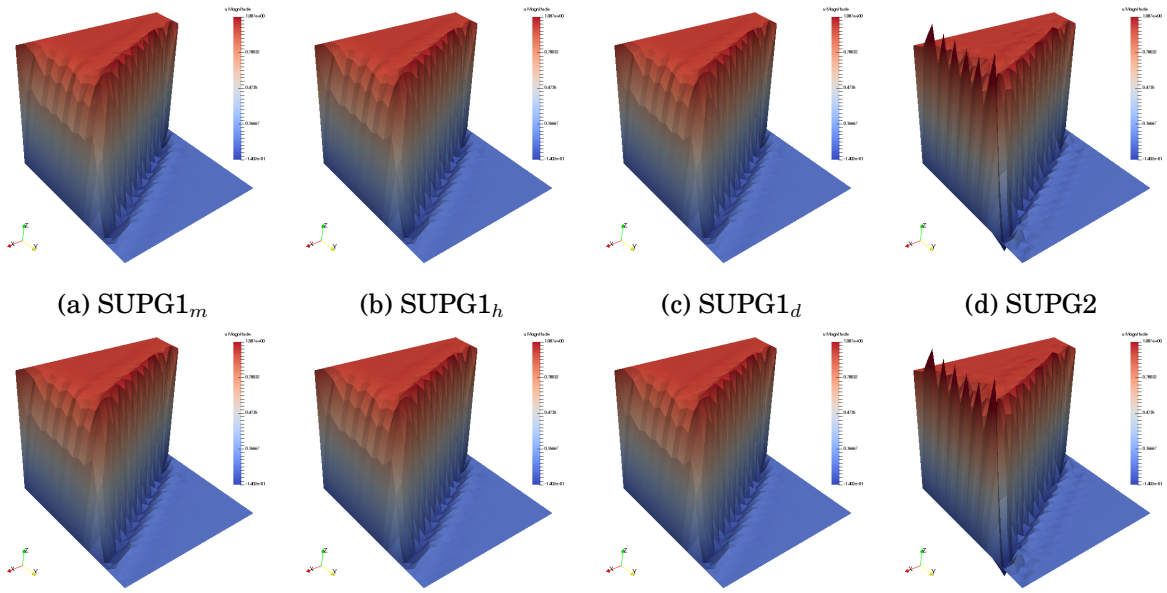
Thermal Boundary Layer

In this test case, we want to compare the different stabilization parameters for advection-diffusion-reaction problems and especially their accuracy on non-homogeneous anisotropic meshes.

We consider a rectangular domain $\Omega = [0, 1] \times [0, 0.5]$, the boundary conditions are :

$$\begin{aligned} u &= 0, \qquad 1 \leq x \leq 1, y = 0, \\ u &= 1, \qquad x = 0, 0 \leq y \leq 0.5, \\ u &= 1, \qquad 0 \leq x \leq 1, y = 0.5 \\ u &= 2y, \qquad x = 1, 0 \leq y \leq 0.5 \end{aligned} \tag{3.56}$$

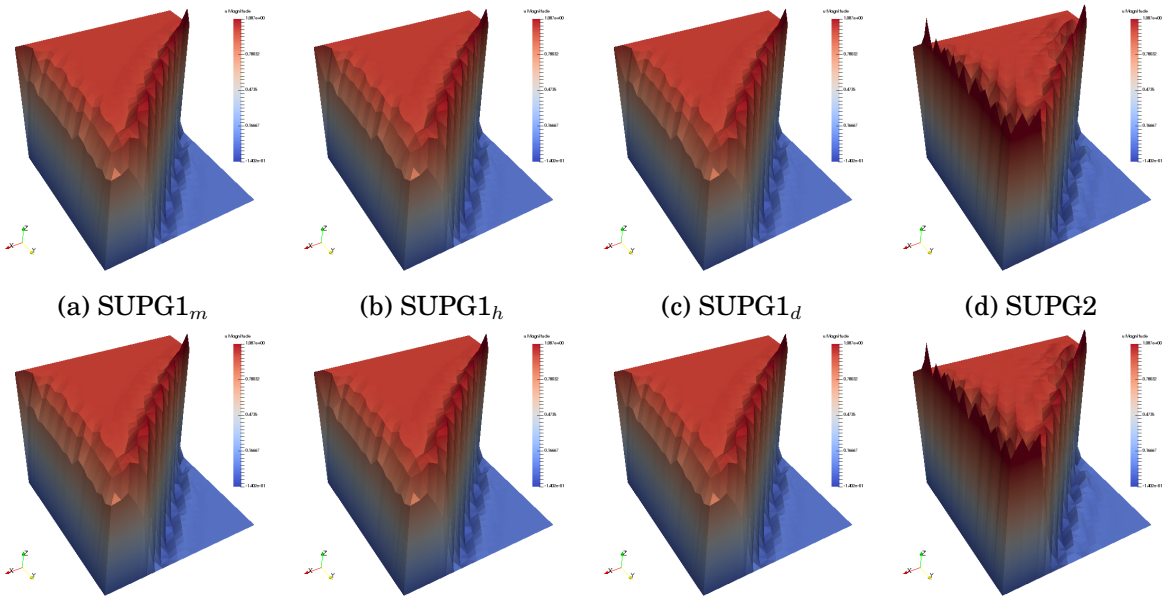
On this geometry, we do consider two configurations, see figure 3.9. The first one with a coarse non-structured mesh, with a diffusion $\kappa = 10^{-5}$. The second case is set on an non-homogeneous anisotropic structured mesh, with $\kappa = 10^{-8}$. This second mesh presents high mesh ratio on the bottom boundary (up to 10^5). The stabilization on this kind of stretched elements is particularly complicated and requires a suitable definition of the characteristic length h. For both configurations, the advection field is defined as $\boldsymbol{a} = (2y, 0)$. The interest of using the advection-diffusion-reaction parameter τ_h^3 will be discussed with different values for the reaction parameter σ.



(a) SUPG1$_m$ (b) SUPG1$_h$ (c) SUPG1$_d$ (d) SUPG2

(e) GLS1$_m$ (f) GLS1$_h$ (g) GLS1$_d$ (h) GLS2

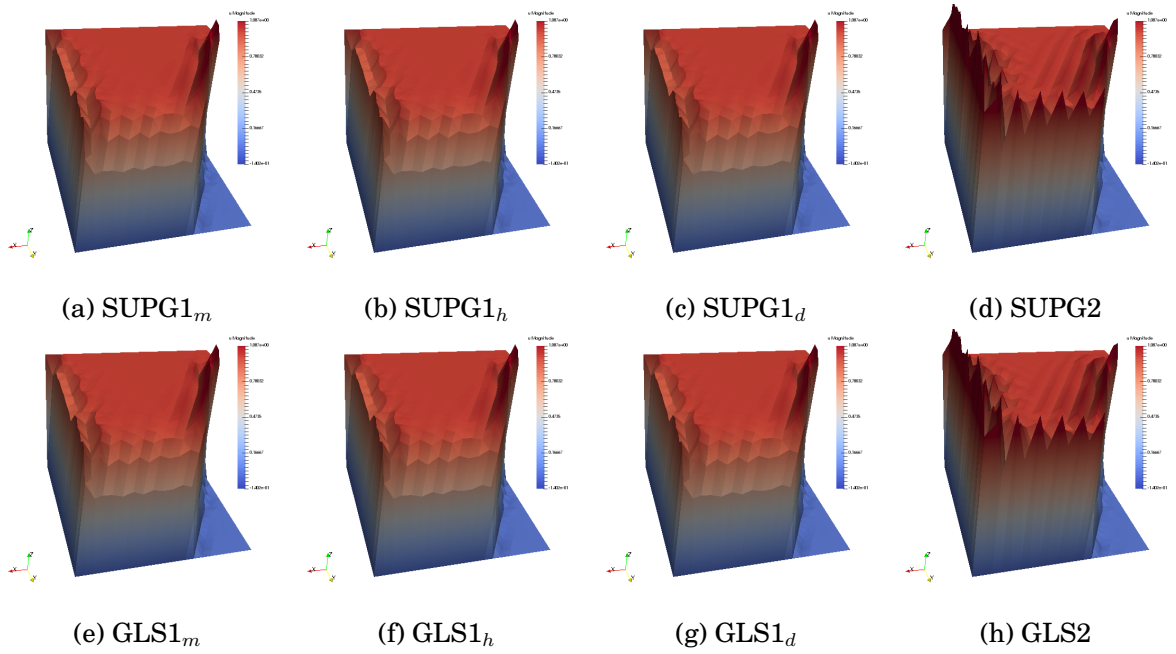
Figure 3.5 – Advection Skew the Mesh profiles for $\alpha = \mathrm{atan}(0.5)$.
Comparison of the two methods with different parameters and characteristic length



(a) SUPG1$_m$ (b) SUPG1$_h$ (c) SUPG1$_d$ (d) SUPG2

(e) GLS1$_m$ (f) GLS1$_h$ (g) GLS1$_d$ (h) GLS2

Figure 3.6 – Advection Skew the Mesh profiles for $\alpha = \mathrm{atan}(1)$.
Comparison of the two methods with different parameters and characteristic lengths



(a) SUPG1$_m$ (b) SUPG1$_h$ (c) SUPG1$_d$ (d) SUPG2

(e) GLS1$_m$ (f) GLS1$_h$ (g) GLS1$_d$ (h) GLS2

Figure 3.7 – Advection Skew the Mesh profiles for $\alpha = \text{atan}(2)$.
Comparison of the two methods with different parameters and characteristic lengths

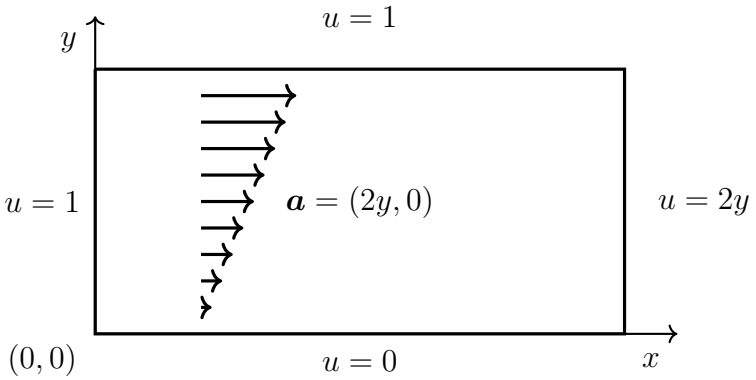
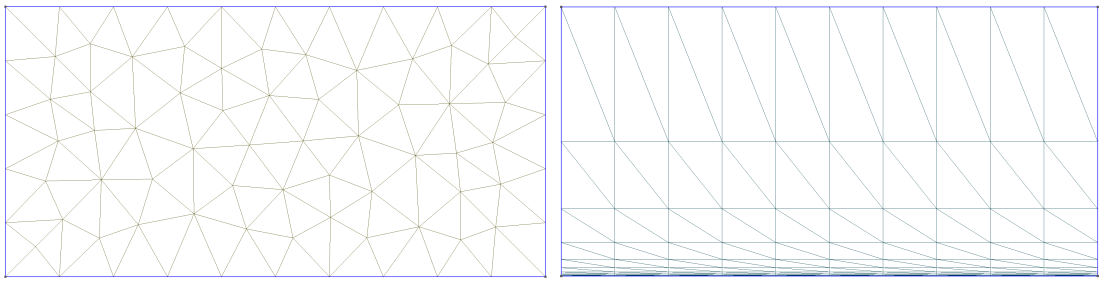


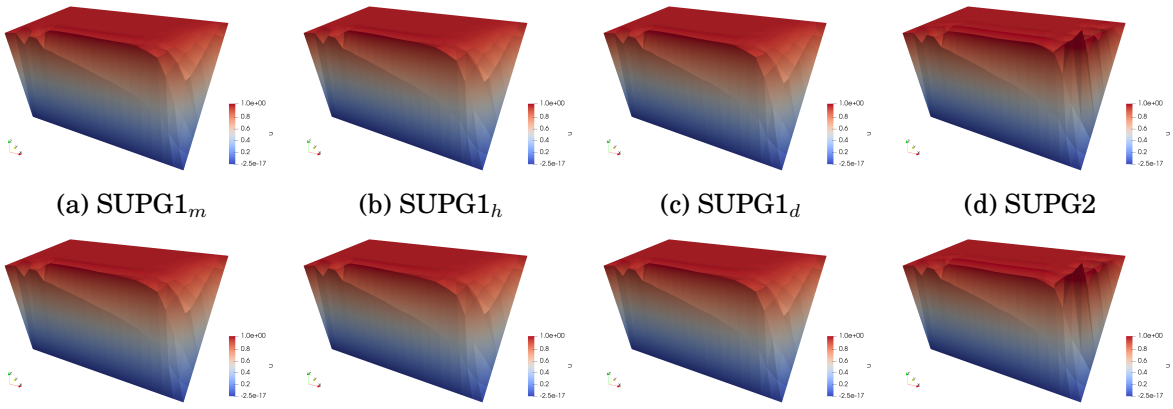
Figure 3.8 – Thermal Boundary Layer Setup

We first compare the results obtained with different methods, parameters, and characteristic lengths. This study is made on the unstructured homogeneous configuration. The setup presents sharp discontinuities and is not trivial to stabilize. The profiles are presented in figure 3.10. We also propose three profile cuts ($x = 0.2$, $x = 0.5$ and $x = 0.8$) to accurately compare the methods, see figure 3.11. We globally retrieve the same conclusions we made in the previous test case: all configurations stabilize, with more oscillations for τ_h^2 parameter. From those first test cases, we can reasonably conclude that the choice of the characteristic length will not influence the stabilization method, at least on isotropic meshes. Also, the underlying method, SUPG or GLS, has very little influence on the solution. For those reasons, we will usually choose SUPG formulation (which is a bit less expensive to assemble) and the τ_h^1 parameter with h_m definition for advection-diffusion on isotropic meshes.



(a) Unstructured Homogeneous Mesh (b) Structured Non-Homogeneous Anisotropic Mesh

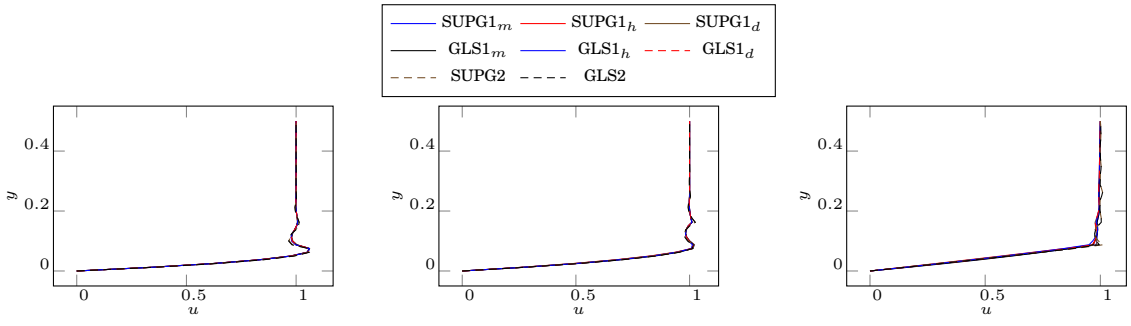
Figure 3.9 – Meshes Used for the two Different Configurations of the Thermal Layer Teste Case



(a) SUPG1$_m$ (b) SUPG1$_h$ (c) SUPG1$_d$ (d) SUPG2

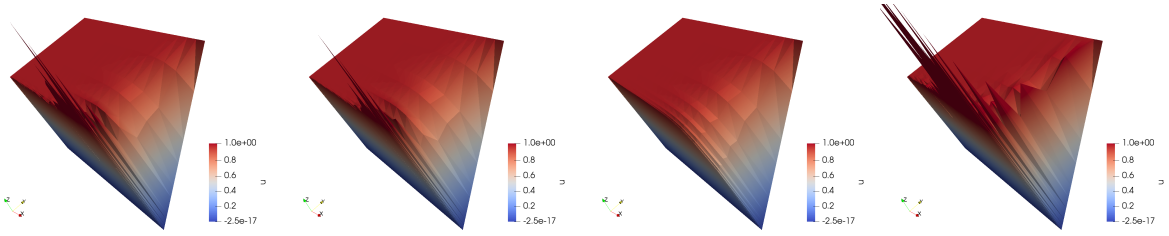
(e) GLS1$_m$ (f) GLS1$_h$ (g) GLS1$_d$ (h) GLS2

Figure 3.10 – Thermal Layer on Coarse Unstructured Homogeneous Mesh. $\sigma = 0$, $\kappa = 10^{-5}$. Comparison of the two methods with different parameters and characteristic length

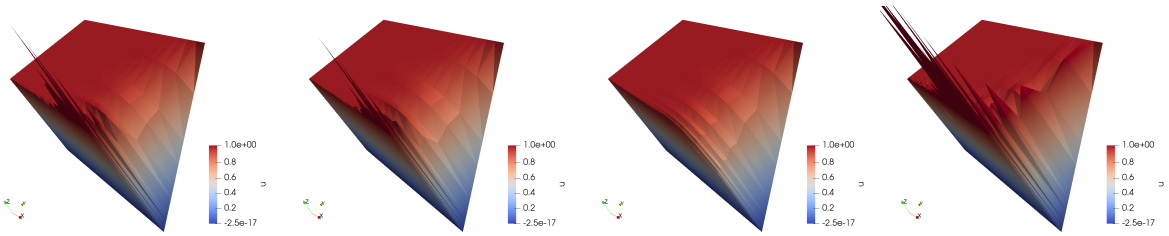


(a) Vertical Cut $x = 0.2$ (b) Vertical Cut $x = 0.5$ (c) Vertical Cut $x = 0.8$

Figure 3.11 – Thermal Layer on Coarse Unstructured Homogeneous Mesh. $\sigma = 0$, $\kappa = 10^{-5}$. Comparison of the two methods with different parameters and characteristic length

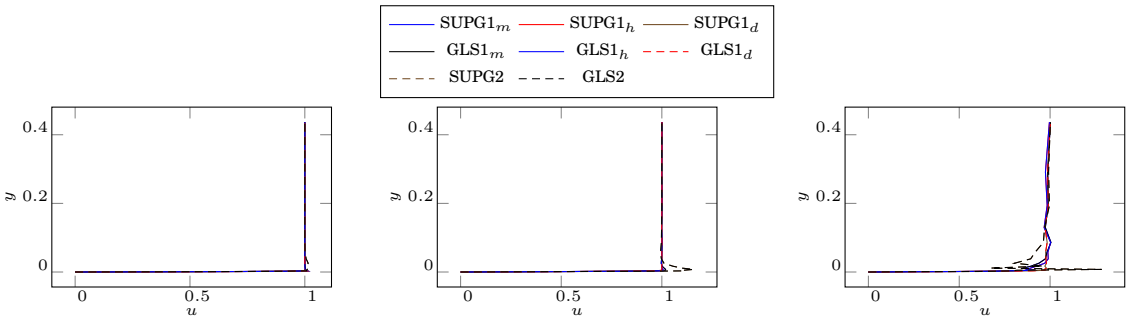


(a) SUPG1$_m$ (b) SUPG1$_h$ (c) SUPG1$_d$ (d) SUPG2



(e) GLS1$_m$ (f) GLS1$_h$ (g) GLS1$_d$ (h) GLS2

Figure 3.12 – Thermal Layer on Structured Anisotropic Mesh. $\sigma = 0$, $\kappa = 10^{-8}$. Comparison of the two methods with different parameters and characteristic length

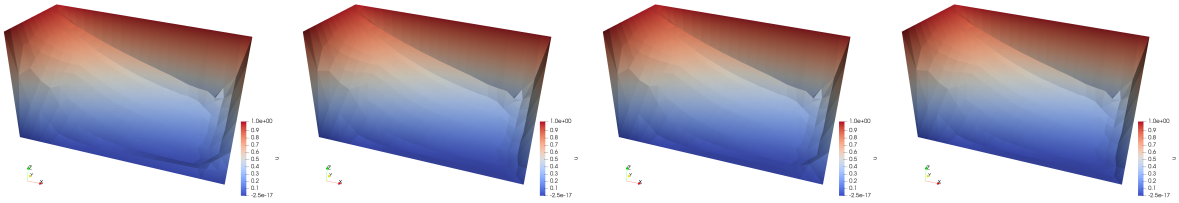


(a) Vertical Cut $x = 0.2$ (b) Vertical Cut $x = 0.5$ (c) Vertical Cut $x = 0.8$

Figure 3.13 – Thermal Layer on Structured Anisotropic Mesh. $\sigma = 0$, $\kappa = 10^{-8}$. Comparison of the two methods with different parameters and characteristic length

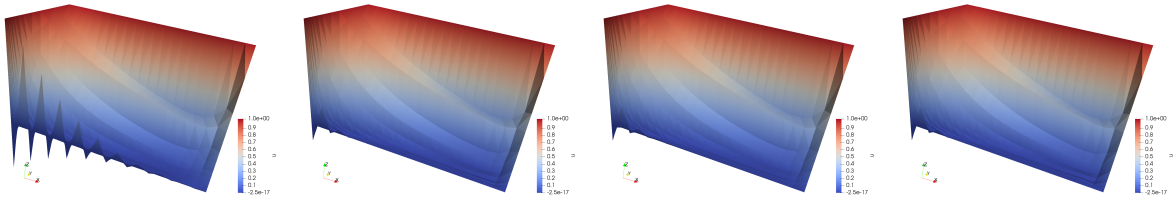
The second configuration proposed on this geometry is setup to illustrate the importance of the characteristic length definition for anisotropic meshes. On figures 3.12 and 3.13, we notice boundary oscillation for τ_h^2 and τ_h^1 with h_m and h_h versions. The h_d definition is the most accurate for anisotropic mesh since it takes into account the direction and the magnitude of the advection field. We will prefer this definition for problems with boundary layers. In this case again, the choice of the method between SUPG and GLS does not significantly influence the results.

As a last configuration on the thermal layer problem, we now add reaction term, $\sigma = 1$. This last study aims to discuss the interest of the parameter τ_h^3 which takes into account the value of the reaction coefficient σ. Since the parameter τ_h^2 is significantly less accurate than τ_h^1 we chose to only perform comparisons with this latter parameter. In both cases, we set $h = h_d$ which is the most robust definition. The comparison is made on the unstructured homogeneous mesh with $\kappa = 10^{-5}$, figures 3.14 and 3.16, and also on the structured one with $\kappa = 10^{-8}$, figures 3.15



(a) SUPG1$_d$ (b) SUPG3$_d$ (c) GLS1$_d$ (d) GLS3$_d$

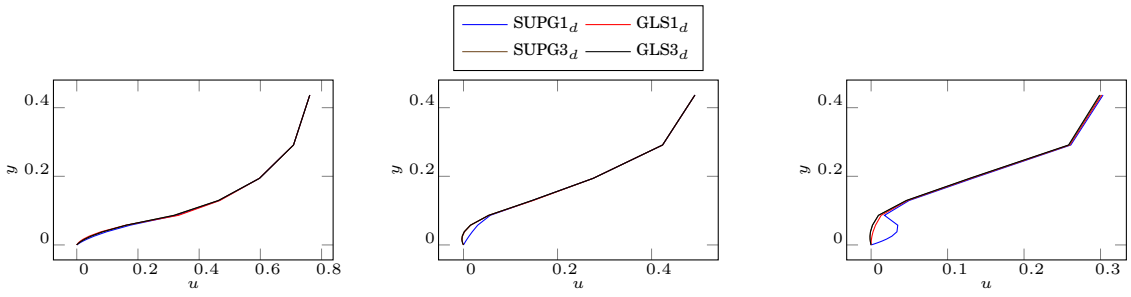
Figure 3.14 – Thermal Layer on Coarse Unstructured Homogeneous Mesh. $\sigma = 1$, $\kappa = 10^{-5}$. Comparison of the two methods with different parameters and characteristic length



(a) SUPG1$_d$ (b) SUPG3$_d$ (c) GLS1$_d$ (d) GLS3$_d$

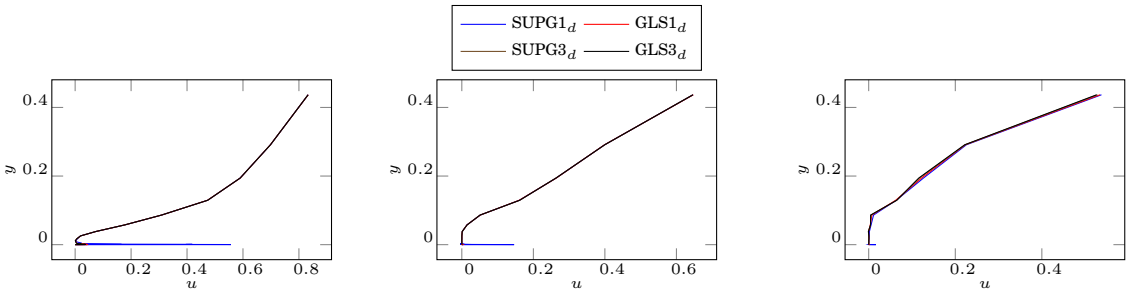
Figure 3.15 – Thermal Layer on Structured Anisotropic Mesh. $\sigma = 1$, $\kappa = 10^{-8}$. Comparison of the two methods with different parameters and characteristic length

and 3.17. On the homogeneous mesh, both parameters and both methods have comparable results. Nevertheless, we can denote a slight smoothing of the solution with parameter τ_h^3. On the structured mesh, the activation of the reaction generates non-physical oscillations on the boundaries. Those oscillations are significantly bigger in the SUPG1$_d$ configuration. Using either GLS instead of SUPG or τ_h^3 instead of τ_h^1 reduces this numerical issue. However, we never succeeded in completely removing this annoying side effect. After many efforts, the best results we obtained are those presented for GLS3$_d$. Note that the use of shock capturing methods does not improve the results.



(a) Vertical Cut $x = 0.2$ (b) Vertical Cut $x = 0.5$ (c) Vertical Cut $x = 0.8$

Figure 3.16 – Thermal Layer on Coarse Unstructured Homogeneous Mesh. $\sigma = 1$, $\kappa = 10^{-5}$. Comparison of the two methods with different parameters and characteristic length



(a) Vertical Cut $x = 0.2$ (b) Vertical Cut $x = 0.5$ (c) Vertical Cut $x = 0.8$

Figure 3.17 – Thermal Layer on Structured Anisotropic Mesh. $\sigma = 1$, $\kappa = 10^{-8}$. Comparison of the two methods with different parameters and characteristic length

4.2 Shock Capturing Methods

The different methods tested in the present subsection are: ID-GC, from Galeão and Carmo (3.48) and (3.49) , ID-A from Almeida *et al.*(3.50), OD-CK from Codina, modified by Knobloch *et al.*(3.53) and OD-K proposed by Knobloch *et al.*(3.54). We also present the results of our HOME shock capturing parameter, (3.55). According to the previous results, the choice of method and characteristic length definition does not have real influence on the solution, for this test case. All simulations are then performed with the method denoted by SUPG1$_d$.

Advection Skew the Mesh

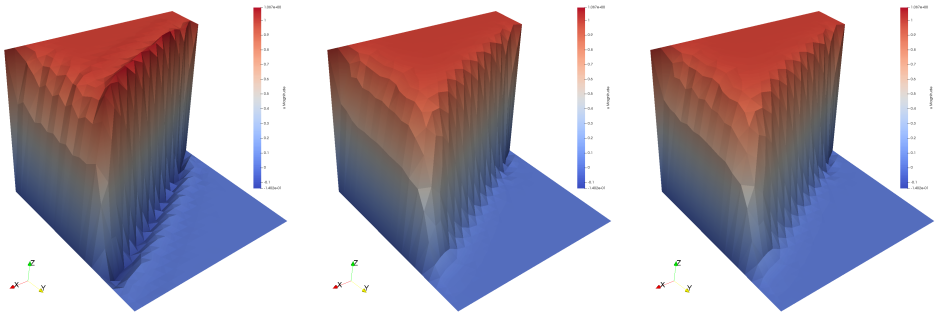
We are considering again the advection skew the mesh test-case presented in section 4.1. The aim of the shock capturing methods is the reduction of the undershoots and overshoots produced by streamline diffusion methods, as identified in figure 3.3.

Starting with a qualitative observation of the profiles on figure 3.18 we notice reduction of the under/overshoots. To have a better idea, we propose a cut of these profiles at $x = 0.5$, see figure 3.19. All the methods are smoothing the profile and the under/overshoot entirely disappears except for OD-CK where the oscillations are only reduced.

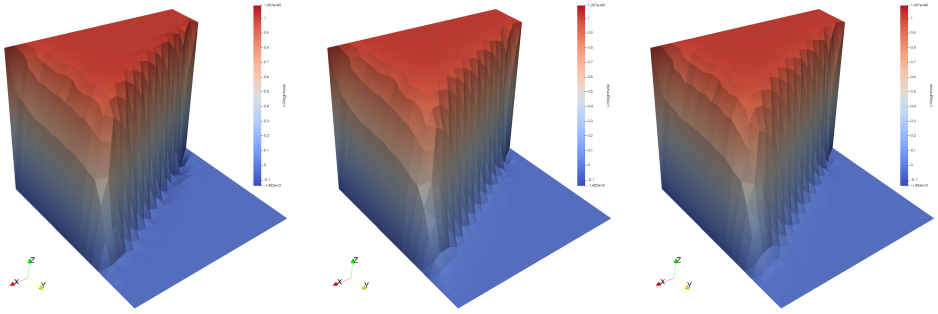
Advection in a Rotating Flow Field

We observed in the latter test case, the efficiency of the different methods. Nevertheless, those methods are reducing the non-physical oscillations by adding diffusivity. We now have to ensure that the considered methods are not too diffusive. For that, we compare their results on the advection in a rotating field test case.

The problem is defined on a unit square ($-0.5 \leq x, y \leq 0.5$), see figure 3.20 with



(a) SUPG1$_d$ (b) ID-GC (c) ID-A



(d) OD-CK (e) OD-CK (f) HOME

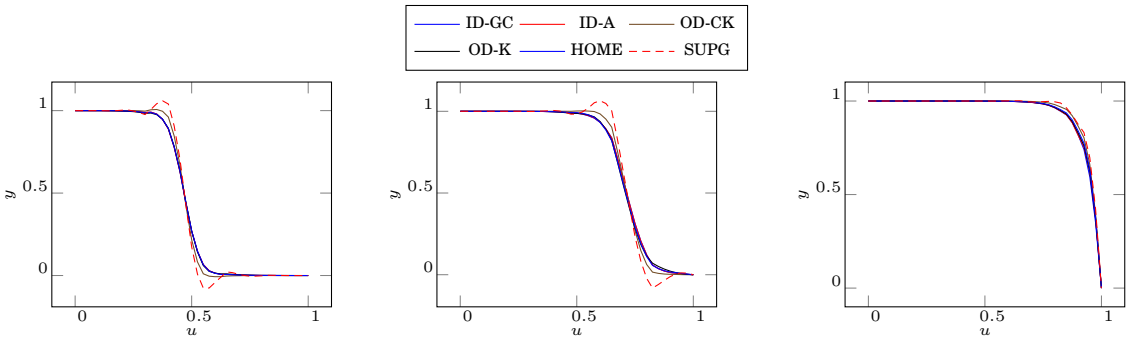
Figure 3.18 – Advection Skew the Mesh profiles for $\alpha = \text{atan}(0.5)$.
Comparison of the different shock capturing methods, using SUPG1$_d$ as stabilization method.

a velocity flow given by $a = (-y, x)$. The boundary conditions are given by $u = 0$ on all the external boundaries and we add an internal boundary condition with

$$u = \frac{1}{2}(\cos(4\pi y + \pi) + 1) \tag{3.57}$$

on the segment $[OA]$, defined by $x = 0$, $-0.5 \leq y \leq 0.5$. The diffusivity is set to $\kappa = 10^{-8}$.

With this setup, the problem is almost purely advective, and so we should not see any loss of energy. If the numerical scheme is too diffusive, a discontinuity will appear along the segment $[OA]$. It is not supposed to happen with streamline diffusion methods which only add diffusivity along the streamline. However, for shock capturing method, we do see a discontinuity, see figure 3.21. As expected, the SUPG used solo does not cause any loss of energy. For the shock capturing methods, the results are very different. OD-K was one of the most effective methods when considering results on the previous test case, but it produces here a massive loss of energy. The discontinuity is less pronounced for ID-A and ID-GC and even less for OD-CK, but this latter method was not very convincing on the previous test case. Finally, our HOME method has the best results in term of energy loss, with a very small discontinuity. Those observations are very encouraging, and we will use this method in further numerical experiences.



(a) Vertical Cut $x = 0.5$, $\alpha = \text{atan}(0.5)$ (b) Vertical Cut $x = 0.5$, $\alpha = \text{atan}(1)$ (c) Vertical Cut $x = 0.5$, $\alpha = \text{atan}(2)$

Figure 3.19 – Advection Skew the Mesh profiles for different values of α.
Comparison of the different shock capturing methods, using SUPG1$_d$ as stabilization method.

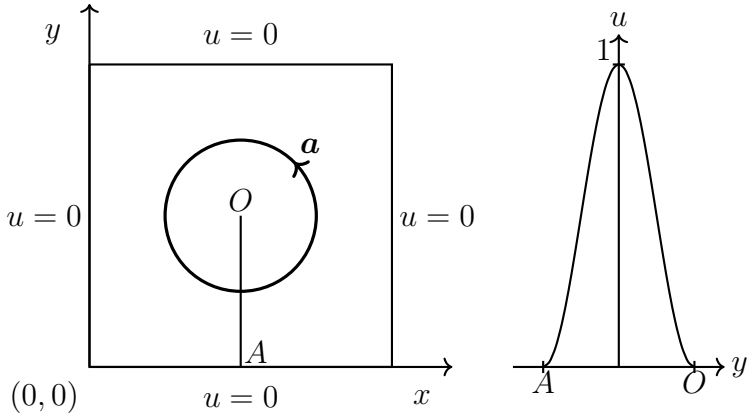


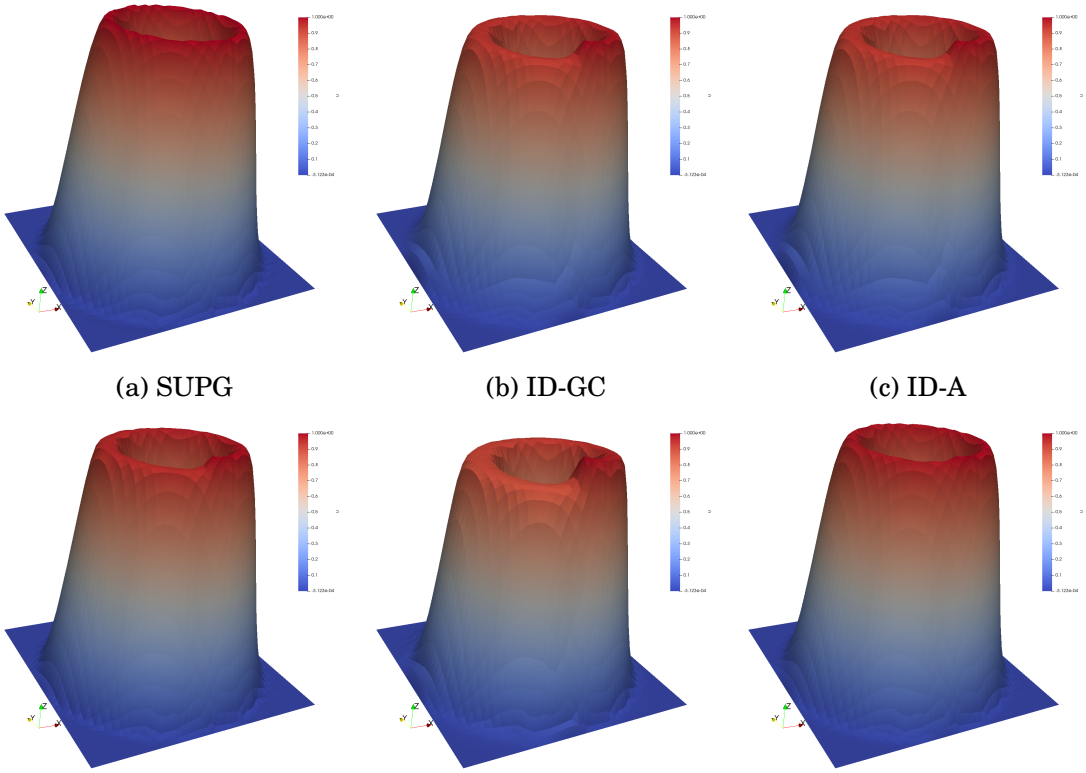
Figure 3.20 – Advection in a rotating flow field setup

4.3 Stabilized Navier-Stokes Equations

The previous preliminary results, on advection diffusion reaction problems, allows to efficiently chose the stabilization method, depending on the nature of the problem. In this section, we want to validate those results for the Navier-Stokes system.

The Driven Cavity

The steady NS equations are solved on a unit square $\Omega = [0,1] \times [0,1]$. A constant velocity field $\boldsymbol{u} = (1,0)$ is imposed on the top boundary $y = 1$, $0 \leq x \leq 1$. Homogeneous Dirichlet condition $\boldsymbol{u} = (0,0)$ is imposed on other boundaries, see figure 3.22. We run simulation with four different configurations: $\text{Re} = 400/h = 0.125$, $\text{Re} = 1000/h = 0.125$, $\text{Re} = 5000/h = 0.0625$ and $\text{Re} = 10000/h = 0.03$. All those simulations are performed on a non-structured mesh. The discretization is a standard Taylor-Hood FE space $\mathbb{TH}^1_h$. We performed all the stabilized simulation with the stabilization term on divergence: without this term, the results are not



(a) SUPG (b) ID-GC (c) ID-A

(d) OD-CK (e) OD-K (f) HOME

Figure 3.21 – Advection in a Rotating Flow Field.
Comparison of the different shock capturing methods, using SUPG1$_d$ as stabilization method.

conclusive for high Reynolds numbers. Our data are compared to reference values from [77]. In this unpublished study, the author made an accurate compilation of different results from many references, [108, 99, 20, 40, 14, 92, 93, 31] and also proposed his own results.

As for the advection-diffusion problem, we will compare the two methods SUPG and GLS, with parameters τ_h^1 and τ_h^2. As we figured out in the previous study, the influence of the characteristic length definition is negligible on homogeneous meshes, thus we only present results with definition h_m for this test case. For each configuration, we present: an horizontal cut along the axis $y = 0.5, 0 \leq x \leq 1$ on figure 3.24, a vertical cut along the axis $x = 0.5, 0 \leq y \leq 1$ on figure 3.23, a pressure profile with contour on figure 3.26and a velocity profile with streamlines on figure 3.25.

First, regarding the cuts 3.24 and 3.23, we observe some differences between the stabilized simulation and the reference for $\mathrm{Re} = 5000$ and $\mathrm{Re} = 10000$. The localization of the extremum is correct but not their values. We hypothesized that the mesh was too coarse to model the diffusion in the smaller scales accurately. We performed a new simulation on a thin mesh ($h = 0.01$) and obtained better results. This example perfectly illustrates the issue of using too coarse meshes with stabilization methods: those techniques may give a stable numerical solution,

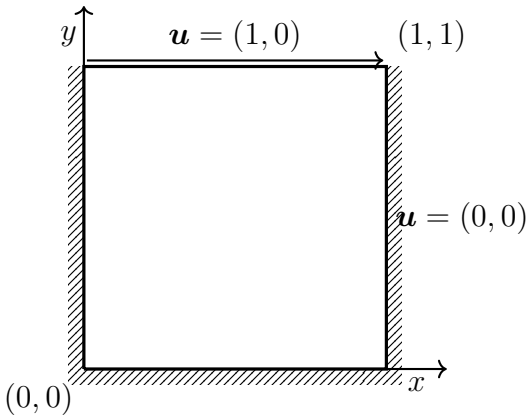
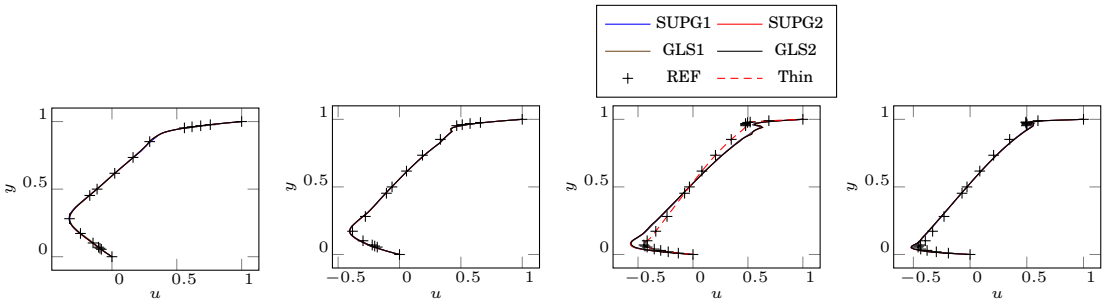


Figure 3.22 – Driven Cavity. Problem Setup



(a) Re = 400 (b) Re = 1000 (c) Re = 5000 (d) Re = 10000

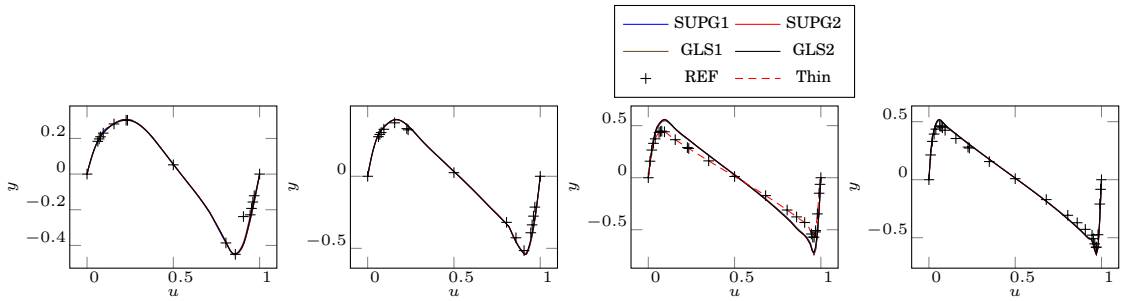
Figure 3.23 – Driven Cavity on Unstructured Homogeneous Mesh. u_x along a Vertical cut $x = 0.5$. Comparison of the two methods with different parameters

but will not properly model the diffusion in the small scales. In the next chapter, we present a solution to this issue, using turbulence models. Eventually, we can conclude in the efficiency of the two considered stabilization methods, as long as the mesh is not too coarse. The methods give comparable results for both parameters, τ_h^1 and τ_h^2, as illustrated on figures 3.25 and 3.26.

Fluid Layer

This second test case for stabilized Navier-Stokes equation was run on a flat plate geometry, proposed on NASA turbulence resources website[1], see figure 3.27. This test case was initially designed to benchmark turbulence codes, but for now, we only want to test the robustness of our Navier-Stokes solver with high Reynolds numbers and anisotropic meshes. We only consider the fluid resolution, without turbulence model. The test case will be studied again, with turbulence, in a further section. The geometry of the model is a rectangle domain $[-0.5, 2] \times [0, 1]$ with the following boundary conditions, see figure 3.28:

[1] https://turbmodels.larc.nasa.gov/



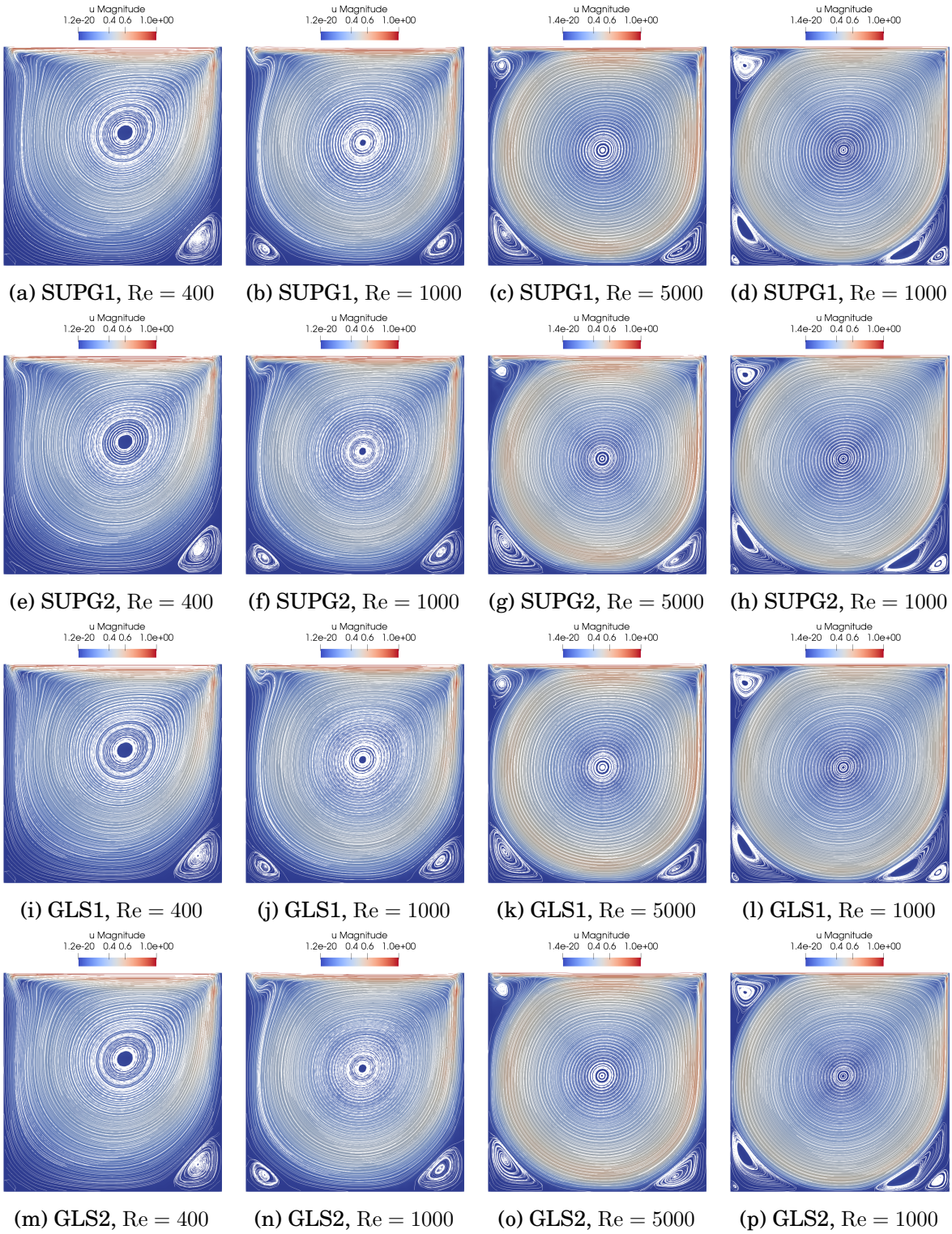
(a) Re = 400 (b) Re = 1000 (c) Re = 5000 (d) Re = 10000

Figure 3.24 – Driven Cavity on Unstructured Homogeneous Mesh. u_y along a Horizontal cut $y = 0.5$. Comparison of the two methods with different parameters

- Inlet ($x = 0.5, 0 \leq y \leq 1$): $\boldsymbol{u} = (1, 0)$,
- Solid Wall ($0 \leq x \leq 2, y = 0$): no-slip condition $\boldsymbol{u} = (0, 0)$,
- Bottom ($-0.5 \leq x \leq 0, y = 0$) : slip condition $\boldsymbol{u} \cdot \boldsymbol{n} = 0$, $(\boldsymbol{T}(\boldsymbol{u}, p)\boldsymbol{n})_{\boldsymbol{t}} = 0$,
- Top ($-0.5 \leq x \leq 2, y = 1$): slip condition $\boldsymbol{u} \cdot \boldsymbol{n} = 0$, $(\boldsymbol{T}(\boldsymbol{u}, p)\boldsymbol{n})_{\boldsymbol{t}} = 0$,
- Outlet ($x = 1, 0 \leq y \leq 1$): free stream condition $\boldsymbol{T}(\boldsymbol{u}, p)\boldsymbol{n} = (0, 0)$.

We faced different issues with this simulation. The geometry presents a harsh singularity at the point $(0, 0)$ as the boundary condition turns from slip to a no-slip condition. This discontinuity became annoying for high Reynolds numbers ($\mathrm{Re} \geq 10^4$). Another difficulty stands in the highly anisotropic mesh. We generated a simplex-mesh from the grid proposed by the NASA. All the rectangular cells were split into two triangles. In the boundary layer on the bottom, the cells have huge aspect ratios, and we had to adapt our stabilization parameters to fit with it. Initially, we performed simulations on the 35×35 grid but this mesh was too coarse, and the solution was not acceptable. We finally present the more satisfying results obtained on the 69×49 grid. The most challenging point of these simulations was the stability of the solution around the singularity at $(0, 0)$. To give a better view of the problem we present the velocity profile as a warp of the velocity field magnitude. This representation highlights the non-physical oscillations in the boundary layers, see figure 3.29. Note that those oscillations are not visible if you only display the streamlines of the velocity field. The pressure may also present significant fluctuations just after the discontinuity. We never succeeded in entirely removing those oscillations, but we aimed to keep them local near the singularity, see figure 3.30. We ran the simulations with high Reynolds number, $\mathrm{Re} = 10^5$. From a physical point of view, the results have no interest since the model lacks diffusivity, but for now, we are only interested in the numerical stability of the different configurations: SUPG/GLS, τ_1/τ_2, $h_m/h_h/h_d$.

From the results of figures 3.29 and 3.30, we clearly see the influence of the characteristic length. This test case confirms the practicality of the h_d version. This version will be now systematically used on anisotropic meshes. Again, we



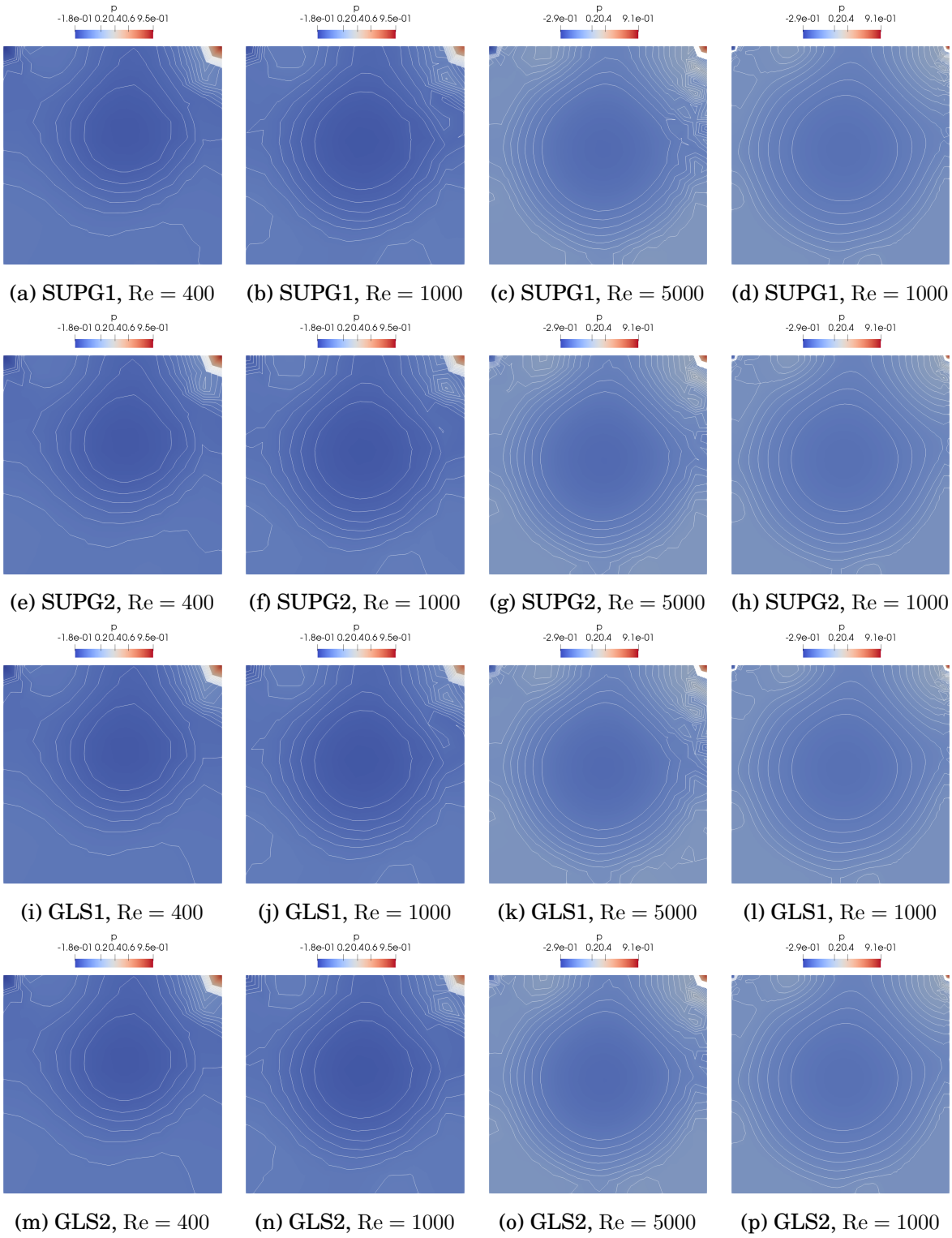
(a) SUPG1, Re = 400 (b) SUPG1, Re = 1000 (c) SUPG1, Re = 5000 (d) SUPG1, Re = 1000

(e) SUPG2, Re = 400 (f) SUPG2, Re = 1000 (g) SUPG2, Re = 5000 (h) SUPG2, Re = 1000

(i) GLS1, Re = 400 (j) GLS1, Re = 1000 (k) GLS1, Re = 5000 (l) GLS1, Re = 1000

(m) GLS2, Re = 400 (n) GLS2, Re = 1000 (o) GLS2, Re = 5000 (p) GLS2, Re = 1000

Figure 3.25 – Driven Cavity on Unstructured Homogeneous Mesh. Velocity Streamlines. Comparison of the two methods with different parameters



(a) SUPG1, Re = 400 (b) SUPG1, Re = 1000 (c) SUPG1, Re = 5000 (d) SUPG1, Re = 1000

(e) SUPG2, Re = 400 (f) SUPG2, Re = 1000 (g) SUPG2, Re = 5000 (h) SUPG2, Re = 1000

(i) GLS1, Re = 400 (j) GLS1, Re = 1000 (k) GLS1, Re = 5000 (l) GLS1, Re = 1000

(m) GLS2, Re = 400 (n) GLS2, Re = 1000 (o) GLS2, Re = 5000 (p) GLS2, Re = 1000

Figure 3.26 – Driven Cavity on Unstructured Homogeneous Mesh. Pressure Contours. Comparison of the two methods with different parameters

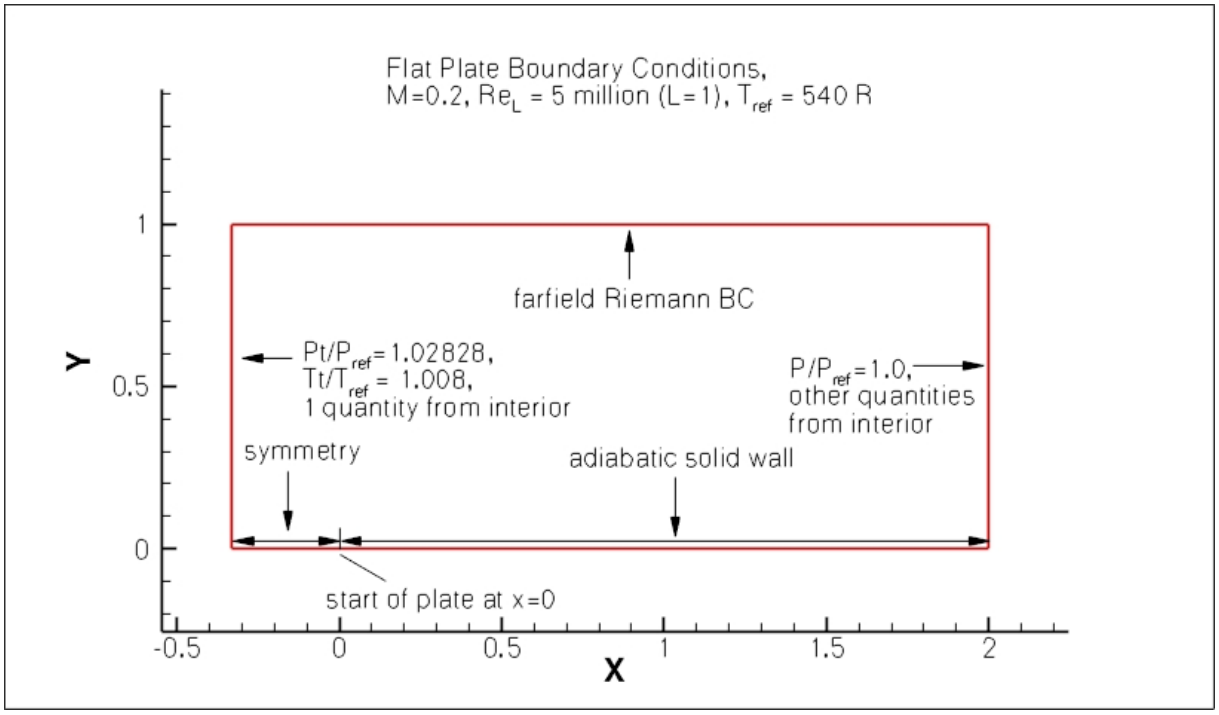


Figure 3.27 – Flate Plate Boundary Conditions, as presented on Nasa resource page.

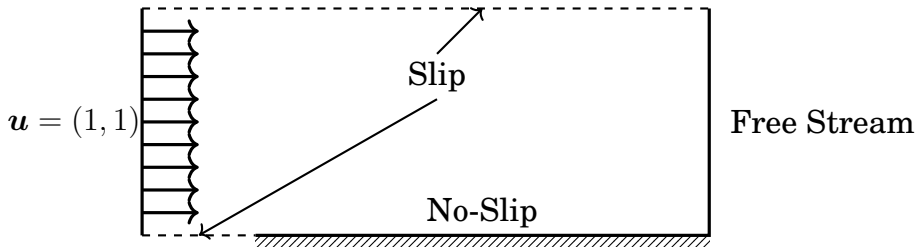
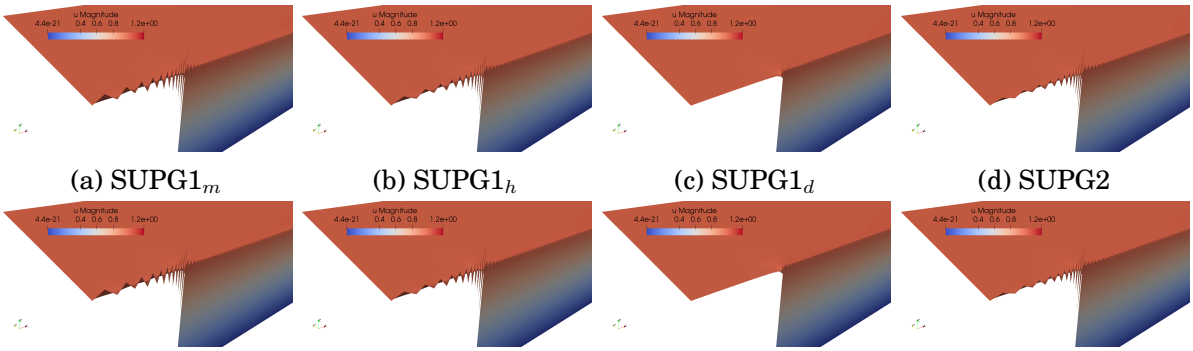


Figure 3.28 – Fluid Layer Setup

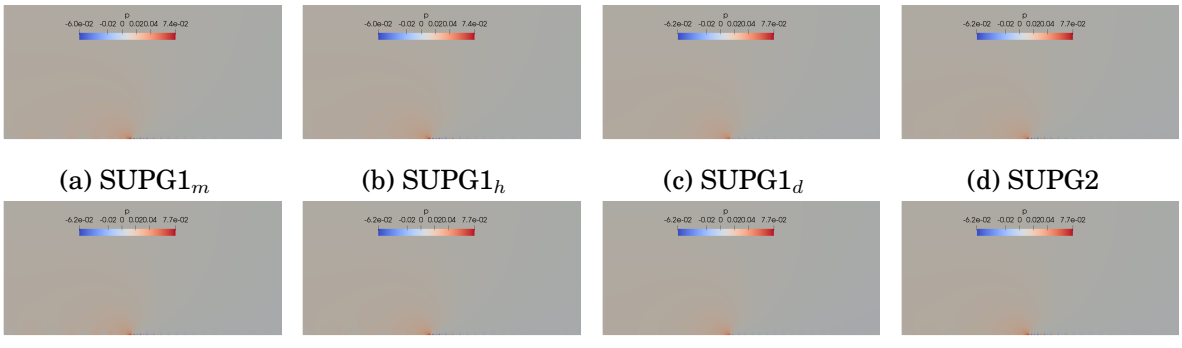
denote only small differences between the SUPG and GLS methods. However, the convergence of our non-linear solver is improved when we use the GLS, and this method will be preferred for high Reynolds simulations.



(a) SUPG1$_m$ (b) SUPG1$_h$ (c) SUPG1$_d$ (d) SUPG2

(e) GLS1$_m$ (f) GLS1$_h$ (g) GLS1$_d$ (h) GLS2

Figure 3.29 – Fluid Layer on structured anisotropic mesh with boundary structure on the bottom. Focus on the region around the singularity. Warp of the magnitude of the velocity field. Comparison of the two methods with different parameters



(a) SUPG1$_m$ (b) SUPG1$_h$ (c) SUPG1$_d$ (d) SUPG2

(e) GLS1$_m$ (f) GLS1$_h$ (g) GLS1$_d$ (h) GLS2

Figure 3.30 – Fluid Layer on structured anisotropic mesh with boundary structure on the bottom. Focus on the region around the singularity. Pressure profile. Comparison of the two methods with different parameters

Conclusion

In this chapter, we presented the stabilization strategy we developed in our aerothermal framework. The SDM is very efficient and offers an appropriate response to the numerical instability of the Galerkin method. The numerical examples of the last section illustrate the importance of the stabilization parameter. The method is usually very robust but encounters some difficulties on anisotropic meshes or near geometric singularities. In that cases, the formulation h_d proposed in this chapter appears to be particularly adapted. It also provides accurate results on homogeneous isotropic meshes. However, the computation of this characteristic length is a bit more expensive than the h_m or h_h versions which do not depend on the convection field.

For advection-diffusion problems, SUPG and GLS usually give similar solutions and the SUPG will be used by default since it is a bit cheaper to compute. However, for advection-diffusion-reaction or the Navier-Stokes equations, GLS produces significantly smoother results. In the table 3.1, we propose a recap of the recommended configurations for each kind of problems.

In all this chapter we only presented and discussed the formulations for triangular meshes. This constraint is proper to the Feel++ library which does not provide the required tools for hypercubes. We expected to recover a more stable system, especially in the boundary layers, when working with hypercubes. Unfortunately, the needed implementation in Feel++ was not fulfilled, and this conjecture remains an open question.

Those stabilization issues were more challenging than initially expected. We spend a considerable amount of time on this topic, but once those stabilization methods were fully mastered, it finally opened the way to the turbulence modeling.

	Isotropic Mesh			Anisotropic Mesh		
	SDM	τ	h	SDM	τ	h
Advection Diffusion	SUPG	τ_1 or τ_2	h_m or h_h	SUPG	τ_1	h_h
Advection Diffusion Reaction	GLS	τ_3	h_m or h_h	GLS	τ_3	h_d
Navier-Stokes	SUPG	τ_1 or τ_2	h_m or h_d	GLS	τ_1	h_d

Table 3.1 – Recap of recommended configurations for different kinds of problems.

CHAPTER 4

Turbulence Modeling

Contents

The behavior of a fluid is defined by the Reynolds number Re,

$$\mathrm{Re} = \frac{U_0 L}{\nu} \tag{4.1}$$

where U_0 is the far-field velocity magnitude, L is the characteristic length of the problem (usually the smallest significant length) and ν is the kinematic viscosity of the fluid. For low Reynolds number, the flow is laminar: the viscosity forces dominate, the model usually admits a steady state and is totally stable. When the

Reynolds number grows, the physic becomes less compliant: the stationary state is not guaranteed anymore, and we usually need stabilization methods if we do not want to pay the price of a very thin mesh. This kind of flow is not turbulent yet but can be defined as a transition state. If the flow is not stationary, he may recover a periodic behavior and is still deterministic. For higher Reynolds, the flow becomes turbulent: the behavior is chaotic with many scales of eddies.

The methodologies presented in the previous chapter provide an accurate framework for modelization of laminar and transitional flows but are inefficient in the simulation of turbulent problems. Even if the stabilization method allows to perform simulations with high Reynolds numbers, they do not take care of the energy which is supposed to be diffused in the smallest scales. We could theoretically solve all the different scales and perform what is know as a Direct Numerical Simulation (DNS) but for very high Reynolds number, the required time and space discretization would be ridiculously not practicable [57].

Here comes the necessity to introduce a turbulence model in order to modelize all the energy exchanges, at all scales, but with no proper resolution and a reasonable discretization. We mainly distinguish two families of turbulence model:

- The Large Eddy Simulation (LES) in which only the small structures are modelized. The larger structures are solved using Navier-Stokes equations. This method is particularly adapted to transient problems and was not studied in detail during this thesis.

- The Reynolds Average Navier-Stokes (RANS) method which consists in the averaging of the chaotic component of the flow. The next section is dedicated to this method and will present the main ingredients.

The field of turbulence modelization is very dense and complicated (especially when you want to tackle it with a mathematical background). In this chapter, we only briefly remind the equations and the main ingredient of the presented models. For more details, we refer the reader to [41], and the further cited references.

1 Reynolds Averaged Navier-Stokes Method

1.1 Time-Averaged Equations

We detail here the averaging of the Navier-Stokes equations, in order to recover the RANS equations. The treatment of the energy equation will be discussed further.

Reynolds Decomposition

The Reynolds Decomposition is used in order to separate the value of a quantity into its mean value and its fluctuations,

$$\boldsymbol{u} = \bar{\boldsymbol{u}} + \boldsymbol{u}', \tag{4.2}$$

where $\bar{u}$ is the mean value, also called steady or averaged component, and u' is the fluctuations.

The Reynolds Averaged Navier-Stokes (RANS) equations are obtained by considering the Reynolds decomposition of the two fluid quantities

$$\begin{aligned} \boldsymbol{u} &= \bar{\boldsymbol{u}} + \boldsymbol{u}' \\ p &= \bar{p} + p' \end{aligned} \tag{4.3}$$

Averaged Equations

After substitution of (4.3) into the Navier-Stokes equations, we obtain

$$\begin{cases} \rho(\bar{\boldsymbol{u}} + \boldsymbol{u}') \cdot \nabla(\bar{\boldsymbol{u}} + \boldsymbol{u}') + \nabla(\bar{p} + p') - \mu\Delta(\bar{\boldsymbol{u}} + \boldsymbol{u}') = \mathbf{f} \\ \nabla \cdot (\bar{\boldsymbol{u}} + \boldsymbol{u}') = 0. \end{cases} \tag{4.4}$$

This system of equation is then averaged. In the couple further formulas, we chose to use the tensor notation which was a bit more convenient.

$$\begin{cases} \overline{\rho(\bar{\boldsymbol{u}}_j + \boldsymbol{u}'_j) \cdot \frac{\partial(\bar{\boldsymbol{u}}_i + \boldsymbol{u}'_i)}{\partial x_j} + \frac{\partial(\bar{p} + p')}{\partial x_i} - \mu\frac{\partial^2(\bar{\boldsymbol{u}}_i + \boldsymbol{u}'_i)}{\partial x_j x_j}} = \overline{\mathbf{f}_i} \\ \overline{\frac{\partial(\bar{\boldsymbol{u}}_i + \boldsymbol{u}'_i)}{\partial x_i}} = 0. \end{cases} \tag{4.5}$$

Given that $\bar{\cdot}$ is a Reynolds operator, we may use the following properties,

$$\bar{a'} = 0, \quad \bar{\bar{a}} = \bar{a}, \quad \overline{a+b} = \bar{a} + \bar{b}, \quad \overline{ab} = \bar{a}\bar{b} + \overline{a'b'}, \tag{4.6}$$

to rewrite the previous equation (4.5),

$$\begin{cases} \rho\bar{\boldsymbol{u}}_j\frac{\partial\bar{\boldsymbol{u}}_i}{\partial x_j} + \rho\overline{\boldsymbol{u}'_j\frac{\partial\boldsymbol{u}'_i}{\partial x_j}} + \frac{\partial\bar{p}}{\partial x_i} - \mu\frac{\partial^2\bar{\boldsymbol{u}}_i}{\partial x_j\partial x_j} = \overline{\mathbf{f}_i} \\ \frac{\partial\bar{\boldsymbol{u}}_i}{\partial x_i} = 0. \end{cases} \tag{4.7}$$

This formulation is very close to the classic incompressible Navier-Stokes equations, except for the term $\overline{\boldsymbol{u}'_j\frac{\partial\boldsymbol{u}'_i}{\partial x_j}}$ which can be rewritten (using the continuity equation of the fluctuation) as $\frac{\partial\overline{\boldsymbol{u}'_i\boldsymbol{u}'_j}}{\partial x_i}$. The quantity $\boldsymbol{R}_{ij} = \rho\overline{\boldsymbol{u}'_i\boldsymbol{u}'_j}$ is known as the **Reynolds**

stress tensor and represents the effects of velocity fluctuation on the averaged system. With this notation, the system (4.7) reads,

$$\begin{cases} \bar{\boldsymbol{u}}_j \dfrac{\partial \bar{\boldsymbol{u}}_i}{\partial x_j} + \dfrac{\partial \bar{p}}{\partial x_i} - \mu \dfrac{\partial^2 \bar{\boldsymbol{u}}_i}{\partial x_j \partial x_j} + \rho \dfrac{\partial \boldsymbol{R}_{ij}}{\partial x_j} = \overline{\mathbf{f}_i} \\ \dfrac{\partial \bar{\boldsymbol{u}}_i}{\partial x_i} = 0. \end{cases} \tag{4.8}$$

Using the Reynolds decomposition in this averaged equations, we introduced 3 unknown quantities: $\bar{\boldsymbol{u}}$, $\bar{p}$ and $\boldsymbol{R}$. Unfortunately, we have gained no additional equations. We now need to close this system with additional equations. That closure is provided by the different turbulence models. In what follow we focus on models derived from the Boussinesq approximation.

1.2 The Boussinesq Approximation and Derived Models

The key of the RANS approach is to find an appropriate model for the Reynolds stress tensor. A common method is to relate the Reynolds stresses to the mean velocity gradients

$$-\boldsymbol{R}_{ij} = \mu_t \frac{\partial \boldsymbol{u}_i}{\partial x_j} - \frac{2}{3}\rho k \delta_{ij} \tag{4.9}$$

where the turbulence kinetic energy is defined as $k = \frac{1}{2}\overline{\boldsymbol{u}_i' \boldsymbol{u}_j'}$ and μ_t is the eddy viscosity.

The RANS models derived from the Boussinesq approximation provide closure equations in order to compute μ_t and k. Many different models can be found in the literature. In the simplest models, the kinetic energy is neglected, and the turbulent viscosity is directly express from the velocity fields. Those algebraic models are usually not accurate enough. The most common models express the turbulent viscosity μ_t and the kinetic energy k as solutions of one (Spalart-Allmaras, Baldwin-Barth, ...) or two equations ($k-\epsilon$, $k-\omega$, SST, ...). Those models are reasonably complex to be solved efficiently and are sufficiently accurate in their dedicated fields of application. More complex models exist but were not studied.

In the context of this thesis, we were first interested in the Spalart-Allmaras model which is known as efficient and relatively simple. In a second time, we chose to implement a solver for the $k-\omega$-SST model which appears to be more adapted to the kind of problems we were studying. Those two models are detailed in the following sections.

1.3 Resolution Strategy

The resolution of the coupled RANS problems is integrated into the iterations of the non-linear solver for the aerothermal problem described in chapter 2. Turbulent problems requires both physical and Ψtc continuations, the turbulence model is then activated when the molecular viscosity reaches a limit value μ^* in the physical continuation process. Then, the turbulent viscosity μ_t is updated at each iteration in the non-linear solver (usually during the update of the Jacobian in Newton algorithm). The actual computation of μ_t depends on the choice of the model and will be detailed for each of them in the following sections.

We summarized the resolution procedure in the algorithm

Algorithm 1: Resolution Procedure for RANS Model using Newton Algorithm

```
/* We denote by W the composite vector W = (u_h, p_h, T_h) */
Choose μ*;                                          // Limit viscosity
Choose N;    // Number of steps in the physical continuation
Choose δ_0 ;                    // Initial Ψtc pseudo time-step
Choose ε;                     // Tolerance for Newton algorithm
for 0 ≤ n ≤ N do                    // Physical continuation loop
    μ_n ← μ^(n/N);
    κ_n ← κ^(n/N);
    δ ← δ_0;
    repeat                                        // Newton iterations
        if μ_n ≤ μ* then
            Update μ_t^k with W^k;          // Solve turbulence model PDE
        else
            μ_t ← 0;
        end
        Solve ((1/δ) D + F'(W^k; μ^n, κ_n, μ_t^k)) S = −F(W^k; μ^n, κ_n, μ_t^k);
        W^k ← W^k + S;
        Update δ;                                       // See section 3
    until ‖F(W^k; μ_n, κ_n, μ_t)‖_2 ≤ ε;
end
return W^n
```

2 Spalart Allmaras Model

The Spalart Allmaras (SA) [96] model is probably one of the most used RANS model. He is very popular thanks to its relative simplicity and accuracy. In this

model, the kinetic energy k is neglected and the eddy viscosity μ_t is computed through the resolution of a non-linear PDE.

2.1 Equations

The model is described by one transport equation. The calibration of the many closure constants and more details can be read in [95, 50].

The eddy viscosity is given by $\mu_t = \rho\bar{\nu} f_{v1}$, where $\bar{\nu}$ is the solution of

$$\underbrace{\boldsymbol{u}\cdot\nabla\bar{\nu}}_{\text{Convection}} - \underbrace{\frac{1}{\sigma}[\nabla\cdot((\nu+\bar{\nu})\nabla\bar{\nu}) + c_{b2}(\nabla\bar{\nu}\cdot\nabla\bar{\nu})]}_{\text{Diffusion}} - \underbrace{c_{b1}(1-f_{t2})S\bar{\nu}}_{\text{Production}} + \underbrace{[c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}]\left(\frac{\bar{\nu}}{d}\right)^2}_{\text{Destruction}} = 0, \tag{4.10}$$

with the following notations,

$$\begin{aligned}
&\chi = \frac{\bar{\nu}}{\nu}, && S = \sqrt{(\nabla\times\boldsymbol{u})\cdot(\nabla\times\boldsymbol{u})} + \frac{\bar{\nu}}{\kappa^2 d^2}f_{v2},\\
&f_{v1} = \frac{\chi^3}{\chi^3+c_{v1}^3}, && f_{v2} = 1-\frac{\chi}{1+\chi f_{v1}}, \qquad f_{t2} = c_{t3}\exp(-c_{t4}\chi^2),\\
&f_w = g\left(\frac{1+c_{w3}^6}{g^6+c_{w3}^6}\right)^{1/6}, && g = r + c_{w2}(r^6-r), \qquad r = \min\left(\frac{\bar{\nu}}{S\kappa^2 d^2}, 10\right),
\end{aligned} \tag{4.11}$$

where d is the distance from the closest surface. We furthermore consider the constant parameters

$$\begin{aligned}
&c_{b1} = 0.1355, \quad c_{b2} = 0.622, \quad c_{w1} = \frac{c_{b1}}{\kappa^2} + \frac{1+c_{b2}}{\sigma}, \quad c_{w2} = 0.2, \quad c_{w3} = 2\\
&c_{v1} = 7.1, \qquad c_{t3} = 1.2, \qquad c_{t4} = 0.5, \qquad \kappa = 0.41, \qquad \sigma = 2/3.
\end{aligned} \tag{4.12}$$

The problem is closed with the following boundary conditions,

$$\begin{cases}
\bar{\nu} &= 0, \quad \text{on } \Gamma_W \text{ (Walls)}\\
\bar{\nu} &= 3\nu, \quad \text{on } \Gamma_I \text{ (Inlets)}\\
\nabla\bar{\nu}\cdot\boldsymbol{n} &= 0, \quad \text{on } \Gamma_O \text{ (Outlets),.}
\end{cases} \tag{4.13}$$

3 $k-\omega$ SST model

The $k-\omega$ SST model is known to be very robust and adaptive. The model was first proposed in [70] and [71]. The model was then improved in [72]. We chose to work on this formulation, known as the SST-2003 model. Note that many other versions of the model were proposed, for instance in [94].

3.1 The Equations

The turbulent eddy viscosity is computed by

$$\mu_t = \frac{\rho a_1 k}{\max(a_1\omega, \boldsymbol{S}F_2)} \tag{4.14}$$

where the turbulent kinetic energy k and the specific dissipation ω are solutions of the coupled transport equations

$$\begin{cases} \dfrac{\partial k}{\partial \omega} + \boldsymbol{u}\nabla k + \beta^* k\omega - \nabla\cdot[\dfrac{\mu+\sigma_k\mu_t}{\rho}\nabla k] = P_k, \\ \dfrac{\partial \omega}{\partial t} + \boldsymbol{u}\nabla\omega + \beta\omega^2 - \nabla\cdot[\dfrac{\mu+\sigma_w\mu_t}{\rho}\nabla\omega] = \dfrac{\gamma}{\mu_t}P_k + 2(1-F1)\dfrac{\sigma_{w2}}{\omega}\nabla k\cdot\nabla\omega, \end{cases} \tag{4.15}$$

with the following closure relations

$$F_1 = \tanh(\arg_1^4), \qquad \arg_1 = \min\left[\max\left(\frac{\sqrt{k}}{\beta^*\omega y}, \frac{500\mu}{\rho y^2\omega}\right), \frac{4\sigma_{w2}k}{CD_{k\omega}y^2}\right] \tag{4.16}$$

$$F_2 = \tanh(\arg_2^2), \qquad \arg_2 = \max\left(\frac{2\sqrt{k}}{\beta^*\omega y}, \frac{500\mu}{\rho y^2\omega}\right) \tag{4.17}$$

$$P_k = \min(\bar{\tau}:\nabla\boldsymbol{u}, 10\beta^* k\omega), \quad CD_{k\omega} = \max\left(2\rho\frac{\sigma_{w2}}{\omega}\nabla k\nabla\omega, 10^{-10}\right), \tag{4.18}$$

$$\boldsymbol{S} = \sqrt{2(\bar{S}:\bar{S})}. \tag{4.19}$$

Each closure constant is a blend of an inner and an outer constant, via

$$\phi = F_1\phi_1 + (1-F_1)\phi_2 \tag{4.20}$$

where ϕ_1 represents constant 1 and ϕ_2 represents constant 2. The closure constants are

$$\begin{array}{llll} \sigma_{k1} = 0.85, & \sigma_{\omega 1} = 0.5, & \beta_1 = 0.075, & \gamma_1 = 5/9 \\ \sigma_{k2} = 1.0, & \sigma_{\omega 2} = 0.856, & \beta_2 = 0.0828, & \gamma_2 = 0.44 \\ \beta^* = 0.09, & \kappa = 0.41, & a_1 = 0.31. & \end{array} \tag{4.21}$$

We also remind some usual notations. $\bar{\tau}$ is the Reynolds stress tensor, defined, for incompressible flow, by

$$\bar{\tau}_{ij} = 2\mu_t\bar{S}_{ij} - \frac{2}{3}\rho k\delta_{ij} \tag{4.22}$$

where δ_{ij} is the Kronecker delta and $\bar{S}$ is given by stress tensor,

$$\bar{S}_{ij} = \frac{1}{2}\left(\frac{\partial \boldsymbol{u}_i}{\partial x_j} + \frac{\partial \boldsymbol{u}_j}{\partial x_i}\right). \tag{4.23}$$

We can also rewrite this tensors as

$$\bar{\tau} = 2\mu_t\bar{S} - \frac{2}{3}\rho k\mathbb{I}_d, \qquad \bar{S} = \frac{1}{2}(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T) \tag{4.24}$$

3.2 Boundary and Initial Conditions

As initial values for k and ω, we choose

$$k_0 = \left(\frac{\mu_0}{\rho l_0}\right)^2, \qquad \omega_0 = \frac{k^{1/2}}{l_0} \tag{4.25}$$

where $\mu_0 = \mathcal{O}(\mu)$ is a constant initial viscosity and l_0 is a default length such that $l_0 \in [l_{\min}, l_{\max}]$, $l_{\min}$ corresponding to the size of the smallest admissible eddies.

At the inflow boundary the prescribed value are

$$k = c_{bc}|u|^2, \qquad \omega = \frac{k^{1/2}}{l_0}, \quad \text{on } \Gamma_{\text{in}}, \tag{4.26}$$

where $c_{bc} \in [0.003, 0.01]$ is an empirical constant.

At the walls, we impose slip condition on the velocity

$$\boldsymbol{u} \cdot \boldsymbol{n} = 0, \quad \text{on } \Gamma_{\text{w}}. \tag{4.27}$$

Wall functions

We denote by $\boldsymbol{t}_w$ the tangential component of the force exerted by the viscous stress tensor $\bar{\varepsilon}(\boldsymbol{u}) = \frac{\mu+\mu_t}{\rho}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)$ on Γ_w

$$\boldsymbol{t}_w = \boldsymbol{n} \cdot \bar{\varepsilon}(\boldsymbol{u}) - ((\boldsymbol{n} \cdot \varepsilon(\boldsymbol{u})) \cdot \boldsymbol{n})\boldsymbol{n}. \tag{4.28}$$

We consider an internal boundary Γ_y located at a distance y from the physical wall Γ_w. Assuming that we are in the log-layer region, the logarithm wall laws leads to the boundary conditions

$$\boldsymbol{t}_w = -u_\tau \frac{\boldsymbol{u}}{|\boldsymbol{u}|}, \qquad k = \frac{u_\tau^2}{\sqrt{\beta^*}}, \qquad \omega = \frac{u_\tau}{\sqrt{\beta^*}\kappa y} \tag{4.29}$$

where $\beta^* = 0.09$, $\kappa = 0.41$ is the von Karman constant. The friction velocity satisfy the equation

$$\frac{|\boldsymbol{u}|}{u_\tau} = \frac{1}{\kappa}\log y^+ + \beta \tag{4.30}$$

assuming that the local Reynolds number $y^+ = \frac{\rho u_\tau y}{\mu}$ is in the range $11.06 \leq y^+ \leq 300$. In theory we have to impose the wall function on the inner boundary Γ_y and the boundary layer of width y should be removed from the domain Ω. In practice, the wall function is applied on the first degree of freedom at the interior of the domain but we can not guarantee that this first dof will not fall into the viscous layer where the current model would not be valid anymore.

To avoid this issue, we chose a solution proposed in [39]. We assume that the first node always corresponds to a fixed value of y^+. Since the deviation to the wall

is assumed to remain very small, we can neglect the width of this layer and then we avoid a costly mesh adaptation for each iteration. The wall function are then directly applied on Γ_w, just as if the nodes of this boundary were shifted in the normal direction by the distance y.

The accuracy of this model depends of the choice of y which has to be chosen as small as possible. [39] explains that the smallest acceptable value for y corresponds to meeting point between the log-layer and the viscous sub-layer. At this point y^+ verifies the both equations

$$\frac{|\boldsymbol{u}|}{u_\tau} = \frac{1}{\kappa}\log y^+ + \beta, \tag{4.31}$$

$$\frac{|\boldsymbol{u}|}{u_\tau} = y^+. \tag{4.32}$$

We can then identify the corresponding value, $\tilde{y}^+$, which verifies

$$\tilde{y}^+ = \frac{1}{\kappa}\log \tilde{y}^+ + \beta, \tag{4.33}$$

for the default value of $\kappa = 0.41$ and $\beta = 5.2$, the non linear equation(4.33) can be solved iteratively and we find $\tilde{y}^+ = 11,06$. All the nodes of Γ_w are now supposed to be at a distance y of the physical wall such that $y^+ = \tilde{y}^+$.

On this boundary Γ_w we can now evaluate the friction velocity u_τ

$$u_\tau = \frac{|\boldsymbol{u}|}{\frac{1}{\kappa}\log \tilde{y}^+ + \beta} = \frac{|u|}{\tilde{y}^+} \tag{4.34}$$

on the other hand, by considering (4.29) we have the relation $u_\tau = (\beta^*)^{\frac{1}{4}}\sqrt{k}$, [39] proposes to set

$$u_\tau = \max\left((\beta^*)^{\frac{1}{4}}\sqrt{k}, \frac{|\boldsymbol{u}|}{\tilde{y}^+}\right), \qquad \boldsymbol{t}_w = -\frac{u_\tau}{\tilde{y}^+}\boldsymbol{u}. \tag{4.35}$$

Resulting of (4.35) we have natural boundary condition to plug in the variational formulation for the RANS equation

$$\int_{\Gamma_w} \boldsymbol{t}_w \cdot \boldsymbol{v}\, d\Gamma = -\int_{\Gamma_w} \frac{u_\tau}{\tilde{y}^+}\boldsymbol{u}\cdot\boldsymbol{v}\, d\Gamma \tag{4.36}$$

when the condition (4.27) will be imposed weakly with term

$$\int_{\Gamma_w} \gamma_h(\boldsymbol{u}\cdot\boldsymbol{n})\cdot(\boldsymbol{v}\cdot\boldsymbol{n})\, d\Gamma = 0 \tag{4.37}$$

with $\gamma_h = \gamma \max\left(\frac{\mu+\mu_t}{h_e}, |\boldsymbol{u}|\right)$, where γ is a penalty constant and h_e is the characteristic length of the cell, see 3 for the evaluation of h_e.

Using (4.29) we now evaluate the eddy viscosity on Γ_w, assuming that the model for this value of y^+ will act as a classic $k-\omega$ model,

$$\mu_t = \rho\frac{k}{\omega} = \rho u_\tau \kappa y = \kappa \tilde{y}^+ \mu \tag{4.38}$$

This value is automatically satisfied if we strongly impose the value of k and ω on Γ_w, but this would results in a one way coupling for the boundary conditions between the fluid equations and the $k-\omega$ system. To avoid this problem, we chose to weakly impose the boundary condition for k and ω,

$$\nabla k \cdot \boldsymbol{n} = -\frac{\partial k}{\partial y} = 0, \tag{4.39}$$

$$\nabla w \cdot \boldsymbol{n} = -\frac{\partial \omega}{\partial y} = \frac{u_\tau}{\sqrt{\beta^* \kappa y^2}} = \frac{\omega}{y} \tag{4.40}$$

we can then replace the unknown $y = \frac{\tilde{y}^+ \mu}{\rho u_\tau}$, and we obtain the natural weak formulation for the boundary condition on Γ_w

$$\int_{\Gamma_w} \frac{\mu + \sigma_k \mu_t}{\rho} \nabla k \cdot nw \; d\Gamma = 0, \tag{4.41}$$

$$\int_{\Gamma_w} \frac{\mu + \sigma_\omega \mu_t}{\rho} \nabla \omega \cdot nw \; d\Gamma = \int_{\Gamma_w} \frac{\mu + \sigma_\omega \mu_t}{\mu} \frac{u_\tau}{\tilde{y}^+} \omega w \; d\Gamma. \tag{4.42}$$

Since these two quantities are not imposed strongly, we have to impose the exact value for the eddy viscosity μ_t using the relation (4.34) and the production term P_k on Γ_k,

$$P_k = 10\beta^* \rho \omega k = 10\frac{u_\tau^3}{\kappa y} = 10\frac{\rho u_\tau^4}{\mu_t} = 10\frac{\rho u_\tau}{\kappa \tilde{y}^+ \mu_t} \tag{4.43}$$

4 Validation: 2D Naca0012 Airfoil

In this section we propose to validate our implementation of the presented turbulence models on a classic 2D benchmark. This validation case is proposed on the NASA Turbulence Modeling Resource page[1]

4.1 Problem Description

The geometry is a 2D cut of a NACA0012 airfoil, in sufficiently large domain to neglect the side effects. The airfoil surface is defined by the curve

$$y = \pm 0.5946891(0.2982227\sqrt{x} - 0.1271252x - 0.3579079x^2 + 0.2919849x^3 - 0.1051746x^4). \tag{4.44}$$

The figure 4.1 is an overview of the domain, discretized with a grid proposed by the NASA. The problem is described for compressible code but with low mach number $M = 0.15$. The Reynolds number is $\mathrm{Re} = 6 \cdot 10^6$. It corresponds to a viscosity $\mu = 0.16 \cdot 10^{-6}$ with a density $\rho = 1$, a farfield velocity $V = 1$ and the characteristic length of the airfoil $L = 1$.

[1] https://turbmodels.larc.nasa.gov/index.html

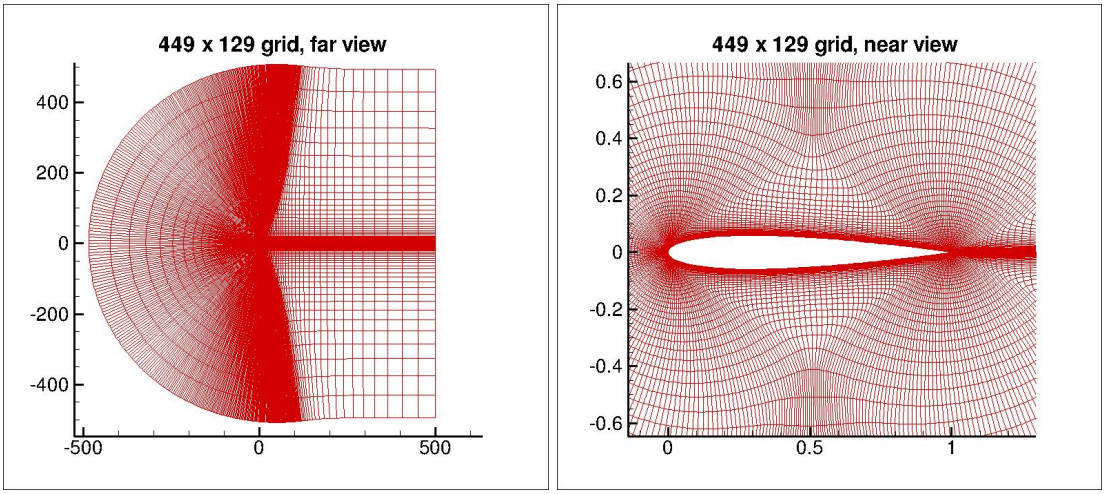


Figure 4.1 – NACA0012 : Grid 449×129 from the NASA website

Conclusion

Part II

Model Order Reduction

CHAPTER 5

The Reduced Basis Method

Contents

The efficient resolution of parametrized PDEs is of great interest for both industry and academic research. With the development of super calculators and suitable methods, it is now possible to perform simulations of complex 3D models with high accuracy. However, the needs of high-speed simulations are also growing. For instance in the context of risk analysis or uncertainty quantification where thousands of simulations are needed. For those many-query methods, it is not realistic to perform a full order resolution (FEM, FVM, ...) of an industrial problem, for each new set of parameters. A solution is to replace the full order model with a reduced one. Different methods of reduction exist: surrogate models, proper generalized decomposition, proper orthogonal decomposition, reduced basis, ...

In this work, we chose to focus on the Reduced Basis Method (RBM) which presents some interesting characteristics:

- Rigorous a posteriori error estimators
- An efficient Offline/Online decomposition.

Those features are provided by two necessary ingredients: a Galerkin projection onto a low-dimensional space spanned by properly selected basis functions and an affine parametric dependency.

The development of the RBM started in the late 70's, initially for non-linear mechanics problems [76]. The method was then extended to many different kinds of problems [2, 32, 78, 84]. The RBM knew in the early 2000's an important improvement with the development of the Certified Reduced Basis Method (CRBM) [80, 103, 81, 83, 88]. The main improvement of the CRBM is the use of efficient error estimators, evaluated through an offline/online procedure. Those error estimators allow quantifying the error made with the reduced model. This error is usually measured on a quantity of interest, called the output of the model. A direct consequence of this fast error evaluation is the possibility to explore the parameter space efficiently and to optimally select the parameters used to build the reduced space X_N.

This chapter is split into three sections. The basics of the method are introduced in the first section using the convenient example of the linear coercive problems. In the second section, we will introduce some specific features, required for the reduction of saddle point models. In the last part we detail the algorithm applied to aerothermal simulations.

1 Method and Notations : Application to Coercive Problems

We first introduce the reduced basis method (RBM) in a very general context.

Let $\Omega \in \mathbb{R}^d$, $d = 2, 3$ be a bounded domain and $X \subset H^1(\Omega)$ a Hilbert space on this domain, we denote by $(.,.)_X$ and $||.||_X = \sqrt{(.,.)_X}$ respectively the inner product and associated norm on X. We also introduce a parameter space $\mathcal{D} \in \mathbb{R}^P$ of dimension $P \geq 1$. Let us consider, for a first time, a very generic problem, depending on a parameter $\boldsymbol{\mu} \in \mathcal{D}$, as finding $u(\boldsymbol{\mu}) \in X$ such that

$$\mathcal{F}(u(\boldsymbol{\mu}); \boldsymbol{\mu}) = 0. \tag{5.1}$$

In the context of parametrized PDE, the resolution of this problem using the FEM as discretization method will lead to the resolution of the discrete problem: find

$u_h(\boldsymbol{\mu}) \in X_h$ such that

$$\mathcal{F}_h(u_h(\boldsymbol{\mu}); \boldsymbol{\mu}) = 0 \tag{5.2}$$

where $X_h \subset X$ is a finite element function space of high dimension $\mathcal{N}$, as introduced in section2.

The resolution of this discrete problem can be very costly, even with an optimized method and comfortable computation capabilities. The idea of the reduced basis method, and more generally of the different model order reduction methods is to build a low (relatively to $\mathcal{N}$) dimensional space in which the resolution of problem (5.2) will give an acceptable approximation.

Note that when we use the RBM, we make a non-trivial assumption on the manifold

$$\mathcal{S}(\mathcal{D}) = \{u(\boldsymbol{\mu}) | \mathcal{F}(u(\boldsymbol{\mu}); \boldsymbol{\mu}) = 0, \boldsymbol{\mu} \in \mathcal{D}\}, \tag{5.3}$$

which has to be approximable by a finite-dimensional function space X_N of relatively small dimension N. The Kolmogorov N-width determines the feasibility of such an approximation. We will not detail this notion in this work, but we assume that, for the considered problems, the Kolmogorov N-width will decay sufficiently rapidly and then it makes sense to consider a reduced model.

In the context of RBM, this space X_N is usually constructed as

$$X_N = \text{span}\{u_h(\boldsymbol{\mu}), \boldsymbol{\mu} \in S_N\} \tag{5.4}$$

where $S_N \subset \mathcal{D}$ is a sample of parameters chosen in $\mathcal{D}$. The construction of this sample can be more or less optimized and will determine the quality of the reduced basis space X_N. For instance, we can imagine choosing S_N as a random or an equidistributed subset of $\mathcal{D}$, but more efficient techniques have been developed using, for instance, a greedy algorithm to optimally select the parameters in S_N.

Parameters Selection through Greedy Algorithm

The greedy algorithm builds a sample S_N in order to minimize a quantity $\Delta_N(\boldsymbol{\mu})$ on $\mathcal{D}$. For instance, this quantity could be the error, $\tilde{\Delta}_N(\boldsymbol{\mu}) = ||u_h(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})||$, between the FEM solution $u_h(\boldsymbol{\mu})$ and the reduced solution $u_N(\boldsymbol{\mu})$. But the algorithm requires the evaluation of the maximum of this quantity on the entire parameter space $\mathcal{D}$, $\tilde{\Delta}_N = \max_{\boldsymbol{\mu} \in \mathcal{D}} \tilde{\Delta}_N(\boldsymbol{\mu})$. Such maximum is obviously not computable.

Hence we introduce two significant simplifications. First, we compute this maximum on a discrete subset $\Xi \subset \mathcal{D}$ (potentially very large). Then, we do not use the real error $\tilde{\Delta}_N(\boldsymbol{\mu})$ but error indicators $\Delta_N(\boldsymbol{\mu})$ which can be evaluated for many values $\boldsymbol{\mu}$ for a minimal cost. Alternatively, we might be more interested in the error measured for a specific quantity of interest: the output of the model. The greedy construction remains the same using an output-based error bound. The nature of these error indicators is problem-dependent and will be detailed further for the different kinds of considered PDE.

We present in the algorithm 2 a generic version of the greedy algorithm. The general idea of the algorithm does not depend on the nature of the problems.

Algorithm 2: Generic Greedy Algorithm

Choose $\Xi \subset \mathcal{D}$; // Super Sampling
Choose $N_{\max}$; // Maximum size of X_N
Choose $\delta_{tol} \in (0,1)$; // Algorithm tolerance
Choose $\boldsymbol{\mu}_1 \in \mathcal{D}$; // Arbitrary chosen first parameter
Set $N \leftarrow 0$, $S_N \leftarrow \{\}$, $X_N \leftarrow \{\}$; // Initialization
repeat
  $N \leftarrow N + 1$;
  $S_N \leftarrow S_{N-1} \cup \{\boldsymbol{\mu}_N\}$;
  Solve $\mathcal{F}_h(u_h(\boldsymbol{\mu}); \boldsymbol{\mu})$;
  $X_N \leftarrow X_{N-1} \oplus \text{span}\{u_h(\boldsymbol{\mu}_N)\}$;
  for $\mu \in \Xi$ **do**
    Compute error indicator, $\Delta_N(\boldsymbol{\mu})$;
  end
  $\mu_{N+1} = \arg\max_{\boldsymbol{\mu} \in \Xi} \Delta_N(\mu)$;
until $\Delta_N(\boldsymbol{\mu}_{N+1}) < \delta_{tol}$ **or** $N \geq N_{max}$;

1.1 Reduced Basis for Coercive Problems

We detail in this section the use of the RBM for coercive problems. The problem is defined as in the previous section. We also introduce a parametrized bilinear form $a(.,.;\boldsymbol{\mu}) : X \times \mathcal{D} \to \mathbb{R}$ and a parametrized linear form $f(.;\boldsymbol{\mu}) : X \times \mathcal{D} \to \mathbb{R}$. The problem (5.1) can be written as the variational formulation of a parametrized PDE, then we want to find $u(\boldsymbol{\mu}) \in X$ such that

$$a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f(v; \boldsymbol{\mu}), \quad \forall v \in X. \tag{5.5}$$

We also do some assumptions on this parametrized weak formulation, for any value of $\boldsymbol{\mu} \in \mathcal{D}$:

(a) $a(.,.;\boldsymbol{\mu})$ is symmetric and positive definite. We can then define a scalar product and the associated norm

$$(.,.)_{\boldsymbol{\mu}} = a(.,.;\boldsymbol{\mu}), \qquad ||.||_{\boldsymbol{\mu}} = \sqrt{(.,.\boldsymbol{\mu})}. \tag{5.6}$$

In practice we will essentially use the scalar product $(.,.)_{\bar{\mu}}$ and norm $||.||_{\bar{\mu}}$ associated to a reference parameter $\bar{\boldsymbol{\mu}}$.

(b) $a(.,.;\boldsymbol{\mu})$ is continuous and coercive with respect to the norm $||.||_X$, i.e., there exist a positive coercivity constant $\alpha(\boldsymbol{\mu}) > 0$ and a finite continuity constant

$\gamma(\boldsymbol{\mu}) < \infty$ such that

$$a(v, v; \boldsymbol{\mu}) \geq \alpha(\boldsymbol{\mu})||v||_X^2, \qquad a(w, v; \boldsymbol{\mu}) \leq \gamma(\boldsymbol{\mu})||w||_X||v||_X, \qquad \forall v, w \in X \tag{5.7}$$

These constants are defined respectively as

$$\alpha(\boldsymbol{\mu}) = \inf_{v \in X} \frac{a(v, v; \boldsymbol{\mu})}{||v||_X^2}, \qquad \gamma(\boldsymbol{\mu}) = \sup_{w \in X} \sup_{v \in X} \frac{a(w, v; \boldsymbol{\mu})}{||v||_X||w||_X}, \qquad \forall \boldsymbol{\mu} \in \mathcal{D} \tag{5.8}$$

(c) The forms $a(., .; \boldsymbol{\mu})$ and $f(.; \boldsymbol{\mu})$ have affine decompositions

$$a(w, v; \boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \theta_q^a(\boldsymbol{\mu}) a_q(w, v), \tag{5.9}$$

$$f(v; \boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \theta_q^f(\boldsymbol{\mu}) f_q(v) \tag{5.10}$$

where each form

$$a_q : X \times X \to \mathbb{R}, \qquad\qquad f_q : X \to \mathbb{R}$$

is independent of parameter $\boldsymbol{\mu}$ and each

$$\theta_q^a : \mathcal{D} \to \mathbb{R}, \qquad\qquad \theta_q^f : \mathcal{D} \to \mathbb{R}$$

is a function which only depends on $\boldsymbol{\mu}$.

This affine decomposition is essential to provide an efficient offline/online strategy. This point is detailed in subsection 1.1.

Offline Procedure: Construction of the RB Space

We construct the reduced basis space using the greedy algorithm as proposed in section 1. We assume that we have a good error bound $\Delta_N(\boldsymbol{\mu})$, defined on a quantity of interest. For coercive problems, the evaluation of this error bound is detailed in section 1.2.

The first parameter $\boldsymbol{\mu}_1$ to initialize the greedy procedure is usually arbitrarily chosen. With $\boldsymbol{\mu}_1$ we obtain the first snapshot $u_h(\boldsymbol{\mu}_1)$ which is not directly used as basis vector, but normalized, with respect to the specific norm $||.||_{\bar{\mu}}$, as $\xi_1 = \frac{u_h}{||u_h||_{\bar{\mu}}}$. The reduced basis space X_1 is then defined as $X_1 = \text{span}\{\xi_1\}$.

After this initialization, at any rank N, we use the greedy algorithm (2) to chose the next parameter $\boldsymbol{\mu}_{N+1}$ and then the next snapshot $u_h(\boldsymbol{\mu}_{N+1})$. The next basis vector ξ_{N+1} is obtained using the Gram-Schmidt process on the set of vectors

$(\xi_1, ..., \xi_N, u(\boldsymbol{\mu}_{N+1}))$, still with respect to the norm $||.||_{\bar{\mu}}$. And eventually, the reduced space is defined as

$$X_{N+1} = \text{span}\{\xi_n, 1 \leq n \leq N+1\}. \tag{5.11}$$

The recursive construction ends when either the expected error or the max size N is reached, as proposed in the algorithm (2). This way, a nested set of reduced basis spaces is built, $X_1 \subset X_2 \subset ... \subset X_N$. Note that even if the orthonormalization process on the basis is not mandatory, it allows controlling the conditioning of the reduced matrix obtained after the Galerkin projection. Otherwise, this matrix can be very ill-conditioned, and it generates numerical instability.

The reduced solution $u_N(\boldsymbol{\mu})$, on X_N, can then be written as

$$u_N(\boldsymbol{\mu}) = \sum_{i=1}^{N} u_{Ni}(\boldsymbol{\mu})\xi_i \tag{5.12}$$

the coefficient $u(\boldsymbol{\mu})_{Ni}$ are evaluated during the online phase of the RBM, detailed in next paragraph.

Online Evaluation : Computation of $u_N(\boldsymbol{\mu})$

Once the reduced space is built we can use this structure to solve the problem introduced in (5.5) on X_N, for any new parameter $\boldsymbol{\mu} \in \mathcal{D}$. The reduced solution $u_N(\boldsymbol{\mu})$ is then the Galerkin projection of the continuous solution $\boldsymbol{u}(\boldsymbol{\mu})$ on the reduced basis $\{\xi_i, 1 \leq i \leq N\}$. This projection implies the resolution of the linear problem

$$A_N(\boldsymbol{\mu})u_N = F_N(\boldsymbol{\mu}) \tag{5.13}$$

where

$$(A_N(\boldsymbol{\mu}))_{ij} = a(\xi_i, \xi_j; \boldsymbol{\mu}), \qquad (F_N(\boldsymbol{\mu}))_i = f(\xi_i; \boldsymbol{\mu}), \qquad \forall\, 1 \leq i, j \leq N. \tag{5.14}$$

The size $N \times N$ of this problem is supposed to be very small compared to the finite element dimension $\mathcal{N}$. The order of the model is reduced as expected. However the assembly of the reduced matrix $A_N(\boldsymbol{\mu})$ and the reduced vector $F_N(\boldsymbol{\mu})$ still involve the finite element dimension through the evaluation of the quantities $a(\xi_i, \xi_j; \boldsymbol{\mu})$ and $f(\xi_i; \boldsymbol{\mu})$. To be efficient the online part has to be totally independent of the full dimension $\mathcal{N}$. The assumption of affine decomposition introduced in (5.9) allows to decompose the evaluation of $A_N(\boldsymbol{\mu})$ and $F_N(\boldsymbol{\mu})$ in a online/offline procedure, with a $\mathcal{N}$-independent online part.

Using (5.9) we can rewrite (5.14)

$$\begin{aligned}(A_N(\boldsymbol{\mu}))_{ij} &= a(\xi_i,\xi_j;\boldsymbol{\mu}), & (F_N(\boldsymbol{\mu}))_i &= f(\xi_i;\boldsymbol{\mu}),\\ &= \sum_{q=1}^{Q_a}\theta_q^a(\boldsymbol{\mu})a_q(\xi_i,\xi_j), & &= \sum_{q=1}^{Q_f}\theta_q^f(\boldsymbol{\mu})f_q(\xi_i)\\ &= \sum_{q=1}^{Q_a}\theta_q^a(\boldsymbol{\mu})\underbrace{(A_N^q)_{ij}}_{\text{precomputed}}, & &= \sum_{q=1}^{Q_f}\theta_q^f(\boldsymbol{\mu})\underbrace{(F_N^q)_i}_{\text{precomputed}}, \qquad \forall\ 1\le i,j\le N.\end{aligned} \tag{5.15}$$

Thanks to the affine decomposition we can now precompute the matrices A_N^q and the vectors F_N^q during the offline phase, since they do not depend of the parameter $\boldsymbol{\mu}$. Hence the online phase only consist in a sum of Q_a $N\times N$-matrices, a sum of Q_f N-size vectors and the resolution of the $N\times N$ linear system.

During the offline phase, we have to introduce a new step in the construction of the nested RB spaces: the update of the precomputed A_N^q and F_N^q. Thanks to the nested structure of the RB spaces X_N, we only have to evaluate the last row, and the last column of the A_N^q matrices, the other values are similar to A_{N-1}^q. The same argument allows only to compute the last value of the vectors F_N^q.

1.2 Error Estimation

The computation of an efficient error bound for coercive problems has been widely studied [80, 21] during last decades, we remind in this section the main ingredients. In the context of the parametrized PDE, we may be interested either in the solution itself or by the evaluation of a particular quantity of interest (output). In this latter situation, we need some assumption on the considered quantity, $s(\boldsymbol{\mu})$.

We assume the existence of a parametrized linear form $l(.;\boldsymbol{\mu}) : X\times\mathcal{D}\to\mathbb{R}$ such that the output $s(\boldsymbol{\mu})$ can be expressed as

$$s(\boldsymbol{\mu}) = l(u(\boldsymbol{\mu});\boldsymbol{\mu}). \tag{5.16}$$

We furthermore assume that the form l admits an affine decomposition

$$l(v;\boldsymbol{\mu}) = \sum_{q=1}^{Q_l}\theta_q^l(\boldsymbol{\mu})l_q(v) \tag{5.17}$$

with the conditions stated in (5.9).

For an efficient error bound of the error $|s_h(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})|$ we need to introduce a dual problem associated to the linear form $l(.;\boldsymbol{\mu})$, as find $\psi(\boldsymbol{\mu})\in X$ such that

$$a(v,\psi(\boldsymbol{\mu});\boldsymbol{\mu}) = -l(v;\boldsymbol{\mu}), \quad \forall v\in X. \tag{5.18}$$

For this dual problem we build a dual reduced space

$$X_N^{du} = \text{span}\{\xi_n^{du}, 1 \leq n \leq N\} \tag{5.19}$$

where the basis vectors ξ_n^{du} are generated through a Gram-Schmidt algorithm on the snapshots $\psi_h(\boldsymbol{\mu}_n)$ as we did for the construction of the primal reduced space X_N.

Once this dual problem and reduced space are introduced, we can define the first main ingredient for the error bound: the dual norm of the residuals associated with the primal and dual problem, respectively $\epsilon_N(\boldsymbol{\mu})$ and $\epsilon_N^{du}(\boldsymbol{\mu})$, defined as

$$\epsilon_N(\boldsymbol{\mu}) = \sup_{v \in X_h} \frac{r_N(v;\boldsymbol{\mu})}{||v||_{\bar{\boldsymbol{\mu}}}} = ||\hat{e}_N(\boldsymbol{\mu})||_{\bar{\boldsymbol{\mu}}} \tag{5.20}$$

$$\epsilon_N^{du}(\boldsymbol{\mu}) = \sup_{v \in X_h} \frac{r_N^{du}(v;\boldsymbol{\mu})}{||v||_{\bar{\boldsymbol{\mu}}}} = ||\hat{e}_N^{du}(\boldsymbol{\mu})||_{\bar{\boldsymbol{\mu}}} \tag{5.21}$$

where

$$r_N(v;\boldsymbol{\mu}) = f(v;\boldsymbol{\mu}) - a(u_N(\boldsymbol{\mu}), v;\boldsymbol{\mu}) \tag{5.22}$$

$$r_N^{du}(v;\boldsymbol{\mu}) = l(v;\boldsymbol{\mu}) + a(v, \psi_N(\boldsymbol{\mu});\boldsymbol{\mu}), \quad \forall v \in X_h \tag{5.23}$$

are the residuals, respectively of the primal and dual problems. $\hat{e}_N(\boldsymbol{\mu}) \in X_h$ and $\hat{e}_N^{du}(\boldsymbol{\mu}) \in X_h$ are their Riesz representations which satisfy

$$(\hat{e}_N(\boldsymbol{\mu}), v)_{\bar{\boldsymbol{\mu}}} = r_N(v;\boldsymbol{\mu}) \tag{5.24}$$

$$(\hat{e}_N^{du}(\boldsymbol{\mu}), v)_{\bar{\boldsymbol{\mu}}} = r_N^{du}(v;\boldsymbol{\mu}), \quad \forall v \in X_h \tag{5.25}$$

the evaluation of these Riesz representations through a efficient offline/online procedure is detailed in the appendix 3.1.

The second ingredient for the error bound are lower bound $\alpha_{LB}(\boldsymbol{\mu})$ and an upper bound $\alpha_{UB}(\boldsymbol{\mu})$ for the coercivity constant introduce in (5.8)

$$\alpha_{LB}(\boldsymbol{\mu}) \leq \alpha(\boldsymbol{\mu}) \leq \alpha_{UB}(\boldsymbol{\mu}), \quad \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{5.26}$$

The evaluation of these bounds has to be independent of the FE dimension $\mathcal{N}$. The successive constraint method (SCM) [47, 89, 101] has been developed specifically to provide an efficient computation of such bounds for coercive or inf-sup problems. In the context of coercive problems, we can also use the Min$-\theta$ approach [63, 104]. We did not detail these methods in this report and refer the interested reader to the cited publications.

We can now introduce the error bounds for the errors on the reduced solution $\Delta_N(\boldsymbol{\mu})$ and for the output $\Delta_N^s(\boldsymbol{\mu})$ defined as

$$\Delta_N(\boldsymbol{\mu}) = \frac{\epsilon_N(\boldsymbol{\mu})}{(\alpha_{LB}(\boldsymbol{\mu}))^{1/2}}, \tag{5.27}$$

$$\Delta_N^{du}(\boldsymbol{\mu}) = \frac{\epsilon_N^{du}(\boldsymbol{\mu})}{(\alpha_{LB}(\boldsymbol{\mu}))^{1/2}}, \tag{5.28}$$

$$\Delta_N^s(\boldsymbol{\mu}) = \Delta_N(\boldsymbol{\mu})\Delta_N^{du}(\boldsymbol{\mu}), \tag{5.29}$$

and satisfying

$$||u_h(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})||_{\bar{\boldsymbol{\mu}}} \leq \Delta_N(\boldsymbol{\mu}) \tag{5.30}$$
$$|s_h(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})| \leq \Delta_N^s(\boldsymbol{\mu}). \tag{5.31}$$

For a study of effectiveness of these error estimators, we refer the reader to [80, 89, 9]. Once we have an efficient procedure for the evaluation of the error bounds, it can be used within the greedy algorithm 2 in order to guide the parameters selection. The evaluation of the reduced output using a dual problem (during the online part) is then made through

$$s_N(\boldsymbol{\mu}) = l(u_N(\boldsymbol{\mu}); \boldsymbol{\mu}) - r_N(\psi_N(\boldsymbol{\mu}); \boldsymbol{\mu}), \quad \forall \boldsymbol{\mu} \in \mathcal{D} \tag{5.32}$$

Remark 6 (Error bound for compliant problems)**.** *A compliant problem is defined when the linear form defining the output also defines the right hand side of the variational formulation* (5.5) *:* $l(.;\boldsymbol{\mu}) = f(.;\boldsymbol{\mu})$*. In this case we don't need the resolution of the dual problem. The error bound on the output is simply defined as* $\Delta_N^s(\boldsymbol{\mu}) = (\Delta_N(\boldsymbol{\mu}))^2$

Remark 7 (Problem with no output)**.** *When we do not need the evaluation of a particular output or when we need to evaluate many outputs, we will directly use the error bound on the solution* $\Delta_N(\boldsymbol{\mu})$ *instead of an error bound on the output. In that case, we will not use the dual problem but we lose the quadratic convergence of the reduced solution. Another solution is proposed in [109] for a generalization to vector valued quantities of interest.*

The two ingredients are evaluated through an efficient offline/online strategy. The coercivity constant can be computed using the Successive Constraint Method (SCM) [47, 89, 101]. The dual norm of the residual is partly precomputed offline, using the affine decomposition. This procedure is detailed in appendix 3.1.

Summarized Offline Algorithm for Coercive Problems

We complete the generic algorithm 2 with the different procedures introduced in this section. Some steps might not be necessary, for instance, if we do not need the

dual problem.

Algorithm 3: Offline Procedure for Coercive Problems

Choose $\Xi \subset \mathcal{D}$; // Super Sampling
Choose $N_{\max}$; // Maximum size of X_N
Choose δ_{tol} ; // Algorithm tolerance
Choose $\boldsymbol{\mu}_1 \in \mathcal{D}$; // Arbitrary chosen first parameter
Set $N \leftarrow 0$, $S_N \leftarrow \{\}$, ,$X_N \leftarrow \{\}$, $X_N^{du} \leftarrow \{\}$; // Initialization
repeat
  $N \leftarrow N + 1$;
  $S_N \leftarrow S_{N-1} \cup \boldsymbol{\mu}_N$;
  Solve $a(u, v; \boldsymbol{\mu}_N) = f(v; \boldsymbol{\mu}_N)$; // (5.5)
  $\xi_N \leftarrow$Gram-Schmidt$(\{\xi_1, ..., \xi_{N-1}, u(\boldsymbol{\mu}_N)\})$;
  $X_N \leftarrow X_{N-1} \oplus \text{span}\{\xi_N\}$;
  Solve dual problem $a(v, \psi; \boldsymbol{\mu}_N) = -l(v; \boldsymbol{\mu})$; // (5.17)
  $\xi_N^{du} \leftarrow$Gram-Schmidt$(\{\xi_1^{du}, ..., \xi_{N-1}^{du}, \psi(\boldsymbol{\mu}_N)\})$;
  $X_N^{du} \leftarrow X_{N-1}^{du} \oplus \text{span}\{\xi_N^{du}\}$;
  Precompute A_N^q and F_N^q ; // (5.15)
  Precompute $(\hat{f}_q, \hat{f}_{q'})$, $C_{qq',N}^{fa}$, $C_{qq',N}^{aa}$; // (43)
  for $\boldsymbol{\mu} \in \Xi$ **do**
    Compute error bound, $\Delta_N(\boldsymbol{\mu})$ or $\Delta_N^s(\boldsymbol{\mu})$;
  end
  $\boldsymbol{\mu}_{N+1} = \arg\max_{\boldsymbol{\mu} \in \Xi} \Delta_N(\boldsymbol{\mu})$ or $\boldsymbol{\mu}_{N+1} = \arg\max_{\boldsymbol{\mu} \in \Xi} \Delta_N^s(\boldsymbol{\mu})$;
until $\Delta_N^{(s)}(\boldsymbol{\mu}_{N+1}) < \delta_{tol}$ **or** $N \geq N_{max}$;

2 Reduced Basis for Saddle Point Problems

In this section, we present the RBM applied to saddle point problems, and we will especially focus on the Stokes equations as an illustration. The RBM is very well developed for coercive problems, but saddle point parametrized systems introduce new difficulties with the construction of stable reduced spaces and the development of efficient a posteriori error bounds[66, 90, 86] . The delicate problem of geometric parameters is particularly developed in [34] with a penalty method and later in [35, 87] with specific construction algorithm of the reduced space.

We propose a quick review of the reduced basis framework for saddle point problem. We present the problem formulation, the existing methods for the construction of a stable reduced space and the detailed error estimators. We provide more technical details about the integration of these features in the Feel++ framework in the implementation part III.

2.1 Problem formulation

Let V and Q two Hilbert spaces on a bounded domain Ω with inner products $(\cdot,\cdot)_V$, $(\cdot,\cdot)_Q$ and associated norms $||\cdot||_V = \sqrt{(\cdot,\cdot)_V}$, $||\cdot||_Q = \sqrt{(\cdot,\cdot)_Q}$ respectively. We also define the product space $Z = V \times Q$ with inner product $((\boldsymbol{u},p),(\boldsymbol{v},q))_Z = (\boldsymbol{u},\boldsymbol{v})_V + (p,q)_Q$ and norm $||\cdot||_Z = \sqrt{(\cdot,\cdot)_Z}$. We consider a parametrized saddle point problem: for any $\boldsymbol{\mu} \in \mathcal{D}$, find $\boldsymbol{u}(\boldsymbol{\mu}) \in V$ and $p(\boldsymbol{\mu}) \in Q$ such that

$$\begin{aligned} a(\boldsymbol{u}(\boldsymbol{\mu}),\boldsymbol{v};\boldsymbol{\mu}) + b(\boldsymbol{v},p(\boldsymbol{\mu});\boldsymbol{\mu}) &= f(\boldsymbol{v};\boldsymbol{\mu}), \qquad \forall \boldsymbol{v} \in V \\ b(\boldsymbol{u}(\boldsymbol{\mu}),q;\boldsymbol{\mu}) &= g(q;\boldsymbol{\mu}) \qquad \forall q \in Q \end{aligned} \tag{5.33}$$

where $a(\cdot,\cdot;\boldsymbol{\mu}) : V \times V \to \mathbb{R}$, $b(\cdot,\cdot;\boldsymbol{\mu}) : V \times Q \to \mathbb{R}$ and $f(\cdot;\boldsymbol{\mu}) : V \to \mathbb{R}$, $g(\cdot;\boldsymbol{\mu}) : Q \to \mathbb{R}$ are bilinear and linear bounded forms respectively, for any $\boldsymbol{\mu} \in \mathcal{D}$.

We furthermore assume that :

(a) $a(\cdot,\cdot;\boldsymbol{\mu})$ and $b(\cdot,\cdot;\boldsymbol{\mu})$ are continuous

$$\gamma_a(\boldsymbol{\mu}) = \operatorname*{supsup}_{\boldsymbol{u}\in V \boldsymbol{v}\in V} \frac{a(\boldsymbol{u},\boldsymbol{v};\boldsymbol{\mu})}{||\boldsymbol{u}||_V||\boldsymbol{v}||_V} < \infty, \quad \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{5.34}$$

$$\gamma_b(\boldsymbol{\mu}) = \operatorname*{supsup}_{\boldsymbol{v}\in V q\in Q} \frac{b(\boldsymbol{v},q;\boldsymbol{\mu})}{||\boldsymbol{v}||_V||q||_Q} < \infty, \quad \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{5.35}$$

(b) $a(\cdot,\cdot;\boldsymbol{\mu})$ is coercive on V

$$\alpha_a(\boldsymbol{\mu}) = \inf_{\boldsymbol{v}\in V} \frac{a(\boldsymbol{v},\boldsymbol{v};\boldsymbol{\mu})}{||\boldsymbol{v}||_V^2} > 0, \quad \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{5.36}$$

(c) $b(\cdot,\cdot;\boldsymbol{\mu})$ satisfies the inf-sup condition, this can be expressed either using the Brezzi constant,

$$\beta_{Br}(\boldsymbol{\mu}) = \inf_{q\in Q} \sup_{\boldsymbol{v}\in V} \frac{b(\boldsymbol{v},q;\boldsymbol{\mu})}{||\boldsymbol{v}||_V||q||_Q} > 0, \quad \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{5.37}$$

or the Babuška constant

$$\beta_{Ba}(\boldsymbol{\mu}) = \inf_{(\boldsymbol{u},p)\in Z} \sup_{(\boldsymbol{v},q)\in Z} \frac{a(\boldsymbol{u},\boldsymbol{v};\boldsymbol{\mu}) + b(\boldsymbol{v},p;\boldsymbol{\mu}) + b(\boldsymbol{u},q;\boldsymbol{\mu})}{||(\boldsymbol{u},p)||_Z||(\boldsymbol{v},q)||_Z} > 0, \quad \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{5.38}$$

(d) The bilinear forms $a(\cdot,\cdot;\boldsymbol{\mu})$, $b(\cdot,\cdot;\boldsymbol{\mu})$ and the linear forms $f(\cdot;\boldsymbol{\mu})$ and $g(\cdot;\boldsymbol{\mu})$ admit affine decompositions

$$a(\boldsymbol{u},\boldsymbol{v};\boldsymbol{\mu}) = \sum_{k=1}^{Q_a} \theta_k^a(\boldsymbol{\mu}) a_k(\boldsymbol{u},\boldsymbol{v}), \qquad f(\boldsymbol{v};\boldsymbol{\mu}) = \sum_{k=1}^{Q_f} \theta_k^f(\boldsymbol{\mu}) f_k(\boldsymbol{v}) \tag{5.39}$$

$$b(\boldsymbol{u},q;\boldsymbol{\mu}) = \sum_{k=1}^{Q_b} \theta_k^b(\boldsymbol{\mu}) b_k(\boldsymbol{u},q), \qquad g(q;\boldsymbol{\mu}) = \sum_{k=1}^{Q_g} \theta_k^g(\boldsymbol{\mu}) g_k(q) \tag{5.40}$$

Truth Approximation

As truth approximation of the continuous problem, we use the finite element solution $(\boldsymbol{u}_h, p_h)$ of (5.33) on a regular triangulation $\mathcal{T}_h$ of the domain Ω. To that end, we introduce finite element function spaces V_h, Q_h and Z_h and their respective scalar products and norms. The coercivity and continuity properties of the forms are naturally inherited from the continuous formulation. The spaces V_h and Q_h are supposed to be properly defined to satisfy the Ladyzhenskaya–Babuška–Brezzi condition. More details about the finite element resolution are given in section 2.

Offline Procedure : Construction of the RB Space

The classical procedure for the construction of the RB space, as seen in section1.1, is not adapted for saddle point problems. This procedure would build the RB spaces V_N^0 ans Q_N as spans of the truth solutions $\{\boldsymbol{u}(\boldsymbol{\mu}_n)\}$ and $\{p(\boldsymbol{\mu}_n)\}$, $1 \leq n \leq N$, respectively,

$$V_N^0 = \text{span}\{\boldsymbol{u}(\boldsymbol{\mu}_n), 1 \leq n \leq N\} = \text{span}\{\boldsymbol{\zeta}_n, 1 \leq n \leq N\}, \tag{5.41}$$
$$Q_N = \text{span}\{p(\boldsymbol{\mu}_n), 1 \leq n \leq N\} = \text{span}\{\eta_n, 1 \leq n \leq N\}. \tag{5.42}$$

With such RB spaces, the online saddle-point system would not be well-posed, since it does not satisfy the inf-sup condition, see [5, 6],

$$\beta_N(\boldsymbol{\mu}) = \inf_{q_N \in Q_N} \sup_{\boldsymbol{v}_N \in V_N} \frac{b(\boldsymbol{v}_N, q_N; \boldsymbol{\mu})}{||q_N||_Q ||\boldsymbol{v}_N||_V} > 0. \tag{5.43}$$

It is proved in [85, 90] that the space $V_N^0 = \text{span}\{\boldsymbol{u}(\boldsymbol{\mu}_n), 1 \leq n \leq N\}$ can be enriched in order to obtain a stable pair (V_N^1, Q_N) in the sense of the inf-sup condition. This space V_N^1 is built by adding the supremizer functions $T^k \eta_n$ for any $1 \leq k \leq Q_b$, $1 \leq n \leq N$.

$$V_N^1 = V_N^0 \oplus \text{span}\{T^k \eta_n, 1 \leq k \leq Q_b, 1 \leq n \leq N\}. \tag{5.44}$$

The supremizer function $T^k q$ is defined for any $q \in Q$ and any $1 \leq k \leq Q_b$ as

$$(T^k q, \boldsymbol{v}_h)_V = b^k(\boldsymbol{v}_h, q), \quad \forall \boldsymbol{v}_h \in V_h. \tag{5.45}$$

In practice, it is widespread not to build the RB space V_N^1 which might become very large, depending on Q_b. The usual alternative is to add the *global* supremizer $T^{\boldsymbol{\mu}_n} \eta_n$, for any parameters $\boldsymbol{\mu}_n \in S_N$,

$$V_N = V_N^0 \oplus \text{span}\{T^{\boldsymbol{\mu}_n} \eta_n, 1 \leq n \leq N\}, \tag{5.46}$$

where $T^{\boldsymbol{\mu}} q$ is defined, for any $\boldsymbol{\mu} \in \mathcal{D}$ and any $q \in Q$ as

$$(T^{\boldsymbol{\mu}} q, \boldsymbol{v}_h)_V = b(\boldsymbol{v}_h, q; \boldsymbol{\mu}), \quad \forall \boldsymbol{v}_h \in V_h. \tag{5.47}$$

With this definition of V_N it is no longer possible to prove the stability of the pair (V_N, Q_N) *a priori*. Nevertheless, different numerical tests conclude in the stability of this option, see [34, 35].

The basis $\{\eta_n, 1 \leq n \leq N\}$ of the RB space Q_N is obtained, just like in the coercive case, by an orthonormalization of the snapshots $p(\boldsymbol{\mu}_n), 1 \leq n \leq N$. We apply the same process to the RB space V_N to recover an orthonormal basis $\{\boldsymbol{\zeta}_n, 1 \leq n \leq 2N\}$. The complete offline procedure implemented for saddle point problems is summarized in algorithm 4. The nature of the error indicator $\Delta_N(\boldsymbol{\mu})$ used in the greedy selection is discussed in section 2.2.

Algorithm 4: Offline Procedure for Saddle Point Problems

Choose $\Xi \subset \mathcal{D}$; // Super Sampling
Choose $N_{\max}$; // Maximum size of X_N
Choose δ_{tol} ; // Algorithm tolerance
Set $N \leftarrow 0$, $S_N \leftarrow \{\}$, ,$V_N \leftarrow \{\}$, $Q_N \leftarrow \{\}$; // Initialization
repeat
  $N \leftarrow N + 1$;
  $S_N \leftarrow S_{N-1} \cup \{\boldsymbol{\mu}_N\}$;
  Find $(\boldsymbol{u}(\boldsymbol{\mu}), p(\boldsymbol{\mu}))$ solution of system (5.33);
  $\eta_N \leftarrow$ Gram-Schmidt$(\{\eta_1, ..., \eta_{N-1}, p(\boldsymbol{\mu}_N)\})$;
  Compute $T^{\boldsymbol{\mu}_N}\eta_N$; // (5.47)
  $\boldsymbol{\zeta}_{2N-1}, \boldsymbol{\zeta}_{2N} \leftarrow$ Gram-Schmidt$(\{\boldsymbol{\zeta}_1, ..., \boldsymbol{\zeta}_{2N-2}, \boldsymbol{u}(\boldsymbol{\mu}_N), T^{\boldsymbol{\mu}_N}\eta_N\})$;
  $V_N \leftarrow V_{N-1} \oplus \text{span}\{\boldsymbol{\zeta}_{2N-1}, \boldsymbol{\zeta}_{2N}\}$;
  $Q_N \leftarrow Q_{N-1} \oplus \text{span}\{\eta_N\}$;
  Precompute online structures;
  for $\boldsymbol{\mu} \in \Xi$ **do**
    Compute error bound, $\Delta_N(\boldsymbol{\mu})$;
  end
  $\boldsymbol{\mu}_{N+1} = \arg\max_{\boldsymbol{\mu} \in \Xi} \Delta_N(\boldsymbol{\mu})$;
until $\Delta_N(\boldsymbol{\mu}_{N+1}) < \delta_{tol}$ **or** $N \geq N_{max}$;

Advanced algorithms exist in the literature to improve the convergence of the method. For instance, [35] and [87] propose specific enrichment of the RB space V_N to provide better error bounds. Unfortunately, we had no time to implement these methods properly, and we refer the readers to the cited references for more details.

Online Evaluation : computation of $(\boldsymbol{u}_N(\boldsymbol{\mu}), p_N(\boldsymbol{\mu}))$

As for the coercive problem, we use a Galerkin projection of on the RB space $Z_N = V_N \times Q_N$. It implies the resolution of the linear system

$$\begin{bmatrix} \boldsymbol{A}_N(\boldsymbol{\mu}) & \boldsymbol{B}_N^{\intercal}(\boldsymbol{\mu}) \\ \boldsymbol{B}_N(\boldsymbol{\mu}) & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_N(\boldsymbol{\mu}) \\ p_N(\boldsymbol{\mu}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_N(\boldsymbol{\mu}) \\ \boldsymbol{G}_N(\boldsymbol{\mu}) \end{bmatrix} \tag{5.48}$$

where

$$\begin{aligned}(\boldsymbol{A}_N(\boldsymbol{\mu}))_{i,j} &= a(\boldsymbol{\zeta}_j, \boldsymbol{\zeta}_i; \boldsymbol{\mu}), \quad (\boldsymbol{F}(\boldsymbol{\mu}))_i = f(\eta_i; \boldsymbol{\mu}), \quad 1 \leq i, j \leq 2N \\ (\boldsymbol{B}_N(\boldsymbol{\mu}))_{l,j} &= b(\boldsymbol{\zeta}_j, \eta_l; \boldsymbol{\mu}), \quad (\boldsymbol{G}_N(\boldsymbol{\mu}))_l = g(\eta_l; \boldsymbol{\mu}), \quad 1 \leq l \leq N.\end{aligned} \tag{5.49}$$

Here again the affine assumption (5.39) is crucial in order to precompute these matrices and vectors.

$$\begin{aligned}(\boldsymbol{A}_N(\boldsymbol{\mu}))_{i,j} &= \sum_{k=1}^{Q_a} \theta_k^a(\boldsymbol{\mu}) \underbrace{a_k(\boldsymbol{\zeta}_j, \boldsymbol{\zeta}_i)}_{\text{precomputed}}, \quad (\boldsymbol{F}(\boldsymbol{\mu}))_i = \sum_{k=1}^{Q_f} \theta_k^f \underbrace{f_k(\boldsymbol{\zeta}_i)}_{\text{precomputed}}, \quad 1 \leq i, j \leq 2N \\ (\boldsymbol{B}_N(\boldsymbol{\mu}))_{l,j} &= \sum_{k=1}^{Q_b} \theta_k^b \underbrace{b_k(\boldsymbol{\zeta}_j, \eta_l)}_{\text{precomputed}}, \quad (\boldsymbol{G}_N(\boldsymbol{\mu}))_l = \sum_{k=1}^{Q_g} \theta_k^g \underbrace{g_k(\eta_l)}_{\text{precomputed}}, \quad 1 \leq l \leq N.\end{aligned} \tag{5.50}$$

These quantities are evaluated once during the offline procedure and then stored for online phases. During the online phase, the assembly of the linear system (5.49) only requires the additions of matrices (resp. vectors) of size $\mathcal{O}(N^2)$ (resp. $\mathcal{O}(N)$).

2.2 Error Estimation

Unfortunately, we did not have enough time to implement error estimators for the saddle-point problems: the computation of the dual norm of the residual is working, but we had no time to adapt the SCM algorithm for saddle-point problems. In the further numerical results, we will use the dual norm of the residual as an error indicator. Nevertheless, we remind in this section the principal error estimators found in the literature [21, 26, 80, 87].

We assume that the RB spaces V_N and Q_N are built and so we are able to evaluate a reduced basis solution $(\boldsymbol{u}_N(\boldsymbol{\mu}), p_N(\boldsymbol{\mu}))$ for any $\boldsymbol{\mu} \in \mathcal{D}$. We are also interested in giving a sharp and efficient bound to the different errors, with respect to the truth model,

$$e_N^{\boldsymbol{u}}(\boldsymbol{\mu}) = \boldsymbol{u}_h(\boldsymbol{\mu}) - \boldsymbol{u}_N(\boldsymbol{\mu}) \in V, \tag{5.51}$$

$$e_N^p(\boldsymbol{\mu}) = p_h(\boldsymbol{\mu}) - p_N(\boldsymbol{\mu}) \in Q, \tag{5.52}$$

$$e_N(\boldsymbol{\mu}) = (e_N^{\boldsymbol{u}}(\boldsymbol{\mu}), e_N^p(\boldsymbol{\mu})) \in Z. \tag{5.53}$$

We also may be interested in the evaluation of the error for a linear output. We assume here that the output $s(\boldsymbol{\mu})$ can be express as

$$s(\boldsymbol{\mu}) = l((\boldsymbol{u}(\boldsymbol{\mu}), p(\boldsymbol{\mu})); \boldsymbol{\mu}) = l^{\boldsymbol{u}}(\boldsymbol{u}(\boldsymbol{\mu}); \boldsymbol{\mu}) + l^p(p(\boldsymbol{\mu}); \boldsymbol{\mu}) \tag{5.54}$$

where $l : Z \to \mathbb{R}$, $l^{\boldsymbol{u}} : V \to \mathbb{R}$ and $l^p : Q \to \mathbb{R}$ are linear forms which admit affine decompositions. In the particular case where $l^{\boldsymbol{u}} = f$ and $l^p = g$ the problem is compliant. We denote by $e_N^s(\boldsymbol{\mu})$ the error on the output $s(\boldsymbol{\mu})$

$$e_N^s(\boldsymbol{\mu}) = s_h(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu}). \tag{5.55}$$

For the construction of the different error bounds, we may need the following ingredients:

- Online-efficient upper and lower bounds for the continuity and coercivity constants introduced in (5.34)-(5.36),

$$\gamma_a^{LB}(\boldsymbol{\mu}) \leq \gamma_a(\boldsymbol{\mu}) \leq \gamma_a^{UB}(\boldsymbol{\mu}), \tag{5.56}$$
$$\alpha_a^{LB}(\boldsymbol{\mu}) \leq \alpha_a(\boldsymbol{\mu}) \leq \alpha_a^{UB}(\boldsymbol{\mu}). \tag{5.57}$$

  Those bounds can be computed using the SCM, as mentioned in section 1.2.

- Online-efficient upper and lower bounds for the inf-sup constant, either the Brezzi (5.37) or the Babuška (5.38) version,

$$\beta_{Br}^{LB}(\boldsymbol{\mu}) \leq \beta_{Br}(\boldsymbol{\mu}) \leq \beta_{Br}^{UB}(\boldsymbol{\mu}), \tag{5.58}$$
$$\beta_{Ba}^{LB}(\boldsymbol{\mu}) \leq \beta_{Ba}(\boldsymbol{\mu}) \leq \beta_{Ba}^{UB}(\boldsymbol{\mu}). \tag{5.59}$$

  These bounds can also be computed using the SCM.

- The dual norms of the residuals associated with the RB approximation

$$||r_N^{\boldsymbol{u}}(\cdot;\boldsymbol{\mu})||_{V'} = \sup_{\boldsymbol{v}\in V} \frac{r_N^{\boldsymbol{u}}(\boldsymbol{v};\boldsymbol{\mu})}{||\boldsymbol{v}||_V}, \tag{5.60}$$
$$||r_N^{p}(\cdot;\boldsymbol{\mu})||_{Q'} = \sup_{q\in Q} \frac{r_N^{p}(q;\boldsymbol{\mu})}{||q||_Q}, \tag{5.61}$$

  where, $\forall \boldsymbol{\mu} \in \mathcal{D}$,

$$r_N^{\boldsymbol{u}}(\boldsymbol{v};\boldsymbol{\mu}) = f(\boldsymbol{v};\boldsymbol{\mu}) - a(\boldsymbol{u}_N(\boldsymbol{\mu}),\boldsymbol{v};\boldsymbol{\mu}) - b(\boldsymbol{v},p_N(\boldsymbol{\mu});\boldsymbol{\mu}), \tag{5.62}$$
$$r_N^{p}(q;\boldsymbol{\mu}) = g(q;\boldsymbol{\mu}) - b(\boldsymbol{u}_N(\boldsymbol{\mu}),q;\boldsymbol{\mu}). \tag{5.63}$$

  We also define the total residual $r_N((\boldsymbol{v},q);\boldsymbol{\mu})$ as

$$r_N((\boldsymbol{v},q);\boldsymbol{\mu}) = r_N^{\boldsymbol{u}}(\boldsymbol{v};\boldsymbol{\mu}) + r_N^{p}(q;\boldsymbol{\mu}), \quad \forall \boldsymbol{v}\in V,\ \forall q\in Q, \tag{5.64}$$

  and its dual norm as

$$\|r_N(\cdot;\boldsymbol{\mu})\|_{Z'} = \sup_{(\boldsymbol{v},q)\in Z} \frac{r_N((\boldsymbol{v},q);\boldsymbol{\mu})}{\|(\boldsymbol{v},q)\|_Z} = \sqrt{\|r_N^{\boldsymbol{u}}(\cdot;\boldsymbol{\mu})\|_{V'}^2 + \|r_N^{p}(\cdot;\boldsymbol{\mu})\|_{Q'}^2}. \tag{5.65}$$

  The efficient evaluation of this dual norms through an offline-online decomposition is detailed in appendix 3.

Error Bounds for the Solutions

A first error bound $\Delta_N(\boldsymbol{\mu})$ for Saddle Point problems, and especially Stokes equations, uses the Babuška inf-sup constant in order to bound the global error of the solutions $e_N(\boldsymbol{\mu})$, with respect to the truth model,

$$||e_N(\boldsymbol{\mu})||_Z \leq \Delta_N(\boldsymbol{\mu}) := \frac{||r_N(\cdot;\boldsymbol{\mu})||_{Z'}}{\beta_{Ba}^{LB}(\boldsymbol{\mu})}, \quad \forall \boldsymbol{\mu}\in\mathcal{D}. \tag{5.66}$$

This error bound is a direct extension of the error estimation for coercive problems.

Another bound is proposed in [35, 34], in order to measure separately the errors on the velocity field $e_N^{\boldsymbol{u}}(\boldsymbol{\mu})$ and the pressure field $e_N^p(\boldsymbol{\mu})$. This formulation requires more elements and uses the Brezzi inf-sup constant to define two bounds $\Delta_N^{\boldsymbol{u}}(\boldsymbol{\mu})$ and $\Delta_N^p(\boldsymbol{\mu})$, for any $\boldsymbol{\mu} \in \mathcal{D}$,

$$||e_N^{\boldsymbol{u}}(\boldsymbol{\mu})||_V \leq \Delta_N^{\boldsymbol{u}}(\boldsymbol{\mu}) := \frac{||r_N^{\boldsymbol{u}}(\cdot;\boldsymbol{\mu})||_{V'}}{\alpha_a^{LB}(\boldsymbol{\mu})} + \left(1 + \frac{\gamma_a^{UB}(\boldsymbol{\mu})}{\alpha_a^{LB}(\boldsymbol{\mu})}\right) \frac{||r^p(\cdot;\boldsymbol{\mu})||_{Q'}}{\beta_{Br}^{LB}(\boldsymbol{\mu})}, \tag{5.67}$$

$$||e_N^p(\boldsymbol{\mu})||_{Q'} \leq \Delta_N^p(\boldsymbol{\mu}) := \frac{||r_N^{\boldsymbol{u}}(\cdot;\boldsymbol{\mu})||_{V'}}{\beta_{Br}^{LB}} + \frac{\gamma_a^{UB}(\boldsymbol{\mu})}{\beta_{Br}^{LB}(\boldsymbol{\mu})} \Delta_N^{\boldsymbol{u}}(\boldsymbol{\mu}). \tag{5.68}$$

From these two bounds, we can then define a new global error estimator for the solution $(\boldsymbol{u}_N(\boldsymbol{\mu}), p_N(\boldsymbol{\mu}))$ as

$$||e_N(\boldsymbol{\mu})||_Z \leq \Delta_N^{Br}(\boldsymbol{\mu}) := \sqrt{(\Delta_N^{\boldsymbol{u}}(\boldsymbol{\mu}))^2 + (\Delta_N^p(\boldsymbol{\mu}))^2}, \quad \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{5.69}$$

For a sharpness study and a comparison of these different error bounds, we refer the readers to the cited references.

Error Bound for Linear Output

In the convenient case of compliant problems, as in the coercive case, we can directly evaluate the error bound on the error $e_N^s(\boldsymbol{\mu})$[87],

$$|e_N^s(\boldsymbol{\mu})| \leq \Delta_N^{s,c}(\boldsymbol{\mu}) := 2\frac{||r_n(\cdot;\boldsymbol{\mu})||_{Z'}^2}{\beta_{Ba}^{LB}(\boldsymbol{\mu})}, \tag{5.70}$$

this formulation is again a direct generalization of the coercive framework.

In the more general case of non-compliant problems, we introduce a dual problem, as find $(\boldsymbol{\psi}(\boldsymbol{\mu}), \lambda(\boldsymbol{\mu})) \in V \times Q$ such that

$$\begin{cases} a(\boldsymbol{v}, \boldsymbol{\psi}(\boldsymbol{\mu}); \boldsymbol{\mu}) + b(\lambda(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}) = -l^{\boldsymbol{u}}(\boldsymbol{v}; \boldsymbol{\mu}) & \forall \boldsymbol{v} \in V \\ b(q, \boldsymbol{\psi}(\boldsymbol{\mu}); \boldsymbol{\mu}) = -l^p(q; \boldsymbol{\mu}), & \forall q \in Q \end{cases} \tag{5.71}$$

where $l^{\boldsymbol{u}}$ and l^p are defined in (5.54). We denote by $(\boldsymbol{\psi}_h(\boldsymbol{\mu}), \lambda_h(\boldsymbol{\mu}))$ the discrete solution of system (5.71) obtained using our truth model, with a discretization and a space definition adapted in order to satisfy the inf-sup condition. We denote by V_n^{du} and Q_N^{du} the dual RB spaces. We assume that these RB spaces are built during the greedy algorithm in a way similar to the coercive case. We can then define the dual residuals,

$$\begin{aligned} r_N^{\boldsymbol{u},du}(\boldsymbol{v};\boldsymbol{\mu}) &= l^{\boldsymbol{u}}(\boldsymbol{v};\boldsymbol{\mu}) + a(\boldsymbol{v}, \boldsymbol{\psi}_N(\boldsymbol{\mu});\boldsymbol{\mu}) + b(\lambda_N(\boldsymbol{\mu}), \boldsymbol{v};\boldsymbol{\mu}), \quad \forall \boldsymbol{v} \in V, \\ r_N^{p,du}(q;\boldsymbol{\mu}) &= l^p(q;\boldsymbol{\mu}) + b(q, \boldsymbol{\psi}_N(\boldsymbol{\mu});\boldsymbol{\mu}), \quad \forall q \in Q, \\ r_N^{du}(\boldsymbol{v}, q;\boldsymbol{\mu}) &= r_N^{\boldsymbol{u},du}(\boldsymbol{v};\boldsymbol{\mu}) + r_N^{p,du}(q;\boldsymbol{\mu}), \end{aligned} \tag{5.72}$$

and the associated dual norms

$$
\begin{aligned}
||r_N^{\boldsymbol{u},du}(\cdot;\boldsymbol{\mu})||_{V'} &= \sup_{\boldsymbol{v}\in V}\frac{r_N^{\boldsymbol{u},du}(\boldsymbol{v};\boldsymbol{\mu})}{||\boldsymbol{v}||_V} \\
||r_N^{p,du}(\cdot;\boldsymbol{\mu})||_{Q'} &= \sup_{q\in Q}\frac{r_N^{p,du}(q;\boldsymbol{\mu})}{||q||_Q} \\
||r_N^{du}(\cdot;\boldsymbol{\mu})||_{Z'} &= \sup_{(\boldsymbol{v},q)\in Z}\frac{r_N^{du}((\boldsymbol{v},q);\boldsymbol{\mu})}{||(\boldsymbol{v},q)||_Z} = \sqrt{||r_N^{\boldsymbol{u},du}(\cdot;\boldsymbol{\mu})||_{V'}^2 + ||r_N^{p,du}(\cdot;\boldsymbol{\mu})||_{Q'}^2}.
\end{aligned}
\tag{5.73}
$$

The error bound for non-compliant linear output is then defined as

$$
|e_N^s(\boldsymbol{\mu})| \leq \Delta_N^s(\boldsymbol{\mu}) := 2\frac{||r_N^{du}(\boldsymbol{\mu})||_{Z'}^{1/2}||r_N(\boldsymbol{\mu})||_{Z'}^{1/2}}{\beta_{Ba}^{LB}(\boldsymbol{\mu})} \tag{5.74}
$$

The evaluation of the dual norms of the residuals associated with each RB space is detailed in appendix 3.2.

3 Numerical Applications : RB for Stokes Problem

To validate our implementation of the RB framework for Saddle Point problems, we present, in this section, our results on a test case proposed in [35].

3.1 Problem Formulation

We consider a Stokes flow in a 2D channel with an obstacle, see figure 5.1. The channel is a rectangle with fixed dimension (4×1). The obstacle is rectangular with variable width (μ_1) and height (μ_2). The parameter $\boldsymbol{\mu} = (\mu_1, \mu_2)$ is bounded such that $\boldsymbol{\mu} \in \mathcal{D} = [0.1, 0.5] \times [0.1, 0.5]$. Thus the parametrized domain is given by $\tilde{\Omega}(\boldsymbol{\mu}) = [0,4] \times [0,1] \backslash \tilde{\mathcal{O}}(\boldsymbol{\mu})$ where $\tilde{\mathcal{O}}(\boldsymbol{\mu}) =]2 - \mu_1/2, 2 + \mu_1/2[\times]0, \mu_2[$ is the obstacle. We denote by $\tilde{\Gamma}(\boldsymbol{\mu})$ the boundary of $\tilde{\Omega}(\boldsymbol{\mu})$, $\tilde{\Gamma}_{in}$ the inflow boundary, $\tilde{\Gamma}_{out}$ the outflow boundary and by $\tilde{\Gamma}_0(\mu) = \tilde{\Gamma}(\mu) \backslash (\tilde{\Gamma}_{in} \cup \tilde{\Gamma}_{out})$ the boundary on where we assume a no-slip condition. Here the $\tilde{}$ symbol refers to the parametrized domain in opposition to the reference domain Ω.

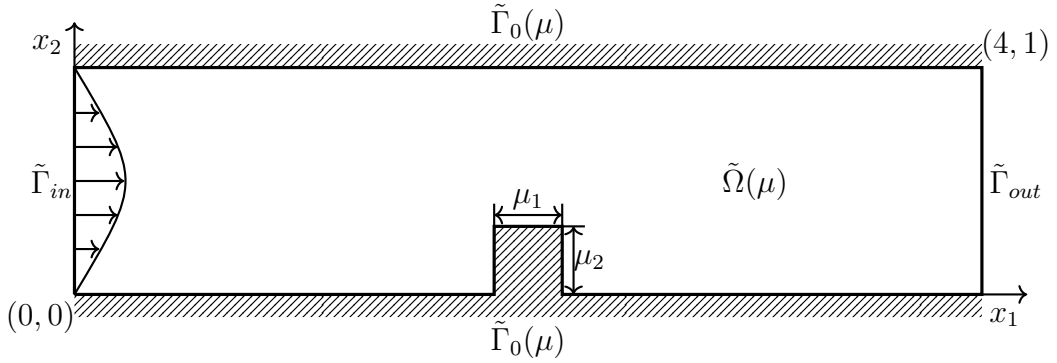


Figure 5.1 – 2D Geometry of the Problem

Formulation on a Parametrized Domain $\tilde{\Omega}(\boldsymbol{\mu})$

The Stokes equations for the velocity $\tilde{\boldsymbol{u}}(\mu) : \tilde{\Omega}(\mu) \to \mathbb{R}^2$ and the presure $\tilde{p}(\mu) : \tilde{\Omega}(\mu) \to \mathbb{R}$ are given by

$$\begin{cases} \tilde{\nabla}\tilde{p}(\mu) - \nu\tilde{\Delta}\tilde{\boldsymbol{u}}(\mu) = 0, & \text{in } \tilde{\Omega}(\mu) \\ \tilde{\nabla} \cdot \boldsymbol{u}(\mu) = 0, & \text{in } \tilde{\Omega}(\mu), \\ \tilde{\boldsymbol{u}}(\mu) = 0, & \text{on } \tilde{\Gamma}_0(\mu), \\ (\nu\tilde{\nabla}\tilde{\boldsymbol{u}}(\mu) - \mathbb{I}\tilde{p}(\mu)) \cdot \tilde{\boldsymbol{n}} = 0, & \text{on } \tilde{\Gamma}_{out}, \\ \tilde{\boldsymbol{u}} = \boldsymbol{\phi}, & \text{on } \Gamma_{in}, \end{cases} \tag{5.75}$$

with $\boldsymbol{\phi} = \begin{pmatrix} 4\tilde{x}_2(1-\tilde{x}_2) \\ 0 \end{pmatrix}$ and $\nu = 1$ the fluid viscosity.

We introduce the finite element space $\tilde{Z}_h(\boldsymbol{\mu}) = \tilde{V}_h(\boldsymbol{\mu}) \times \tilde{Q}_h(\boldsymbol{\mu}) = \mathbb{TH}^k_{ch}(\tilde{\Omega}(\boldsymbol{\mu}))$ and the usual associated scalar products and norms. The weak formulation on the parametrized domain $\tilde{\Omega}(\boldsymbol{\mu})$ can be written as: find $(\tilde{\boldsymbol{u}}_h, \tilde{p_h}) \in \tilde{Z}_h(\boldsymbol{\mu})$ such that

$$\tilde{a}(\tilde{\boldsymbol{u}}_h, \tilde{\boldsymbol{v}}_h) + \tilde{b}(\tilde{\boldsymbol{v}}_h, \tilde{p_h}) = \tilde{f}(\tilde{\boldsymbol{v}}_h), \qquad \forall \boldsymbol{v}_h \in \tilde{V}_h(\mu), \tag{5.76}$$

$$\tilde{b}(\tilde{\boldsymbol{u}}_h, \tilde{q_h}) = \tilde{g}(\tilde{q_h}), \qquad \forall q_h \in \tilde{Q}_h(\mu), \tag{5.77}$$

where

$$\tilde{a}(\boldsymbol{u}, \boldsymbol{v}) = \int_{\tilde{\Omega}(\mu)} \nu\tilde{\nabla}\boldsymbol{u} : \tilde{\nabla}\boldsymbol{v} \, d\tilde{\Omega}, \qquad \tilde{f}(\boldsymbol{v}) = 0, \tag{5.78}$$

$$\tilde{b}(\tilde{\boldsymbol{u}}, \tilde{q}) = -\int_{\tilde{\Omega}(\mu)} q(\tilde{\nabla} \cdot \boldsymbol{u}) \, d\tilde{\Omega}, \qquad \tilde{g}(q_h) = 0. \tag{5.79}$$

Formulation on the Reference Domain Ω

In order to apply the RBM to this problem, we perform the simulation on a reference domain $\Omega = \tilde{\Omega}(\boldsymbol{\mu}_r)$. We denote by $Z_h = V_h \times Q_h = \mathbb{TH}^k_{ch}(\Omega)$ the FE space on

the reference domain Ω. We will use a mapping from $\tilde{\Omega}(\boldsymbol{\mu})$ to Ω in order to solve the problem (5.76) in the space Z_h.

We first introduce $\forall \boldsymbol{\mu} \in \mathcal{D}$ a domain decomposition of $\tilde{\Omega}(\boldsymbol{\mu})$,

$$\overline{\tilde{\Omega}(\mu)} = \bigcup_{k=1}^{P} \overline{\tilde{\Omega}^k(\mu)}, \quad \text{s.t.} \quad \tilde{\Omega}^k(\mu) \cap \tilde{\Omega}^l(\mu) = \emptyset, \quad \forall 1 \leq k < l \leq P. \tag{5.80}$$

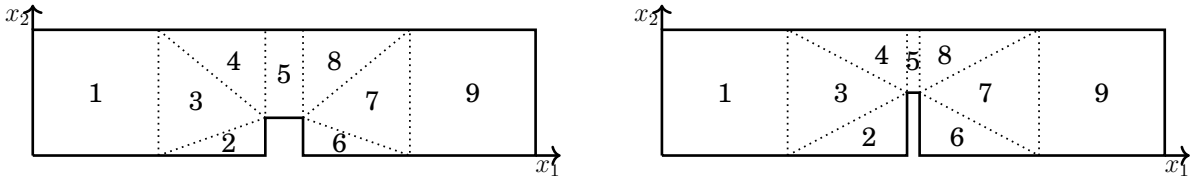
This construction is also naturally verified for the reference domain Ω,

$$\overline{\Omega} = \bigcup_{k=1}^{P} \overline{\Omega^k}, \quad \text{s.t.} \quad \Omega^k \cap \Omega^l = \emptyset, \quad 1 \leq k < l \leq P. \tag{5.81}$$

We now define P continuous and bijective affine mappings $\mathcal{G}^k(.;\boldsymbol{\mu})$ such that $\forall \boldsymbol{\mu} \in \mathcal{D}$, $\mathcal{G}^k(\tilde{\Omega}^k(\boldsymbol{\mu})) = \Omega^k$, $\forall 1 \leq k \leq P$ and $\mathcal{G}^k(\tilde{\boldsymbol{x}};\boldsymbol{\mu}) = \mathcal{G}^l(\tilde{\boldsymbol{x}};\boldsymbol{\mu})$, $\forall \tilde{\boldsymbol{x}} \in \overline{\tilde{\Omega}^k(\boldsymbol{\mu})} \cap \overline{\tilde{\Omega}^l(\boldsymbol{\mu})}$, $\forall 1 \leq k < l \leq P$,

$$\mathcal{G}^k(\tilde{\boldsymbol{x}};\boldsymbol{\mu}) = \boldsymbol{G}^k(\boldsymbol{\mu})\tilde{\boldsymbol{x}} + \boldsymbol{H}^k(\boldsymbol{\mu}), \quad 1 \leq k \leq P. \tag{5.82}$$

We propose such a decomposition for the domain $\Omega = \tilde{\Omega}(\boldsymbol{\mu}_r)$ in figure 5.2. Note that the design of this decomposition will definitely change the affine decomposition, but not the number of terms (as long as the decomposition is adapted to the geometry and the parametrization).



(a) $\mu = (\mu_r, \mu_r)$ (b) $\mu = (0.1, 0.5)$

Figure 5.2 – Domain Decomposition used for the mapping

Using the usual composition rules for differentiation operators, we have the relations,

$$\begin{aligned} &\partial/\partial x_i = \boldsymbol{G}^k_{ij} \partial/\partial x_j \\ &d\tilde{\Omega}(\mu) = J^k(\mu) d\Omega, \quad J^k(\mu) = |\det(\boldsymbol{G}^k(\mu))^{-1}|. \end{aligned} \tag{5.83}$$

We can then rewrite the weak formulation (5.76) on the reference domain Ω as find $(\boldsymbol{u}_h(\boldsymbol{\mu}), p_h(\boldsymbol{\mu})) \in Z_h$ such that

$$a(\boldsymbol{u}_h, \boldsymbol{v}_h) + b(\boldsymbol{v}_h, p_h) = f(\boldsymbol{v}_h), \qquad \forall \boldsymbol{v}_h \in V_h, \tag{5.84}$$

$$b(\boldsymbol{u}_h, q_h) = g(q_h), \qquad \forall q_h \in Q_h, \tag{5.85}$$

with

$$a(\boldsymbol{u}, \boldsymbol{v}; \mu) = \sum_{k=1}^{P} G^k_{mj}(\mu) G^k_{lj} J^k(\mu) \int_{\Omega^k} \frac{\partial u_i}{\partial x_m} \frac{\partial v_i}{\partial x_l} \, d\Omega, \qquad f(\boldsymbol{v}) = 0 \tag{5.86}$$

$$b(\boldsymbol{u}, q) = \sum_{k=1}^{P} -G_{ji}^{k} J^{k}(\mu) \int_{\Omega^k} \frac{\partial u_i}{\partial x_j} q \, d\Omega, \qquad g(q) = 0. \tag{5.87}$$

The detailed affine decomposition is presented in appendix 4.

3.2 Numerical Results

We use a FE space $Z_h = V_h \times Q_h = \mathbb{TH}^1_{ch}$. The discretization leads to $\mathcal{N} = 66857$ velocity and pressure degrees of freedom The mesh of the reference domain is presented in figure 5.3. We illustrate the results with some velocity and pressure profiles in figure 5.4. These solutions were computed on the reference domain and then warped onto the real domain for visualization.

The supersampling Ξ is an equidistributed grid of $\mathcal{D}$ with 1000 points. We are interested in this case in the convergence of the errors

$$\begin{aligned} \|e_N^{\boldsymbol{u}}(\boldsymbol{\mu})\|_2 &= \|\boldsymbol{u}_h(\boldsymbol{\mu}) - \boldsymbol{u}_N(\boldsymbol{\mu})\|_2, \\ \|e_N^{p}(\boldsymbol{\mu})\|_2 &= \|p_h(\boldsymbol{\mu}) - p_N(\boldsymbol{\mu})\|_2, \\ \|e_N(\boldsymbol{\mu})\|_2 &= \sqrt{\|e_N^{\boldsymbol{u}}(\boldsymbol{\mu})\|_2^2 + \|e_N^{p}(\boldsymbol{\mu})\|_2^2}. \end{aligned} \tag{5.88}$$

We computed these quantities for 100 different values of $\boldsymbol{\mu}$. The figure 5.5 presents the evolution of the maximum, minimum and mean value of these errors, for $N \in [\![1, 50]\!]$. The reduced basis was built with 50 different parameters in S_N, using the dual norm of the residual as an error indicator in the greedy algorithm 4. We clearly observe the convergence of the method. We also notice an unexpected behavior with the error growing again near the value $N = 20$. We found no explanation for this "jump" for now, however these results validate our implementation of the method.

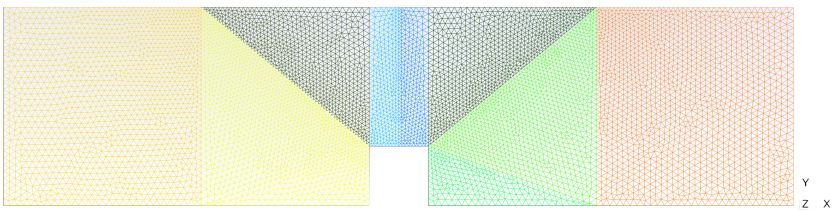
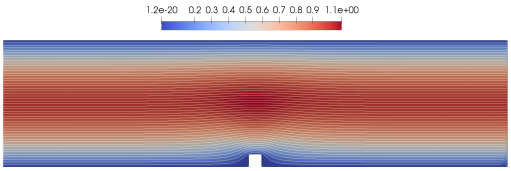
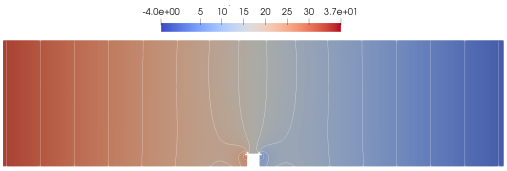


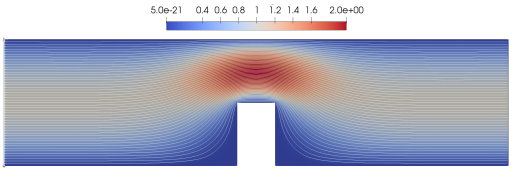
Figure 5.3 – Mesh Used for the FE simulation.



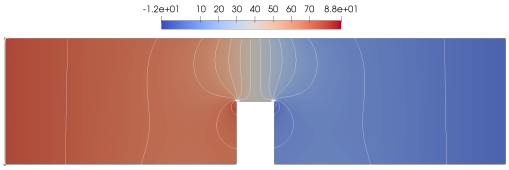
(a) Velocity Profile with Streamlines, $\boldsymbol{\mu} = (0.1, 0.1)$



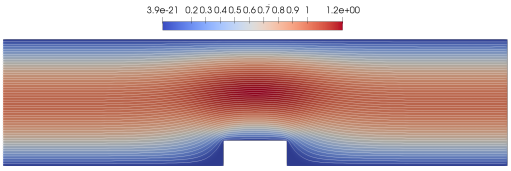
(b) Pressure Profile with Contours, $\boldsymbol{\mu} = (0.1, 0.1)$



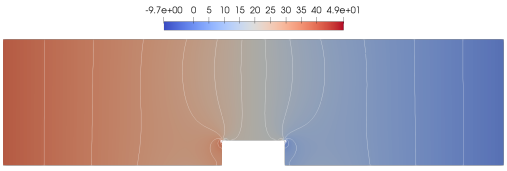
(c) Velocity Profile with Streamlines, $\boldsymbol{\mu} = (0.3, 0.5)$



(d) Pressure Profile with Contours, $\boldsymbol{\mu} = (0.3, 0.5)$

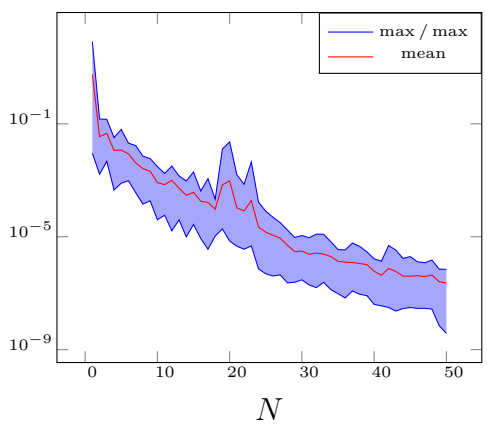


(e) Velocity Profile with Streamlines, $\boldsymbol{\mu} = (0.5, 0.2)$

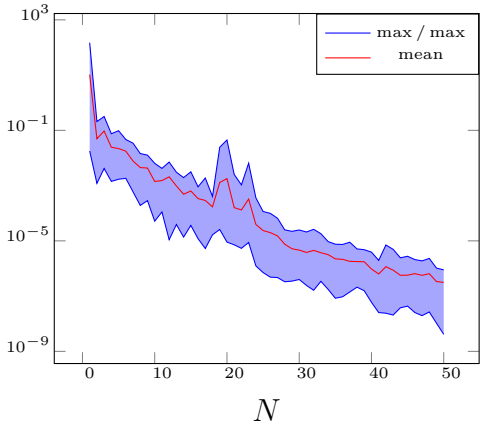


(f) Pressure Profile with Contours, $\boldsymbol{\mu} = (0.5, 0.2)$

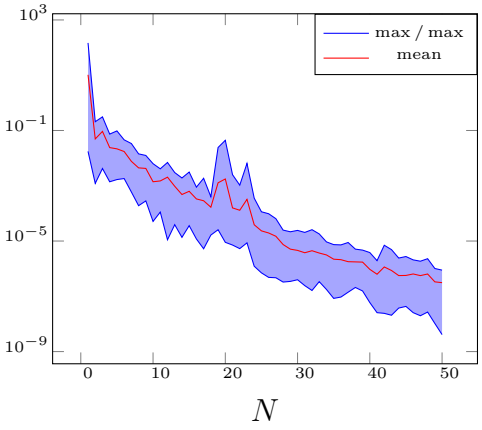
Figure 5.4 – Velocity and pressure profiles for 3 different parameters.



(a) $\|e_N^{\boldsymbol{u}}(\boldsymbol{\mu})\|_2 = \|\boldsymbol{u}_h(\boldsymbol{\mu}) - \boldsymbol{u}_N(\boldsymbol{\mu})\|_2$



(b) $\|e_N^{p}(\boldsymbol{\mu})\|_2 = \|p_h(\boldsymbol{\mu}) - p_N(\boldsymbol{\mu})\|_2$



(c) $\|e_N(\boldsymbol{\mu})\|_2 = \sqrt{\|e_N^{\boldsymbol{u}}(\boldsymbol{\mu})\|_2^2 + \|e_N^{p}(\boldsymbol{\mu})\|_2^2}$

Figure 5.5 – Convergence of the errors with respect to the number of vectors in the reduced basis. Maximum, minimum and mean values evaluated from 100 approximations.

Conclusion

In this chapter, we presented the main ingredients of the Reduced Basis Method. We started by introducing the method with the coercive problems. Then we detailed the algorithms required for the construction of a stable reduced space in the case of saddle-point problems. In the last section of this chapter, we tested the implementation of our RB saddle-point framework on a Stokes problem with geometric parameters.

The theoretical part of this chapter was essentially a reminder of techniques which are already well developed in the literature. However, it seemed important to rewrite these algorithms. Indeed, the RBM was already implemented in Feel++ for coercive problems, but the framework for decoupled multi-physics problems (and saddle-point in particular) was entirely implemented during this thesis work. This implementation is detailed in part III.

With the validation of our framework for saddle-point problems, we can now consider the reduction of Stokes problems. It is a first step towards the reduction of Navier-Stokes and aerothermal problems. We now need tools to deal with the non-linearity. The next chapter will present solutions like the Empirical Interpolation Method which allows building an affine approximation of non-affine and non-linear operators.

CHAPTER 6

Non-Affine and Non-Linear Problems

Contents

We presented in the previous chapter the Reduced Basis Method for two different kinds of problems, all with the common assumptions: the affine decomposition and the linearity. In this chapter, we will present solutions to deal with complex problems which might be non-affine or non-linear.

One of the main advantages of the RBM is the offline/online procedure which allows very fast online $\mathcal{N}$-independent computations, once the expensive offline phase is done. However, it is only possible thanks to the affine decomposition of the operators as we detailed in the section 1.1. Many problems do not present an affine decomposition. For instance, we cannot recover an affine dependency for the

stabilization operators introduced in the chapter 3. Another limitation of the RBM as presented in the previous chapter is the delicate problem of non-linear operators. We need a solution to deal with the non-linearity. The iterative solvers introduced in the chapter 2 allows to solve the non-linearity. However, we want to keep an online resolution independent of the FE dimension $\mathcal{N}$, even for non-linear problems. The Empirical Interpolation Method (EIM) [3, 65, 23, 37] is an answer to both of these problems: the non-affine dependency and the non-linearity of the models. This method allows recovering an affine approximation which will be used within the RBM.

In the present chapter, we first remind the principle of this method. Then we propose a discrete version of the EIM which was implemented during this thesis. And finally, we present the Simultaneous EIM and RB (SER) construction which was also integrated into our discrete version of EIM. The last section is dedicated to a brief validation of our implementation.

1 The Empirical Interpolation Method

Assuming that a parametrized PDE is not affine in parameter $\boldsymbol{\mu}$ mean that it depends at least on a non-affine - and potentially non-linear - function $w(u(\boldsymbol{\mu}), \boldsymbol{x}, \boldsymbol{\mu}) : X \times \mathbb{R}^d \times \mathcal{D} \to \mathbb{R}$. In this section we present how the Empirical Interpolation Method allows to recover an affine approximation w_M of this function such that

$$w(u(\boldsymbol{\mu}), \boldsymbol{x}, \boldsymbol{\mu}) \approx w_M(u(\boldsymbol{\mu}), \boldsymbol{x}, \boldsymbol{\mu}) = \sum_{m=1}^{M} \beta_m(u(\boldsymbol{\mu}), \boldsymbol{\mu}) q_m(\boldsymbol{x}). \tag{6.1}$$

The construction of this approximation requires an expensive offline phase which essentially consists in the selection of a set of parameters $\bar{S}_M = \bar{\boldsymbol{\mu}}_m, 1 \leq m \leq M$. From this set of parameters will be built a function space $\bar{W}_M$ and a set of interpolation point $\bar{T}_M$. We detail this greedy algorithm in the next paragraphs.

Offline Construction : Greedy Algorithm

Let $\bar{\Xi} \subset \mathcal{D}$ a sample of the parameter space $\mathcal{D}$. The algorithm starts with an arbitrary choice of $\bar{\boldsymbol{\mu}}_1 \in \bar{\Xi}$, assuming that $\bar{\xi}_1 = w(u(\bar{\boldsymbol{\mu}}_1), \cdot, \bar{\boldsymbol{\mu}}_1) \neq 0$. The first interpolation point t_1 is chosen as the maximum of the function $\bar{\xi}_1$ over the domain Ω. The first basis function q_1 is then a normalization of the snapshot $\bar{\xi}_1$,

$$t_1 = \arg\max_{\boldsymbol{x} \in \Omega} |\bar{\xi}_1(\boldsymbol{x})|, \qquad q_1 = \frac{\bar{\xi}_1}{\bar{\xi}(t_1)}. \tag{6.2}$$

And then we set

$$\bar{S}_1 = \{\bar{\boldsymbol{\mu}}_1\}, \quad \bar{W}_1 = \operatorname{span}\{\bar{\xi}_1\} = \operatorname{span}\{q_1\}, \quad T_1 = \{t_1\} \tag{6.3}$$

Once this initialization is done, for any $M \geq 2$, the next parameters $\bar{\boldsymbol{\mu}}_M$ are chosen to maximize the EIM approximation error,

$$\bar{\boldsymbol{\mu}}_M = \arg\max_{\bar{\boldsymbol{\mu}} \in \bar{\Xi}} ||w(u(\bar{\boldsymbol{\mu}}), \cdot, \bar{\boldsymbol{\mu}}) - w_{M-1}(u(\bar{\boldsymbol{\mu}}), \cdot, \bar{\boldsymbol{\mu}})||_\infty. \tag{6.4}$$

From this new parameter $\bar{\boldsymbol{\mu}}_M$ we define the residual, between the exact function w and its affine approximation w_M of rank M,

$$r_M(\boldsymbol{x}) = w(u(\bar{\boldsymbol{\mu}}_M), \boldsymbol{x}, \bar{\boldsymbol{\mu}}_M) - w_M(u(\bar{\boldsymbol{\mu}}_M), \boldsymbol{x}, \bar{\boldsymbol{\mu}}_M). \tag{6.5}$$

The next interpolation point t_M is chosen to maximize this residual. The next basis function q_M is the normalization of the residual.

$$t_M = \arg\sup_{x \in \Omega} |r_M(\boldsymbol{x})|, \qquad q_M = \frac{r_M(\boldsymbol{x})}{|r_M(t_M)|}. \tag{6.6}$$

And we can set

$$\bar{S}_M = \bar{S}_{M-1} \cup \{\bar{\boldsymbol{\mu}}_M\}, \qquad \bar{W}_M = \bar{W}_{M-1} \oplus \text{span}\{q_M\}, \qquad T_M = T_{M-1} \cup \{t_M\}. \tag{6.7}$$

Online : Evaluation of the $\beta_m(u, \boldsymbol{\mu})$

For a new parameter $\boldsymbol{\mu}$, and a function $u \in X$, the coefficients $\beta_m(u, \boldsymbol{\mu})$ are evaluated in order to ensure the exactness of the affine approximation w_M at the M interpolation points $\{t_m\}_{m=1}^M$. In practice, the computation is made through the resolution of a $M \times M$ system,

$$\boldsymbol{A}_M \boldsymbol{\beta}_M(u, \boldsymbol{\mu}) = \boldsymbol{F}_M(u, \boldsymbol{\mu}), \tag{6.8}$$

where $(\boldsymbol{A}_M)_{ij} = q_j(t_i)$, $(\boldsymbol{\beta}_M(u, \boldsymbol{\mu}))_i = \beta_i(u, \boldsymbol{\mu})$ and $(\boldsymbol{F}_M(u, \boldsymbol{\mu}))_i = w(u, t_i, \boldsymbol{\mu})$. Once this resolution is made, we can build the affine approximation

$$w(u, x, \boldsymbol{\mu}) \approx w_M(u, \boldsymbol{x}, \boldsymbol{\mu}) = \sum_{m=1}^{M} \beta_m(u, \boldsymbol{\mu}) q_m(\boldsymbol{x}). \tag{6.9}$$

The considered matrix $\boldsymbol{A}_M$ is independent of the parameter $\boldsymbol{\mu}$ and can be assembled and stored during the offline phase. But the right hand side $\boldsymbol{F}_M(\boldsymbol{\mu})$ has to be rebuilt for each new parameter $\boldsymbol{\mu}$, by evaluating the function w on the interpolation points.

1.1 Reduced Basis for Non-Linear Problems

We detailed above how to recover an affine approximation to efficiently use the RBM with non-affine problems. We are now interested in using the EIM with an iterative solver for non-linear problems.

Let consider a non-linear parametrized problem. We assume that we have an iterative algorithm - Picard or Newton for instance - in which the solution at the k-th iteration is denoted by $u^k(\boldsymbol{\mu})$. We suppose that a step of the considered iterative method can be express as find $u^{k+1}(\boldsymbol{\mu})$ such that

$$a(u^{k+1}(\boldsymbol{\mu}), v; u^k(\boldsymbol{\mu}); \boldsymbol{\mu}) = f(v; u^k(\boldsymbol{\mu}); \boldsymbol{\mu}), \quad \forall v \in X, \tag{6.10}$$

where X is a suitable function space on domain Ω. $a(\cdot, \cdot; u; \boldsymbol{\mu}) : X \times X \to \mathbb{R}$ and $l(\cdot; u; \boldsymbol{\mu}) : X \to$ are continuous bilinear and linear forms, for any parameter $\boldsymbol{\mu} \in \mathcal{D}$ and any $u \in X$.

The RBM will not be applied directly on the parametrized problem (6.10), which is obviously not affine in parameters, but on an affine approximation given by the EIM, written in a generic formulation as

$$\sum_{q=1}^{Q_a^M} \theta_q^a(u^k(\boldsymbol{\mu}); \boldsymbol{\mu}) a_q(u^{k+1}, v) = \sum_{q=1}^{Q_f^M} \theta_q^f(u^k(\boldsymbol{\mu}); \boldsymbol{\mu}) f_q(v), \quad \forall v \in X \tag{6.11}$$

This affine approximation is obtained using once or several times the EIM algorithm, depending on the number of non-affine components of the original problem. Note that some of the coefficients θ_q^a and θ_q^f might not depend on the iterates u^k and remain linear. Some of these coefficients might also come from an affine part of the original problem, their evaluation is then analytical and does not require the use of EIM.

Once we have an affine decomposition thanks to the EIM, we can perform a usual RB offline/online procedure. Offline, the resolution is made using our favorite iterative solver until convergence, $u^k(\boldsymbol{\mu}) \to u(\boldsymbol{\mu})$. This solution -snapshot- will be used to build the RB space X_N, as detailed in chapter 5.

The online resolution will require some non-linear iterations as well, which can be written, using the same affine approximation: find $u_N^{k+1}(\boldsymbol{\mu})$ such that

$$\sum_{q=1}^{Q_a^M} \theta_q^a(u_N^k(\boldsymbol{\mu}); \boldsymbol{\mu}) a_q(u_N^{k+1}, v) = \sum_{q=1}^{Q_f^M} \theta_q^f(u_N^k(\boldsymbol{\mu}); \boldsymbol{\mu}) f_q(v), \quad \forall v \in X_N. \tag{6.12}$$

We rewrite this equation under a matricial formulation after the projection on the RB space $W_N = \text{span}\{\xi_n, 1 \leq n \leq N\}$,

$$\sum_{j=1}^{N} \sum_{q=1}^{Q_a^M} \sum_{n=1}^{N} \theta_q^a(u_N^k(\boldsymbol{\mu}); \boldsymbol{\mu}) (\boldsymbol{A}_N^q)_{ij} u_{Nj}^{k+1} = \sum_{q=1}^{Q_f^M} \theta_q^f(u_N^k(\boldsymbol{\mu}); \boldsymbol{\mu}) (\boldsymbol{F}_N^q)_i, \quad 1 \leq i \leq N \tag{6.13}$$

where, $\forall 1 \leq i, j \leq N$,

$$(\boldsymbol{A}_N^q)_{ij} = a_q(\xi_j, \xi_i) \quad (\boldsymbol{F}_N^{q'})_j = f_{q'}(\xi_j) \quad \forall 1 \leq q \leq Q_a^M, \ \forall 1 \leq q' \leq Q_f^M \tag{6.14}$$

The matrices $\boldsymbol{A}_N^q$ and the vectors $\boldsymbol{F}_N^q$ are precomputed once during the online phase. The online resolution then consists in several resolutions of the $N \times N$ system

$$\left[\sum_{q=1}^{Q_a^M} \theta_q^a(u_N^k(\boldsymbol{\mu});\boldsymbol{\mu})\boldsymbol{A}_N^q\right] u_N^{k+1}(\boldsymbol{\mu}) = \left[\sum_{q=1}^{Q_f^M} \theta_q^f(u_N^k(\boldsymbol{\mu});\boldsymbol{\mu})\boldsymbol{F}_N^q\right]. \tag{6.15}$$

The reduced solution $u_N(\boldsymbol{\mu})$ is obtained after convergence of the non-linear solver, $u_N^k(\boldsymbol{\mu}) \to u_N(\boldsymbol{\mu})$.

2 EIM for Discrete Operators

We saw in the previous section that the Empirical Interpolation Method is a required enabler for RBM applied to non-affine or non-linear problems. In the Feel++ framework, this method was already implemented, see [105], but it required an analytical expression of the non-affine component w. Also it was not adapted for the stabilization operators presented in chapter 3. To tackle this problem, we implemented a discrete version of the algorithm, known in the literature as multi-component EIM or Matrix EIM [97]. This method is also called Discrete Empirical Interpolation Method (DEIM) [16], when a Proper Orthogonal Decomposition (POD) is used to select the EIM vectors. The method and the global algorithm are identical to the one described in previous section but the formalism and the implementation, detailed in chapter 10, are different, and then we chose to dedicate a section to this particular version of the EIM.

The algorithm aims at building an affine approximation of a parametrized tensor $\boldsymbol{T}(u(\boldsymbol{\mu}),\boldsymbol{\mu})$. The algorithm as presented in this section is independent of the order of the tensor, $r = 1, 2, ...$ and can be used either for discrete linear or bilinear operators. The affine approximation of tensor $\boldsymbol{T}(u(\boldsymbol{\mu}),\boldsymbol{\mu})$ can be express as

$$\boldsymbol{T}(u(\boldsymbol{\mu}),\boldsymbol{\mu}) \approx \boldsymbol{T}_M(u(\boldsymbol{\mu}),\boldsymbol{\mu}) = \sum_{m=0}^{M} \beta_m(\boldsymbol{\mu})\boldsymbol{\Phi}^m \tag{6.16}$$

where $\beta_m(\boldsymbol{\mu})$ are scalar functions, $\beta_m : \mathcal{D} \to \mathbb{R}$, and $\boldsymbol{\Phi}^m$ are order r tensors independent of $\boldsymbol{\mu}$. The affine approximation $\boldsymbol{T}_M$ lives in a space $\bar{W}_M = \text{span}\{\boldsymbol{\Phi}^m\}$ of low dimension $M << \mathcal{N}$. The space $\bar{W}_M$ is built during an offline phase, using a greedy construction.

$\bar{W}_1$ is built by choosing an arbitrary first parameter $\bar{\boldsymbol{\mu}}_1$. For this parameter we compute $\boldsymbol{T}(u(\bar{\boldsymbol{\mu}}_1),\bar{\boldsymbol{\mu}}_1)$ and look for the index i_1 of the maximum entry of the tensor $\boldsymbol{T}(u(\bar{\boldsymbol{\mu}}_1),\bar{\boldsymbol{\mu}}_1)$

$$i_1 = \arg\max_{i\in\mathbb{I}} |\boldsymbol{T}(u(\bar{\boldsymbol{\mu}}_1),\bar{\boldsymbol{\mu}}_1)_i|. \tag{6.17}$$

We can then initialize the greedy algorithm with

$$\boldsymbol{\Phi}_1 = \frac{1}{|\boldsymbol{T}(u(\bar{\boldsymbol{\mu}}_1), \bar{\boldsymbol{\mu}}_1)_{i_1}|}\boldsymbol{T}(u(\bar{\boldsymbol{\mu}}_1), \bar{\boldsymbol{\mu}}_1), \qquad \bar{W}_1 = \text{span}\{\boldsymbol{\Phi}_1\}, \qquad \mathcal{I}_1 = \{i_1\}, \qquad \bar{S}_1 = \{\bar{\boldsymbol{\mu}}_1\}. \tag{6.18}$$

Remark 8. *The nature of the indices i depends on the order r of the tensor. An index can be defined, in a general way as $i \in \mathbb{I} = [\![1 : \mathcal{N}]\!]^r$.*

We assume now that $\bar{W}_{M-1}$ is built. The next parameters $\bar{\boldsymbol{\mu}}_M$ is chosen to maximize the norm of the residual, $\boldsymbol{R}_{M-1}(\boldsymbol{\mu})$,

$$\boldsymbol{R}_{M-1}(\boldsymbol{\mu}) = \boldsymbol{T}(u(\boldsymbol{\mu}), \boldsymbol{\mu}) - \boldsymbol{T}_{M-1}(u(\boldsymbol{\mu}), \boldsymbol{\mu}), \qquad \bar{\boldsymbol{\mu}}_M = \arg\max_{\boldsymbol{\mu}\in\bar{\Xi}} ||\boldsymbol{R}_M(\boldsymbol{\mu})||_\infty. \tag{6.19}$$

Then we can select the next interpolation index i_M as the maximum entry of the tensor $\boldsymbol{R}_{M-1}(\bar{\boldsymbol{\mu}}_M)$

$$i_M = \arg\max_{i\in\mathbb{I}} |(\boldsymbol{R}_{M-1}(\bar{\boldsymbol{\mu}}_M))_i|, \tag{6.20}$$

and we set

$$\begin{aligned} &\boldsymbol{\Phi}_M = \frac{1}{|(\boldsymbol{R}_{M-1}(\bar{\boldsymbol{\mu}}_M))_{i_M}|}\boldsymbol{R}_{M-1}(\bar{\boldsymbol{\mu}}_M), \qquad && \bar{W}_M = \bar{W}_{M-1} \oplus \text{span}\{\boldsymbol{\Phi}_M\}, \\ &\mathcal{I}_M = \mathcal{I}_{M-1} \cup \{i_M\}, && \bar{S}_M = \bar{S}_{M-1} \cup \{\bar{\boldsymbol{\mu}}_M\}. \end{aligned} \tag{6.21}$$

The complete offline methodology is summarized in algorithm 5.

Remark 9. *We focused in this chapter on the treatment of the non-linearity of the problems. However, the EIM proposed in this section is also very convenient to automatically provide the affine decomposition of complex affine operators. For instance, it is widespread to have huge and complex affine decomposition when we are working with geometric parameters. EIM allows to recover this decomposition and to avoid the painful calculus session. We successfully tested this use on the Stokes test case 3.*

2.1 Online evaluation

The coefficients $\beta_m(u, \boldsymbol{\mu})$ are defined to ensure the exactness of the interpolation in entries corresponding to the indices $i_m \in \mathcal{I}_M$. Then we can evaluate these coefficients by solving the $M \times M$ system

$$\boldsymbol{A}_M \boldsymbol{\beta}_M(u, \boldsymbol{\mu}) = \boldsymbol{F}_M(u, \boldsymbol{\mu}), \tag{6.22}$$

where $(\boldsymbol{A}_M)_{lj} = (\boldsymbol{\Phi}_j)_{i_l}$, $(\boldsymbol{\beta}_M(u, \boldsymbol{\mu}))_l = \beta_l(u, \boldsymbol{\mu})$ and $(\boldsymbol{F}_M(u, \boldsymbol{\mu}))_l = (\boldsymbol{T}(u, \boldsymbol{\mu}))_{i_l}$.

The matrix $\boldsymbol{A}_M$ is built during the offline phase of the algorithm since it does not depend on the parameter $\boldsymbol{\mu}$. However, the assembly of the vector $\boldsymbol{F}_M(\boldsymbol{\mu})$ is

Algorithm 5: EIM offline algorithm for discrete operators

Choose $\bar{\Xi} \subset \mathcal{D}$; // Super Samplings
Choose $M_{\max} \in \mathbb{N}$; // Max size for $\bar{W}_M$
Choose δ_{EIM} ; // Algorithm tolerances
Choose $\bar{\boldsymbol{\mu}}_1 \in \bar{\Xi}$; // Initialization of EIM construction
Set $i_M \leftarrow \arg\max_{i \in \mathbb{I}} |\boldsymbol{T}(u_h(\bar{\boldsymbol{\mu}}_1), \bar{\boldsymbol{\mu}}_1)_i|$, $\boldsymbol{\Phi}_M \leftarrow \frac{\boldsymbol{T}(u_h(\bar{\boldsymbol{\mu}}_1), \bar{\boldsymbol{\mu}}_1)}{|(\boldsymbol{T}(u_h(\bar{\boldsymbol{\mu}}_1), \bar{\boldsymbol{\mu}}_1))_{i_1}|}$;
Set $M \leftarrow 1$, $\bar{S}_1 \leftarrow \{\bar{\boldsymbol{\mu}}_1\}$, $W_1 \leftarrow \{\boldsymbol{\Phi}_1\}$, $\mathcal{I}_1 \leftarrow \{i_1\}$;
repeat
  $\bar{\boldsymbol{\mu}}_{M+1} \leftarrow \arg\max_{\boldsymbol{\mu} \in \bar{\Xi}} || \underbrace{\boldsymbol{T}(u_N(\boldsymbol{\mu}), \boldsymbol{\mu}) - \boldsymbol{T}_M(u_N(\boldsymbol{\mu}), \boldsymbol{\mu})}_{\boldsymbol{R}_M(\boldsymbol{\mu})} ||_\infty$;
  if $||\boldsymbol{R}_M(\bar{\boldsymbol{\mu}}_{M+1})||_\infty \leq \delta_{EIM}$ **then**
    break;
  else
    $M \leftarrow M + 1$;
    $i_M \leftarrow \arg\max_{i \in \mathbb{I}} |(\boldsymbol{R}_{M-1}(\bar{\boldsymbol{\mu}}_M))_i|$;
    $\boldsymbol{\Phi}_M \leftarrow \frac{\boldsymbol{R}_{M-1}(\bar{\boldsymbol{\mu}}_M)}{|(\boldsymbol{R}_{M-1}(\bar{\boldsymbol{\mu}}_M))_{i_M}|}$;
    $\bar{S}_M \leftarrow \bar{S}_{M-1} \cup \{\bar{\boldsymbol{\mu}}_M\}$, $\bar{W}_M \leftarrow \bar{W}_{M-1} \oplus \text{span}\{\{\}\boldsymbol{\Phi}_M\}$, $\mathcal{I}_M = \mathcal{I}_{M-1} \cup \{i_M\}$;
  end
until $M \geq M_{\max}$;

not trivial if we want to guarantee the $\mathcal{N}$-independence of the online phase. We have to assemble this vector for each new evaluation of the coefficients $\beta_m(u, \boldsymbol{\mu})$. Since $\boldsymbol{T}(u, \boldsymbol{\mu})$ depends non-affinely of the parameter $\boldsymbol{\mu}$, we cannot pre-compute anything. A trivial solution would be to reassemble the *full* tensor $\boldsymbol{T}(u, \boldsymbol{\mu})$ and then to extract the entries corresponding to the indices in $\mathcal{I}_M$. However, this assembly depends directly on the FEM dimension $\mathcal{N}$, and is potentially very costly. To remain independent of this high dimension $\mathcal{N}$ we have to assemble the tensor only on the necessary degrees of freedom. We detail this procedure in the next paragraphs and its implementation in chapter 10.

Thanks to the compact support nature of the basis functions, we only need to save a limited number of data to rebuild a reduced tensor $\boldsymbol{T}_{\mathcal{I}_M}(u(\boldsymbol{\mu}), \boldsymbol{\mu})$. We first define the set $D_{\mathcal{I}_M}$ as the set of degrees of freedom (DoF) associated with the indices in $\mathcal{I}_M$. Then we define $N_{\mathcal{I}_M}$ the set of nodes in the triangulation $\mathcal{T}_h$ of the domain Ω, associated to the DoFs in $D_{\mathcal{I}_M}$. From this set of nodes $N_{\mathcal{I}_M}$, we can extract an interpolation mesh $\mathcal{T}_{\mathcal{I}_M}$ out of the original $\mathcal{T}_h$. This sub-mesh is composed with the elements of $\mathcal{T}_h$ containing at least one node of $N_{\mathcal{I}_M}$. The interpolation mesh is then of very low dimension $\mathcal{O}(M) << \mathcal{N}$. We construct and store this mesh at the end of the EIM greedy algorithm when the space $\bar{W}_M$ is complete. From this interpolation mesh, we can build an interpolation function space $X_{\mathcal{I}_M} = X_h(\mathcal{T}_{\mathcal{I}_M})$. This low dimensional function space will be used to assemble the tensor $\boldsymbol{T}_{\mathcal{I}_M}(u(\boldsymbol{\mu}), \boldsymbol{\mu})$, for any $\boldsymbol{\mu} \in \mathcal{D}$, only on the needed DoFs.

2.2 Operator on a Product Space

An issue appeared when we applied EIM to discrete operators on product of spaces: when the physics corresponding to the different spaces do not have the same order of magnitude, the use of EIM on the whole operator may totally "miss" the non-affine dependency of one (or more) block in the composite operator. We propose here to build multiple EIM, one for each block, to correctly decompose each sub-operator.

Let $W = X^1, ..., X^{N_s}$, N_s Hilbert spaces. We also introduce N_s FE space $W_h = X_h^1, ..., X_H^{N_s}$ such that $X_h^i \subset X^i$, $\forall 1 \leq i \leq N_s$. The respective dimension of each FE space X_h^i is denoted by $\mathcal{N}_i$ and $\mathcal{N} = \sum_{i=1}^{N_s} \mathcal{N}_i$. We consider the discrete parametrized operator $\boldsymbol{T}(w(\boldsymbol{\mu}); \boldsymbol{\mu})$, typically obtained by the projection of a parametrized system of PDE on W_h. To set these ideas, we take here the example of a bilinear discrete operator $\boldsymbol{T}(\cdot;\cdot) : W \times \mathcal{D} \to \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$. The assembled matrix $\boldsymbol{T}(w(\boldsymbol{\mu}); \boldsymbol{\mu})$ can be written under a block form,

$$\boldsymbol{T}(w(\boldsymbol{\mu}); \boldsymbol{\mu}) = \begin{bmatrix} \boldsymbol{M}^{11}(w(\boldsymbol{\mu}); \boldsymbol{\mu}) & \cdots & \boldsymbol{M}^{1N_s}(w(\boldsymbol{\mu}); \boldsymbol{\mu}) \\ \vdots & & \vdots \\ \boldsymbol{M}^{N_s1}(w(\boldsymbol{\mu}); \boldsymbol{\mu}) & \cdots & \boldsymbol{M}^{N_sN_s}(w(\boldsymbol{\mu}); \boldsymbol{\mu}) \end{bmatrix}. \tag{6.23}$$

From this point we can consider two options. The first one is to perform an EIM algorithm on the whole matrix/tensor $\boldsymbol{T}$. The other one is to perform multiple EIM algorithm on each submatrices $\boldsymbol{M}_{ij}$. With this second solution we ensure that all the different physics of the system have a proper affine approximation of maximum rank M. On the other hand, performing an EIM on each block can be very costly and not necessary if the physics are of comparable order of magnitude.

The affine approximation of maximum rank M for $\boldsymbol{T}$ is then express, in a block version,

$$\boldsymbol{T}_M(w(\boldsymbol{\mu}); \boldsymbol{\mu}) = \begin{bmatrix} \boldsymbol{M}_M^{11}(w(\boldsymbol{\mu}); \boldsymbol{\mu}) & \cdots & \boldsymbol{M}_M^{1N_s}(w(\boldsymbol{\mu}); \boldsymbol{\mu}) \\ \vdots & & \vdots \\ \boldsymbol{M}_M^{N_s1}(w(\boldsymbol{\mu}); \boldsymbol{\mu}) & \cdots & \boldsymbol{M}_M^{N_sN_s}(w(\boldsymbol{\mu}); \boldsymbol{\mu}) \end{bmatrix} \tag{6.24}$$

where, for any $1 \leq i, j \leq N_s$,

$$\boldsymbol{M}_M^{ij}(w; \boldsymbol{\mu}) = \sum_{m=1}^{M} \beta_m^{ij}(w; \boldsymbol{\mu}) \boldsymbol{\Phi}_m^{ij}. \tag{6.25}$$

Remark 10. *A lot of the β_m^{ij} in this generic formulation might be zero in practice. Also, we do not need to run the EIM on each block but only on blocks with non-affine / non-linear contributions.*

3 Simultaneous Empirical Interpolation and Reduced Basis Method for Non-linear Problems

We detailed in the two previous section the EIM algorithm allowing to recover an affine approximation for non-affine and non-linear problems. During the greedy algorithm, an important part is the selection of the parameters $\bar{\boldsymbol{\mu}}_m \in \bar{S}_M$. For any rank $M \geq 2$, the new parameter $\bar{\boldsymbol{\mu}}_M$ is chosen as

$$\bar{\boldsymbol{\mu}}_M = \arg\max_{\boldsymbol{\mu}\in\bar{\Xi}} ||\boldsymbol{R}_{M-1}(\boldsymbol{\mu})||_\infty = \arg\max_{\boldsymbol{\mu}\in\bar{\Xi}} ||\boldsymbol{T}(u(\boldsymbol{\mu}),\boldsymbol{\mu}) - \boldsymbol{T}_{M-1}(u(\boldsymbol{\mu}),\boldsymbol{\mu})||_\infty. \tag{6.26}$$

The evaluation of $\bar{\boldsymbol{\mu}}_M$ then requires to compute $\boldsymbol{T}(u(\boldsymbol{\mu}),\boldsymbol{\mu})$ for any $\boldsymbol{\mu} \in \bar{\Xi}$. In the case of non-linear problems, this implies to solve a non-linear FEM problem for each $\boldsymbol{\mu} \in \bar{\Xi}$. Depending of the dimension of the problem, this is often really costly and not reachable for too large sampling $\bar{\Xi}$. A solution, proposed in [22], is to use the Simultaneous Empirical Interpolation and Reduced Basis (SER) method in order to build alternatively the RB space $X_N = \text{span}\{\xi_n\}$ and the EIM space $\bar{W}_M = \text{span}\{\boldsymbol{\Phi}_m\}$. The RB approximation $u_N(\boldsymbol{\mu})$ is then used in the evaluation of the residual $\boldsymbol{R}_M(\boldsymbol{\mu})$ instead of the FEM solution $u_h(\boldsymbol{\mu})$.

The SER algorithm was already interfaced with EIM in the Feel++ library. However, we interfaced the method with the discrete version described in the previous section. We present the theory of the method in this section. The implementation work is detailed in chapter 10.

We consider a non-linear parametrized problem which requires an affine approximation for the tensor $\boldsymbol{T}(u(\boldsymbol{\mu}),\boldsymbol{\mu})$. Since the reduced basis methodology needs an affine decomposition, SER starts with the initialization of the EIM algorithm, for $M = 1$. During this initialization, the FEM problem is solved once to build the first EIM basis vector $\boldsymbol{\Phi}_1$, see (6.18). With this first EIM basis vector, we have a first affine approximation $\boldsymbol{T}_1(u(\boldsymbol{\mu}),\boldsymbol{\mu}) = \beta_1(u(\boldsymbol{\mu}),\boldsymbol{\mu})\boldsymbol{\Phi}_1$, $\forall \boldsymbol{\mu} \in \mathcal{D}$. This approximation is totally inaccurate but will be used as a starting point to build the first RB vector ξ_1, using the classical RBM detail in chapter 5.

Once the first RB basis is computed, we dispose of a -very bad- first RB approximation $u_1(\boldsymbol{\mu})$, for any $\boldsymbol{\mu} \in \mathcal{D}$. This RB approximation will be used to evaluate the EIM residual $\boldsymbol{R}_{1,1}(\boldsymbol{\mu})$ and chose the next parameter $\bar{\boldsymbol{\mu}}_2$,

$$\bar{\boldsymbol{\mu}}_2 = \arg\max_{\boldsymbol{\mu}\in\bar{\Xi}} ||\boldsymbol{R}_{1,1}(\boldsymbol{\mu})||_\infty = \arg\max_{\boldsymbol{\mu}\in\bar{\Xi}} ||\boldsymbol{T}(u_1(\boldsymbol{\mu}),\boldsymbol{\mu}) - \boldsymbol{T}_1(u_1(\boldsymbol{\mu}),\boldsymbol{\mu})||_\infty \tag{6.27}$$

This iterative construction is summarized in the algorithm 6, for the sake of clarity, we used an elementary version of the offline RB construction but any advanced features presented in chapter 5 could be included in the loop.

SER algorithm iterates between the construction of a new EIM basis vector $\boldsymbol{\Phi}_M$ and a new RB vector ξ_N. The important features of this coconstruction are

- The use of the RB approximation $u_N(\boldsymbol{\mu})$ in the evaluation of $\boldsymbol{R}_{M,N}(\boldsymbol{\mu}) = \boldsymbol{T}(u_N(\boldsymbol{\mu}), \boldsymbol{\mu}) - \boldsymbol{T}_M(u_N(\boldsymbol{\mu}), \boldsymbol{\mu})$. This residual is used in the greedy algorithm to select the next parameter. This major improvement changes the number of FEM resolutions needed from $N_{\text{EIM+RB}} = \text{card}(\bar{\Xi}) + N$ to $N_{\text{SER}} = N + M$, where N is the dimension of the RB space and M the dimension of the EIM space.

- The update of the affine decomposition throughout the RB greedy algorithm.

- The simultaneously growing accuracy of both approximations $u_N(\boldsymbol{\mu})$ and $\boldsymbol{T}_M(u, \boldsymbol{\mu})$. This growing accuracy allows recovering a very good convergence after some iterations in the SER algorithm.

Difference with the Original Algorithm

SER algorithm was first presented in [24] and [22]. In these references, the RB approximation is used in the EIM greedy algorithm to select the parameters but also to build the EIM basis vectors. After some numerical experiments, we decided to use the RB approximation only during the selection process. The basis vectors are still built using the FEM solutions. This modification of the SER algorithm greatly increases the accuracy of the EIM expansion. However, the number of FEM solutions to be computed changes from $N_{\text{SER}^*} = N + 1$ to $N_{\text{SER}} = N + M$.

This increase is not negligible. However, in the original SER algorithm (denoted SER* hereafter), the multi-levels SER*(l) is usually used. It consists of performing multiple passes in SER algorithm, in order to improve the RB approximation. Each new level is built using the RB approximation of the previous level. With this procedure, the number of FEM evaluations is now $N_{\text{SER}^*(l)} = lN + 1$.

With our version of SER, the multi-levels method becomes useless since we already used a truth solution for the construction. We compare the convergence of SER and SER* in the following preliminary results.

4 Preliminary Results

To illustrate the method and also to validate our implementation of the discrete SER algorithm, we propose some numerical results on a 2D non-linear and non-affinely parametrized benchmark, proposed in [37]. The discrete version of EIM does not have a great interest in this test case since we have an analytic expression of the non-linear term. However, this benchmark is excellent to validate our implementation. The interest of the discrete EIM will be illustrated in the chapter 7 with the treatment of the stabilization operators.

Algorithm 6: SER Algorithm

Choose $\Xi \subset \mathcal{D}$, $\bar{\Xi} \subset \mathcal{D}$; // Super Samplings
Choose $N_{\max} \in \mathbb{N}$, $M_{\max} \in \mathbb{N}$; // Max size for X_N and $\bar{W}_M$
Choose δ_{EIM}, δ_{RB} ; // Algorithm tolerances
Choose $\boldsymbol{\mu}_1 \in \Xi$; // Initialization of RB construction
Set $N \leftarrow 0$, $S_0 \leftarrow \{\}$, $X_0 \leftarrow \emptyset$;
Choose $\bar{\boldsymbol{\mu}}_1 \in \bar{\Xi}$; // Initialization of EIM construction
Set $i_M \leftarrow \arg\max_{i \in \mathbb{I}} |\boldsymbol{T}(u_h(\bar{\boldsymbol{\mu}}_1), \bar{\boldsymbol{\mu}}_1)_i|$, $\boldsymbol{\Phi}_M \leftarrow \frac{\boldsymbol{T}(u_h(\bar{\boldsymbol{\mu}}_1), \bar{\boldsymbol{\mu}}_1)}{|(\boldsymbol{T}(u_h(\bar{\boldsymbol{\mu}}_1), \bar{\boldsymbol{\mu}}_1))_{i_1}|}$;
Set $M \leftarrow 1$, $\bar{S}_1 \leftarrow \{\bar{\boldsymbol{\mu}}_1\}$, $W_1 \leftarrow \{\boldsymbol{\Phi}_1\}$, $\mathcal{I}_1 \leftarrow \{i_1\}$;
Set $\text{state}_{EIM} \leftarrow$ todo, $\text{state}_{RB} \leftarrow$ todo;
repeat
  if $\text{state}_{RB} =$ todo **then** // RB Iteration
    $N \leftarrow N + 1$;
    $S_N \leftarrow S_{N-1} \cup \{\boldsymbol{\mu}_N\}$;
    $u_h(\boldsymbol{\mu}_N) \leftarrow$ FE solution using current EIM affine approximation;
    $X_N \leftarrow X_{N-1} \oplus \text{span}\{u_h(\boldsymbol{\mu}_N)\}$;
    $\mu_{N+1} = \arg\max_{\mu \in \Xi} \Delta_N(\mu)$;
    if $\Delta_N(\boldsymbol{\mu}_{N+1}) \leq \delta_{RB}$ **or** $N \geq N_{\max}$ **then**
      $\text{state}_{RB} \leftarrow$ finished;
    end
  end
  if $\text{state}_{EIM} =$ todo **then**
    $\bar{\boldsymbol{\mu}}_{M+1} \leftarrow \arg\max_{\boldsymbol{\mu} \in \bar{\Xi}} ||\underbrace{\boldsymbol{T}(u_N(\boldsymbol{\mu}), \boldsymbol{\mu}) - \boldsymbol{T}_M(u_N(\boldsymbol{\mu}), \boldsymbol{\mu})}_{\boldsymbol{R}_{M,N}(\boldsymbol{\mu})}||_\infty$;
    if $||\boldsymbol{R}_{M,N}(\bar{\boldsymbol{\mu}}_{M+1})||_\infty \leq \delta_{EIM}$ **or** $M \geq M_{\max}$ **then**
      $\text{state}_{EIM} \leftarrow$ finished;
    else
      $M \leftarrow M + 1$;
      $i_M \leftarrow \arg\max_{i \in \mathbb{I}} |(\boldsymbol{R}_{M-1}(\bar{\boldsymbol{\mu}}_M))_i|$;
      $\boldsymbol{\Phi}_M \leftarrow \frac{\boldsymbol{R}_{M-1}(\bar{\boldsymbol{\mu}}_M)}{|(\boldsymbol{R}_{M-1}(\bar{\boldsymbol{\mu}}_M))_{i_M}|}$;
      $\bar{S}_M \leftarrow \bar{S}_{M-1} \cup \{\bar{\boldsymbol{\mu}}_M\}$, $\bar{W}_M \leftarrow \bar{W}_{M-1} \oplus \text{span}\{\boldsymbol{\Phi}_M\}$, $\mathcal{I}_M = \mathcal{I}_{M-1} \cup \{i_M\}$;
    end
  end
until $\text{state}_{EIM} =$ finished **and** $\text{state}_{RB} =$ finished;

4.1 Problem Formulation

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

$$-\Delta u(\boldsymbol{\mu}) + \mu_1 \frac{e^{\mu_2 u(\boldsymbol{\mu})} - 1}{\mu_2} = 100 \sin(2\pi x) \sin(2\pi y), \tag{6.28}$$

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

For this case, the output s is the average of u on the whole domain,

$$s(\boldsymbol{\mu}) = \int_\Omega u(\boldsymbol{\mu}) \, d\Omega \, . \tag{6.29}$$

We introduce a discretization of Ω and a FE space $X_h = \mathbb{L}^k_{ch}(\Omega)$. We can then write the weak formulation, $\forall \boldsymbol{\mu} \in \mathcal{D}$, as find $u_h(\boldsymbol{\mu}) \in X_h$ such that

$$\int_\Omega \nabla u_h(\boldsymbol{\mu}) \cdot \nabla v_h \, d\Omega + \int_\Omega \mu_1 \frac{e^{\mu_2 u_h(\boldsymbol{\mu})} - 1}{\mu_2} v_h \, d\Omega \; = \int_\Omega 100 \sin(2\pi x) \sin(2\pi y) v_h \, d\Omega \, , \quad \forall v_h \in X_h \tag{6.30}$$

The term $\mu_1 \frac{e^{\mu_2 u_h(\boldsymbol{\mu})} - 1}{\mu_2}$ will be approximated by an EIM approximation

$$\mu_1 \frac{e^{\mu_2 u_h(\boldsymbol{\mu})} - 1}{\mu_2} = \sum_{m=1}^{M} \beta_m(\boldsymbol{\mu}) g^m. \tag{6.31}$$

This affine approximation will be used to build the RB approximation $u_N(\boldsymbol{\mu})$.

4.2 Numerical Results

The simulation is made with $X_h = \mathbb{L}^2_{ch}(\Omega)$. The mesh is basically a 33×33 grid of simplex. We present a profile of the solution for an arbitrary chosen $\boldsymbol{\mu}$, on figure 6.1. The non-linearity is solved using a fixed point algorithm during the offline and the online phases. We choose a uniform grid 15×15 as sampling for the EIM greedy algorithm. A thinner discretization would be very costly, at least for the classical EIM greedy algorithm. We used the dual norm of the residual as an error indicator in the RB greedy selection.

We compare the truth approximation $u_h(\boldsymbol{\mu})$, computed with the exact formulation, and the RB approximation $u_N(\boldsymbol{\mu})$ computed using the EIM affine approximation. We are particularly interested in the quantities

$$\begin{aligned} e^u_N(\boldsymbol{\mu}) &= \frac{\|u_h(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})\|_0}{\|u_h(\boldsymbol{\mu})\|_0}, \\ e^s_N(\boldsymbol{\mu}) &= \frac{|s_h(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})|}{|s_h(\boldsymbol{\mu})|}. \end{aligned} \tag{6.32}$$

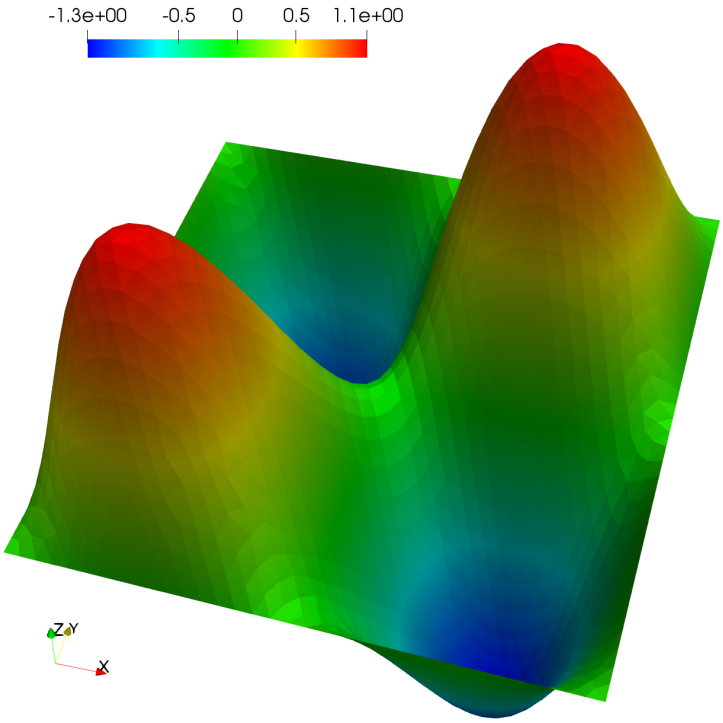
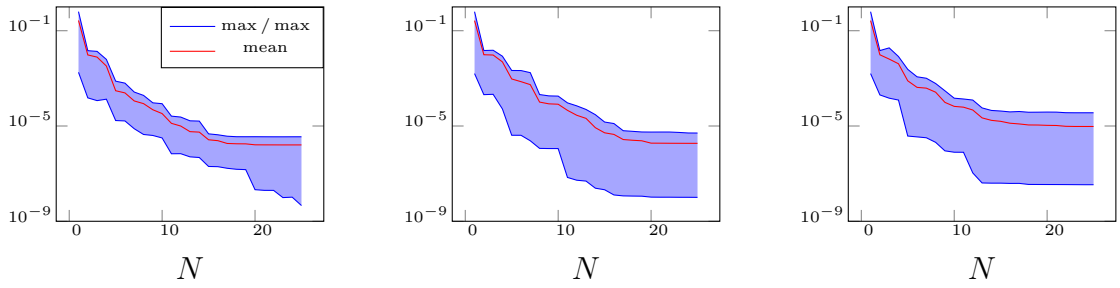


Figure 6.1 – Example of the solution for problem (6.28)
Profile of the FE solution $u_h(\boldsymbol{\mu})$ with $\boldsymbol{\mu} = (1.91, 2.77)$.

This comparison is made for 100 different parameters, chosen randomly in $\mathcal{D}$. We study the convergence of these errors with $M = 25$ EIM basis vectors. We also considered three different configurations: EIM with classical greedy, SER and SER* (see paragraph 32). We plotted the evolution of these errors for each configuration on figures 6.2 and 6.3. We also compare the maximal errors for each method on figure 6.4.

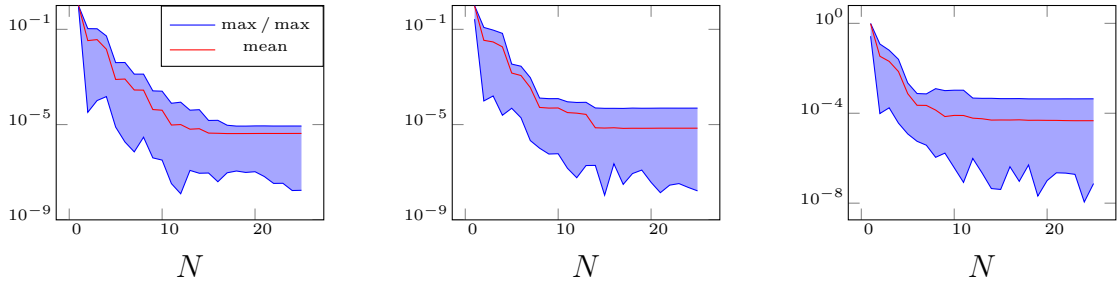
The variant SER*, which uses the RB approximation to build the EIM basis vector, is apparently less accurate than our version of the SER algorithm. This method will not be used anymore. The convergence is slightly slower than the convergence of the EIM solution, but eventually, the errors are comparable.

The results of this benchmark validate our implementation of SER with our new EIM. The convergence study is very promising concerning the use of SER. It is now time to test these methods on more complicated cases. We propose an interesting application in the next chapter, with the reduction of the stabilization operators introduced in 3.



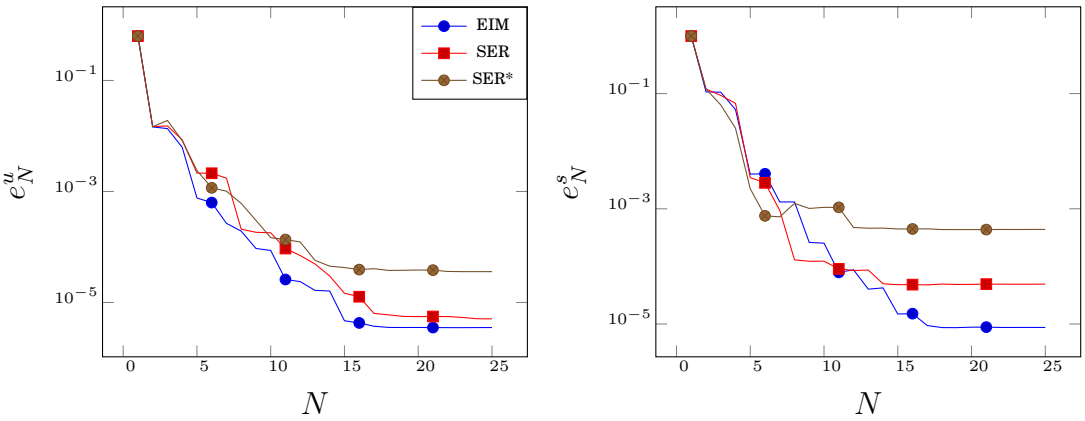
(a) EIM Classical Greedy (b) SER (c) SER*

Figure 6.2 – Convergence of the errors $e_N^u(\boldsymbol{\mu}) = \frac{\|u_h(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})\|_0}{\|u_h(\boldsymbol{\mu})\|_0}$ with respect to the number of vectors in the reduced basis. Maximum, minimum and mean values evaluated from 100 approximations. $M_{\max} = 25$



(a) EIM Classical Greedy (b) SER (c) SER*

Figure 6.3 – Convergence of the errors $e_N^s(\boldsymbol{\mu}) = \frac{|s_h(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})|}{|s_h(\boldsymbol{\mu})|}$ with respect to the number of vectors in the reduced basis. Maximum, minimum and mean values evaluated from 100 approximations. $M_{\max} = 25$



(a) $e_N^u(\boldsymbol{\mu}) = \frac{\|u_h(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})\|_2}{\|u_h(\boldsymbol{\mu})\|_2}$ (b) $e_N^s(\boldsymbol{\mu}) = \frac{|s_h(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})|}{|s_h(\boldsymbol{\mu})|}$

Figure 6.4 – Comparison of the maximum error over 100 approximations, for different methods. $M_{\max} = 25$

Conclusion

In this chapter, we detailed the methods which allow applying the RBM to non-linear and non-affine problems. The EIM is the key to recover an affine approximation and then to keep an efficient offline/online procedure. We presented a discrete version of EIM, recently implemented in the Feel++ RB framework. This new implementation of the method allows recovering an affine approximation of vectors or matrices and then to tackle more general problems. The EIM can also separates the non-linear dependency from the spatial one. It allows to use iterative solvers during the online phase and to preserve the $\mathcal{N}$-independence. The offline phase of EIM for non-linear problems is greatly improved using the SER construction. We proposed a slight modification of the original algorithm to improve the accuracy of the approximation provided by the coconstruction.

CHAPTER 7

The Reduced Basis Method Applied to Aerothermal Problems

Contents

The reduction of an aerothermal problems includes some non-trivial issues. In this chapter, we detail the choices we made and the numerical solutions we used to build a reduced version of the framework presented in the first part.

First of all, we have to deal with the Navier-Stokes equations. [78, 49] proposed a first stable approximation for the incompressible Navier-Stokes flow. Rigorous a

posteriori error estimator were first developed later in [102, 21] and more recently in [68, 82]. Using the quadratic non-linearity of the Navier-Stokes equations, the RBM introduced in the chapter 5 can be adapted with no major difficulties. We usually build a stable RB space - in the sense of the Brezzi-Rappaz-Raviart (BRR) condition - by enriching the velocity space as detailed in the section 2.

The RB method for trilinear problems has already been used for natural convection problems [91, 26, 56]. However, the construction of the RB space is usually made monolithically. We propose in this section to build a product space $V_N \times Q_N \times X_N$ to improve the stability and the accuracy of the reduced approximation.

The introduction of the stabilization operators will be treated with the EIM method presented in the previous chapter. This solution was already used for convection dominated problems using DEIM [75]. We proposed here to extend the idea to the whole aerothermal system. Other approaches were proposed for the stabilization of reduced convection dominated problems. In [64] the stabilization in the online stage is made independently of the offline one by adding an appropriate vanishing viscosity in the high RB modes. In [36, 107], the SUPG stabilization method is used with Proper Orthogonal Decomposition (POD) to reduce transient convection dominated problems.

In the section 1, we detail the construction of our reduced aerothermal framework, using the RBM for trilinear multi-physic problems and the EIM for the stabilization operators. Then, the following sections are dedicated to numerical applications. The first one is a simulation with no natural convection in the neighborhood of an electronic component submitted to cooling air flow. The second application is a simulation of mixed natural and forced convection in an airplane cabin proposed by Airbus Group.

1 Reduction of the Aerothermal Model

We consider the problem describe in chapter 2, and we remind here the system of equation,

$$\begin{cases} \rho \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p - 2\nabla \cdot (\mu \boldsymbol{S}(\boldsymbol{u})) = -\rho\beta(T - T_0)\boldsymbol{g}, \\ \nabla \cdot \boldsymbol{u} = 0, \\ \boldsymbol{u} \cdot \nabla T - \nabla \cdot (\kappa \nabla T) = 0, \\ +\text{Boundary Conditions 2.14}. \end{cases} \tag{7.1}$$

We also introduce the Hilbert spaces V, Q and X with the scalar products $(\cdot,\cdot)_V$, $(\cdot,\cdot)_Q$, $(\cdot,\cdot)_X$ and associated norms $||\cdot||_V$, $||\cdot||_Q$, $||\cdot||_X$, respectively. We define the product space $Z = V \times Q$ with the scalar product $((\boldsymbol{u},p),(\boldsymbol{v},q))_Z = (\boldsymbol{u},\boldsymbol{v})_V + (p,q)_Q$ and associated norm $||(\cdot,\cdot)||_Z = \sqrt{||\cdot||_V + ||\cdot||_Q}$. We finally note the full composite space $W = V \times Q \times X$ with scalar product $(\cdot,\cdot)_W$ and norm $\|\cdot\|_W$.

We supposed that the problem 7.1 is parametrized by a parameter $\boldsymbol{\mu} \in \mathcal{D} \in \mathbb{R}^P$

of dimension $P \geq 1$. The discretization of this system, using the finite element method (see chapter 2, section 2), leads to the weak problem: for any $\boldsymbol{\mu} \in \mathcal{D}$, find $(\boldsymbol{u}_h(\boldsymbol{\mu}), p_h(\boldsymbol{\mu}), T_h(\boldsymbol{\mu})) \in V_h \times Q_h \times X_h$ such that

$$\begin{aligned} c(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{v}_h; \boldsymbol{\mu}) + d_u(\boldsymbol{u}_h, \boldsymbol{v}_h; \boldsymbol{\mu}) - b(\boldsymbol{v}_h, p_h; \boldsymbol{\mu}) + b(\boldsymbol{u}_h, q_h; \boldsymbol{\mu}) &= f(T_h, \boldsymbol{v}_h; \boldsymbol{\mu}), \\ a(T_h, \boldsymbol{u}_h, S_h; \boldsymbol{\mu}) + d_T(T_h, S_h; \boldsymbol{\mu}) &= g(S_h; \boldsymbol{\mu}), \\ \forall (\boldsymbol{v}_h, q_h, S_h) \in V_h \times Q_h \times X_h, & \end{aligned} \tag{7.2}$$

where $\forall \boldsymbol{\mu} \in \mathcal{D}$,

- $c : V \times V \times \to \mathbb{R}$ and $a : X \times V \times X \to \mathbb{R}$ are continuous trilinear forms
- $d_u : V \times V \to$, $b : V \times Q \to \mathbb{R}$, $f : X \times V \to \mathbb{R}$ and $d_T : X \times X \to \mathbb{R}$ are continuous bilinear forms
- $g : X \to \mathbb{R}$ is a continuous linear form.

We also assume that these forms can be written using an affine decomposition, $\forall \boldsymbol{\mu} \in \mathcal{D}$,

$$\begin{aligned} c(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}; \boldsymbol{\mu}) &= \sum_{k=1}^{Q_c} \theta_k^c(\boldsymbol{\mu}) c_k(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}), & a(T, \boldsymbol{u}, S; \boldsymbol{\mu}) &= \sum_{k=1}^{Q_a} \theta_k^a(\boldsymbol{\mu}) a_k(T, \boldsymbol{u}, S), \\ d_u(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{\mu}) &= \sum_{k=1}^{Q_{d_u}} \theta_k^{d^{\boldsymbol{u}}}(\boldsymbol{\mu}) d_{u,k}(\boldsymbol{u}, \boldsymbol{v}), & d_T(T, S; \boldsymbol{\mu}) &= \sum_{k=1}^{Q_{d_T}} \theta_k^{d^T}(\boldsymbol{\mu}) d_{T,k}(T, S), \\ b(\boldsymbol{u}, q; \boldsymbol{\mu}) &= \sum_{k=1}^{Q_b} \theta_k^b(\boldsymbol{\mu}) b_k(\boldsymbol{u}, q), & f(T, \boldsymbol{v}; \boldsymbol{\mu}) &= \sum_{k=1}^{Q_f} \theta_k^f(\boldsymbol{\mu}) f_k(T, \boldsymbol{v}) \\ g(\boldsymbol{v}; \boldsymbol{\mu}) &= \sum_{k=1}^{Q_g} \theta_k^g(\boldsymbol{\mu}) g_k(\boldsymbol{v}) & & \end{aligned} \tag{7.3}$$

Truth Approximation

As truth approximation of this problem, we will use the finite element solution. The resolution of this system of equation is widely described in the part I. Thereafter we assume that $\forall \boldsymbol{\mu} \in \mathcal{D}$ we dispose of a FE solution $(\boldsymbol{u}_h(\boldsymbol{\mu}), p_h(\boldsymbol{\mu}), T_h(\boldsymbol{\mu})) \in V \times Q \times X$ of the system 7.1.

1.1 Offline Procedure: Construction of the RB Space

The reduced basis space will be built on a sampling S_N, as described in the chapter 5. This sampling can be iteratively determined using a greedy algorithm, if

we have an error indicator for our problem, or chosen randomly otherwise. In both cases, the aim of the offline procedure it to iteratively construct three set of spaces $V_1 \subset \cdots \subset V_N$, $Q_1 \subset \cdots \subset Q_N$ and $X_1 \subset \cdots \subset X_N$.

The first spaces V_1, Q_1 and X_1 are built as

$$\begin{aligned} V_1 &= \text{span}\{\boldsymbol{u}_h(\boldsymbol{\mu}_1), \boldsymbol{T}^{\boldsymbol{\mu}}\eta_1\} = \text{span}\{\boldsymbol{\zeta}_1, \boldsymbol{\zeta}_2\}, \\ Q_1 &= \text{span}\{p_h(\boldsymbol{\mu}_1)\} = \text{span}\{\eta_1\}, \\ X_1 &= \text{span}\{T_h(\boldsymbol{\mu}_1)\} = \text{span}\{\xi_1\}. \end{aligned} \tag{7.4}$$

As mentioned in the chapter 5, the actual basis $\{\boldsymbol{\zeta}_n\}$, $\{\eta_n\}$ and $\{\xi_n\}$ are obtained from an orthonormalization of the respective snapshots $\{\boldsymbol{u}_h(\boldsymbol{\mu}_n), \boldsymbol{T}^{\boldsymbol{\mu}_n}\eta_n\}$, $\{p_h(\boldsymbol{\mu}_n)\}$ and $\{T_h(\boldsymbol{\mu}_n)\}$. $\boldsymbol{T}^{\boldsymbol{\mu}}$ is the supremizer operator, defined in the chapter 5, section 2. However, the random selection of the $\boldsymbol{\mu}_n$ or the greedy selection might provide snapshots that already live in the RB spaces. For instance, if the difference between the parameters $\boldsymbol{\mu}_{n-1}$ and $\boldsymbol{\mu}_n$ is only acting on the fluid temperature and not on the fluid dynamic, the new snapshots $(\boldsymbol{u}_h(\boldsymbol{\mu}_n), p_h(\boldsymbol{\mu}_n))$ will not be relevant to enrich the space V_n and Q_n. It is then important to eliminate these useless contributions.

This selection phase is made during the orthonormalization process. When a new solution $\boldsymbol{u}_h(\boldsymbol{\mu}_n)$ is computed for the parameter $\boldsymbol{\mu}_n$, we first evaluate its component orthogonal to the space V_{n-1},

$$\boldsymbol{\zeta}_n^* = \boldsymbol{u}_h(\boldsymbol{\mu}_n) - \sum_{k=1}^{N_{n-1}^{\boldsymbol{u}}} (\boldsymbol{u}_h(\boldsymbol{\mu}_n), \boldsymbol{\zeta}_k)_V, \tag{7.5}$$

where $N_{n-1}^{\boldsymbol{u}}$ is the size of the space V_{n-1}. At this point, if the norm, $\|\boldsymbol{\zeta}_n^*\|_V$, is too small, this solution is not normalized and not added to the space V_n. We do the same verification during the orthonormalization of the bases of the spaces Q_n and X_n. This way, we ensure that the vectors of each reduced basis remain linearly independent. A direct consequence of this construction is the variable sizes of the RB spaces. From now on, we denote by N the size of the sampling S_N. We denote respectively by $N_N^{\boldsymbol{u}}$, N_N^p and N_N^T the dimensions of the RB spaces V_N, Q_N and X_N. We have the immediate, but notable relations: $N_N^{\boldsymbol{u}} \leq 2N$, $N_N^p \leq N$ and $N_N^T \leq N$.

We present a summarized version of the selective Gram-Schmidt procedure in the algorithm 7. We also propose a greedy selection for the aerothermal problems in the algorithm 8.

1.2 Online Evaluation

The idea of the online evaluation is similar to the coercive or saddle point problems presented in the chapter 5. We perform a Galerkin projection of our continuous problem on the reduced space $W_N = V_N \times Q_N \times X_N$. The main difference with the current system of equations is the treatment of the non-linearity. We

Algorithm 7: Selective Gram-Schmidt Orthonormalization

Input: $\{\xi_1, \cdots, \xi_N\}$, u_n
Output: $\{\xi_1, \cdots, \xi_N\}$
Choose δ_{tol} ; // Algorithm tolerance
$\xi^* \leftarrow u_n$;
for $1 \leq k \leq N$ **do**
  $\xi^* \leftarrow \xi^* - (u_n, \xi_k)_{X_h}$;
end
if $\|\xi^*\|_X \geq \delta_{tol}$ **then**
  $N \leftarrow N + 1$;
  return $\{\xi_1, \cdots, \xi_N\} \cup \{\frac{1}{\|\xi^*\|_X}\xi^*\}$
else
  return $\{\xi_1, \cdots, \xi_N\}$
end

Algorithm 8: Offline Procedure for Aerothermal Problems

Choose $\Xi \subset \mathcal{D}$; // Super Sampling
Choose $N_{\max}$; // Maximum size of X_N
Choose δ_{tol} ; // Algorithm tolerance
Set $S_N \leftarrow \{\}$, ,$V_N \leftarrow \{\}$, $Q_N \leftarrow \{\}$, $X_N \leftarrow \{\}$; // Initialization
Set $N_0^{\boldsymbol{u}} \leftarrow 0$, $N_0^p \leftarrow 0$, $N_0^T \leftarrow 0$;
repeat
  $S_N \leftarrow S_{N-1} \cup \{\boldsymbol{\mu}_N\}$, $N_N^{\boldsymbol{u}} \leftarrow N_{N-1}^{\boldsymbol{u}}$,$N_N^p \leftarrow N_{N-1}^p$, $N_N^T \leftarrow N_{N-1}^T$;
  Find $(\boldsymbol{u}_h(\boldsymbol{\mu}_N), p_h(\boldsymbol{\mu}_N), T_h(\boldsymbol{\mu}_N))$ solution of system (7.1);
  $\{\eta_1, \cdots, \eta_{N^p}\} \leftarrow$Selective Gram-Schmidt$(\{\eta_1, \cdots, \eta_{N^p}\}, p_h(\boldsymbol{\mu}_N))$;
  if $p_h(\boldsymbol{\mu}_N)$ *added* **then**
    $\{\boldsymbol{\zeta}_1, \cdots, \boldsymbol{\zeta}_{N^{\boldsymbol{u}}}\} \leftarrow$Selective Gram-Schmidt$(\{\boldsymbol{\zeta}_1, \cdots, \boldsymbol{\zeta}_{N^{\boldsymbol{u}}}\}, \boldsymbol{T}^{\boldsymbol{\mu}}\eta_{N^p})$;
  end
  $\{\boldsymbol{\zeta}_1, \cdots, \boldsymbol{\zeta}_{N^{\boldsymbol{u}}}\} \leftarrow$Selective Gram-Schmidt$(\{\boldsymbol{\zeta}_1, \cdots, \boldsymbol{\zeta}_{N^{\boldsymbol{u}}}\}, \boldsymbol{u}_h(\boldsymbol{\mu}_N)\})$;
  $\{\xi_1, \cdots, \xi_{N^T}\} \leftarrow$Selective Gram-Schmidt$(\{\xi_1, \cdots, \xi_{N^T}\}, T_h(\boldsymbol{\mu}_N)\})$;
  $V_N \leftarrow \text{span}\{\boldsymbol{\zeta}_1, \cdots, \boldsymbol{\zeta}_{N_N^{\boldsymbol{u}}}\}$;
  $Q_N \leftarrow \text{span}\{\eta_1, \cdots, \eta_{N_N^p}\}$;
  $X_N \leftarrow \text{span}\{\xi_1, \cdots, \xi_{N_N^T}\}$;
  Precompute online structures;
  for $\mu \in \Xi$ **do**
    Compute error bound, $\Delta_N(\boldsymbol{\mu})$;
  end
  $\mu_{N+1} = \arg\max_{\boldsymbol{\mu} \in \Xi} \Delta_N(\boldsymbol{\mu})$; // Or can be chosen randomly
until $\Delta_N(\boldsymbol{\mu}_{N+1}) < \delta_{tol}$ **or** $N \geq N_{max}$;

choose to use a Newton iterative method to deal with this non-linearity. Note that the method described after is easily adaptable to other iterative methods.

We assume that the RB spaces V_N, Q_N and X_N have been built during the offline phase, with the respective orthonormal basis $\mathcal{B}_N^{\boldsymbol{u}} = \{\boldsymbol{\zeta}_i\}_{i=1}^{N_N^{\boldsymbol{u}}}$, $\mathcal{B}_N^p = \{\eta_i\}_{i=1}^{N_N^p}$ and $\mathcal{B}_N^T = \{\xi_i\}_{i=1}^{N_N^T}$. The Galerkin projection of the problem (7.1) on these spaces, leads to the matricial equation: find $(\boldsymbol{u}_N(\boldsymbol{\mu}), p_N(\boldsymbol{\mu}), T_N(\boldsymbol{\mu}))$ such that

$$\underbrace{\begin{bmatrix} \boldsymbol{D}_N^{\boldsymbol{u}}(\boldsymbol{\mu}) + \boldsymbol{C}_N(\boldsymbol{u}_N;\boldsymbol{\mu}) & \boldsymbol{B}_N(\boldsymbol{\mu})^\intercal & -\boldsymbol{F}_N(\boldsymbol{\mu}) \\ -\boldsymbol{B}_N(\boldsymbol{\mu}) & 0 & 0 \\ 0 & 0 & \boldsymbol{A}_N(\boldsymbol{u}_N;\boldsymbol{\mu}) + \boldsymbol{D}_N^T(\boldsymbol{\mu}) \end{bmatrix}}_{\boldsymbol{M}_N(\boldsymbol{u}_N;\boldsymbol{\mu})} \begin{bmatrix} \boldsymbol{u}_N(\boldsymbol{\mu}) \\ p_N(\boldsymbol{\mu}) \\ T_N(\boldsymbol{\mu}) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ G_N(\boldsymbol{\mu}) \end{bmatrix}, \tag{7.6}$$

where, $\forall \boldsymbol{\mu} \in \mathcal{D}$ and for any $1 \leq i,j \leq N_N^{\boldsymbol{u}}$, $1 \leq m \leq N_N^p$, $1 \leq n,l \leq N_N^T$,

$$\begin{aligned} &(\boldsymbol{D}_N^{\boldsymbol{u}}(\boldsymbol{\mu}))_{ij} = d_u(\boldsymbol{\zeta}_j, \boldsymbol{\zeta}_i; \boldsymbol{\mu}), \qquad (\boldsymbol{C}_N(\boldsymbol{u};\boldsymbol{\mu}))_{ij} = c(\boldsymbol{\zeta}_j, \boldsymbol{u}, \boldsymbol{\zeta}_i; \boldsymbol{\mu}), \qquad (\boldsymbol{F}_N(\boldsymbol{\mu}))_{i,l} = f(\xi_l, \boldsymbol{\zeta}_i; \boldsymbol{\mu}), \\ &(\boldsymbol{B}_N(\boldsymbol{\mu}))_{mj} = b(\boldsymbol{\zeta}_j, \eta_m; \boldsymbol{\mu}) \\ &(\boldsymbol{D}_N^T(\boldsymbol{\mu}))_{nl} = d_T(\xi_l, \xi_n; \boldsymbol{\mu}), \qquad (\boldsymbol{A}_N(\boldsymbol{u};\boldsymbol{\mu}))_{nl} = a(\xi_l, \boldsymbol{u}, \xi_n; \boldsymbol{\mu}), \qquad (\boldsymbol{G}_N(\boldsymbol{\mu}))_n = g(\xi_n; \boldsymbol{\mu}). \end{aligned} \tag{7.7}$$

The system (7.6) can be solved using a Newton method. For that, we have to evaluate, $\forall (\boldsymbol{u}, p, T) \in V_N \times Q_N \times X_N$, $\forall \boldsymbol{\mu} \in \mathcal{D}$ the Jacobian $\boldsymbol{J}_N(\boldsymbol{u}, p, T,\ \boldsymbol{\mu})$ (or Fréchet Derivative) of the application $\mathcal{F}(\boldsymbol{\mu}) : (\boldsymbol{u}, p, T) \to \boldsymbol{M}(\boldsymbol{u};\boldsymbol{\mu})(\boldsymbol{u}, p, T)^\intercal$. We also need the residual $\boldsymbol{R}_N(\boldsymbol{u}, p, T; \boldsymbol{\mu})$ of the system (7.6). The evaluation of these two objects during the online phase has to be independent of the finite element dimension $\mathcal{N}$.

Thanks to the quadratic form of our problem, we can easily write the Jacobian $\boldsymbol{J}_N$, $\forall (\boldsymbol{u}, p, T) \in V_N \times Q_N \times X_N$, $\forall \boldsymbol{\mu} \in \mathcal{D}$

$$\boldsymbol{J}_N(\boldsymbol{u}, p, T; \boldsymbol{\mu}) = \begin{bmatrix} \boldsymbol{D}_N^{\boldsymbol{u}}(\boldsymbol{\mu}) + \boldsymbol{C}_N(\boldsymbol{u}_N;\boldsymbol{\mu}) + \boldsymbol{C}_N^*(\boldsymbol{u}_N;\boldsymbol{\mu}) & \boldsymbol{B}_N(\boldsymbol{\mu})^\intercal & -\boldsymbol{F}_N(\boldsymbol{\mu}) \\ -\boldsymbol{B}_N(\boldsymbol{\mu}) & 0 & 0 \\ \boldsymbol{A}_N^*(T_N;\boldsymbol{\mu}) & 0 & \boldsymbol{A}_N(\boldsymbol{u}_N;\boldsymbol{\mu}) + \boldsymbol{D}_N^T(\boldsymbol{\mu}) \end{bmatrix} \tag{7.8}$$

where, $\forall \boldsymbol{\mu} \in \mathcal{D}$ and for any $1 \leq i,j \leq N_N^{\boldsymbol{u}}$, $1 \leq n,l \leq N_N^T$,

$$(\boldsymbol{C}_N^*(\boldsymbol{u};\boldsymbol{\mu}))_{ij} = c(\boldsymbol{u}, \boldsymbol{\zeta}_j, \boldsymbol{\zeta}_i; \boldsymbol{\mu}), \qquad (\boldsymbol{A}_N^*(\boldsymbol{u};\boldsymbol{\mu}))_{nj} = a(T, \boldsymbol{\zeta}_j, \xi_n; \boldsymbol{\mu}). \tag{7.9}$$

By using the affine decomposition assumption (2.2), we can rewrite the definitions of the matrices $\boldsymbol{D}_N^{\boldsymbol{u}}$, $\boldsymbol{B}_N$, $\boldsymbol{F}_N$ and $\boldsymbol{D}_N^T$, $\forall \boldsymbol{\mu} \in \mathcal{D}$ and for any $1 \leq i,j \leq N_N^{\boldsymbol{u}}$, $1 \leq m \leq N_N^p$, $1 \leq n,l \leq N_N^T$,

$$\begin{aligned} (\boldsymbol{D}_N^{\boldsymbol{u}}(\boldsymbol{\mu}))_{i,j} &= \sum_{k=1}^{Q_{d^{\boldsymbol{u}}}} \theta_k^{d^{\boldsymbol{u}}}(\boldsymbol{\mu}) \underbrace{d_k^{\boldsymbol{u}}(\boldsymbol{\zeta}_j, \boldsymbol{\zeta}_i)}_{\text{precomputed}}, \qquad (\boldsymbol{D}_N^T(\boldsymbol{\mu}))_{n,l} = \sum_{k=1}^{Q_{d^T}} \theta_k^{d^T}(\boldsymbol{\mu}) \underbrace{d_k^T(\xi_l, \xi_n)}_{\text{precomputed}} \\ (\boldsymbol{B}_N(\boldsymbol{\mu}))_{m,j} &= \sum_{k=1}^{Q_b} \theta_k^b(\boldsymbol{\mu}) \underbrace{b_k(\boldsymbol{\zeta}_j, \eta_m)}_{\text{precomputed}}, \qquad (\boldsymbol{F}(\boldsymbol{\mu}))_{i,l} = \sum_{k=1}^{Q_f} \theta_k^f(\boldsymbol{\mu}) \underbrace{f_k(\xi_l, \boldsymbol{\zeta}_i)}_{\text{precomputed}}. \end{aligned} \tag{7.10}$$

We use the same idea to precompute the quadratic contributions, $\forall \boldsymbol{\mu} \in \mathcal{D}$, $\forall (\boldsymbol{u}, p, T) \in V_N \times Q_N \times X_N$,

$$
\begin{aligned}
(\boldsymbol{C}_N(\boldsymbol{u};\boldsymbol{\mu}))_{ij} &= c(\boldsymbol{\zeta}_j, \boldsymbol{u}, \boldsymbol{\zeta}_i; \boldsymbol{\mu}) \\
&= \sum_{n=1}^{N_N^{\boldsymbol{u}}} \boldsymbol{u}_n c(\boldsymbol{\zeta}_j, \boldsymbol{\zeta}_n, \boldsymbol{\zeta}_i; \boldsymbol{\mu}) \\
&= \sum_{k=1}^{Q_c} \sum_{n=1}^{N_N^{\boldsymbol{u}}} \theta_k^c(\boldsymbol{\mu}) \boldsymbol{u}_n \underbrace{c_k(\boldsymbol{\zeta}_j, \boldsymbol{\zeta}_n, \boldsymbol{\zeta}_i)}_{\text{precomputed}}, \\
(\boldsymbol{A}_N(\boldsymbol{u};\boldsymbol{\mu}))_{ij} &= a(\xi_j, \boldsymbol{u}, \xi_i; \boldsymbol{\mu}) \\
&= \sum_{n=1}^{N_N^{\boldsymbol{u}}} \boldsymbol{u}_n a(\xi_j, \boldsymbol{\zeta}_n, \xi_i; \boldsymbol{\mu}) \\
&= \sum_{k=1}^{Q_a} \sum_{n=1}^{N_N^{\boldsymbol{u}}} \theta_k^a(\boldsymbol{\mu}) \boldsymbol{u}_n \underbrace{a_k(\xi_j, \boldsymbol{\zeta}_n, \xi_i)}_{\text{precomputed}},
\end{aligned}
\qquad
\begin{aligned}
(\boldsymbol{C}_N^*(\boldsymbol{u};\boldsymbol{\mu}))_{ij} &= c(\boldsymbol{u}, \boldsymbol{\zeta}_j, \boldsymbol{\zeta}_i; \boldsymbol{\mu}) \\
&= \sum_{n=1}^{N_N^{\boldsymbol{u}}} \boldsymbol{u}_n c(\boldsymbol{\zeta}_n, \boldsymbol{\zeta}_j, \boldsymbol{\zeta}_i; \boldsymbol{\mu}) \\
&= \sum_{k=1}^{Q_c} \sum_{n=1}^{N_N^{\boldsymbol{u}}} \theta_k^c(\boldsymbol{\mu}) \boldsymbol{u}_n \underbrace{c_k(\boldsymbol{\zeta}_n, \boldsymbol{\zeta}_j, \boldsymbol{\zeta}_i)}_{\text{precomputed}}, \\
(\boldsymbol{A}_N^*(T;\boldsymbol{\mu}))_{ij} &= a(T, \boldsymbol{\zeta}_j, \xi_i; \boldsymbol{\mu}) \\
&= \sum_{n=1}^{N_N^T} T_n a(\xi_n, \boldsymbol{\zeta}_j, \xi_i; \boldsymbol{\mu}) \\
&= \sum_{k=1}^{Q_a} \sum_{n=1}^{N_N^T} \theta_k^a(\boldsymbol{\mu}) T_n \underbrace{a_k(\xi_n, \boldsymbol{\zeta}_j, \xi_i)}_{\text{precomputed}}.
\end{aligned}
\tag{7.11}
$$

Then we have to store $Q_a + Q_c$ order-3 tensors of size $N_N^{\boldsymbol{u}} \times N_N^{\boldsymbol{u}} \times N_N^{\boldsymbol{u}}$ and $N_N^{\boldsymbol{u}} \times N_N^T \times N_N^T$. It remains reasonable (in term of memory cost) and allows to totally ignore the finite element dimension during the assembly of the Jacobian $\boldsymbol{J}_N$.

The residual $\boldsymbol{R}_N(\boldsymbol{u}, p, T; \boldsymbol{\mu})$ can be expressed using the same submatrices, $\forall \boldsymbol{\mu} \in \mathcal{D}$

$$
\boldsymbol{R}_N(\boldsymbol{u}, p, T; \boldsymbol{\mu}) = \begin{bmatrix} \boldsymbol{D}_N^{\boldsymbol{u}}(\boldsymbol{\mu}) + \boldsymbol{C}_N(\boldsymbol{u};\boldsymbol{\mu}) & \boldsymbol{B}_N(\boldsymbol{\mu})^\intercal & -\boldsymbol{F}(\boldsymbol{\mu}) \\ -\boldsymbol{B}_N(\boldsymbol{\mu}) & & \\ & & \boldsymbol{A}_N(\boldsymbol{u};\boldsymbol{\mu}) + \boldsymbol{D}_N^T(\boldsymbol{\mu}) \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ p \\ T \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \boldsymbol{G}_N(\boldsymbol{\mu}) \end{bmatrix},
\tag{7.12}
$$

and so we can use the method described previously and the precomputed tensors to evaluate the residual $\boldsymbol{R}_N$ independently of the finite element dimension. The complete online non-linear resolution is summarized in algorithm 9.

Remark 11. *During the Newton algorithm, the blocks coming from the bilinear contributions are only assembled once. The blocks coming from the trilinear contributions have to reassembled for each new linear resolution.*

The convergence of the Newton algorithm during the online procedure was an issue for some problems. We had to consider some improvements to help the convergence. First, for any new parameter $\boldsymbol{\mu}$, we search the closest parameter (in the sense of the usual norm for vectors) in S_N. We note $\bar{\boldsymbol{\mu}}$ this reference parameter. Then we initialize the Newton algorithm using the projection of the solution $(\boldsymbol{u}(\bar{\boldsymbol{\mu}}), p(\bar{\boldsymbol{\mu}}), T(\bar{\boldsymbol{\mu}}))$. This projection has been evaluated and stored during the offline phase. This initialization greatly improves the convergence of the solver. However, it might not be enough. When this procedure is not sufficient, we add a continuation on the parameter. The online problem is solved for a sequence $(\boldsymbol{\mu}_k)_0^K$ such that

$\boldsymbol{\mu}_0 = \bar{\boldsymbol{\mu}}$ and $\boldsymbol{\mu}_K = \boldsymbol{\mu}$. This procedure is particularly costly, but it is only required for a few numbers of parameters and is not used if the direct resolution converges.

Algorithm 9: Newton Algorithm for the online non-linear resolution

Input: $\boldsymbol{\mu}$
Output: $(\boldsymbol{u}_N(\boldsymbol{\mu}), p_N(\boldsymbol{\mu}), T_N(\boldsymbol{\mu}))$
Choose δ_{tol} ; // Algorithm tolerance
Choose $(\boldsymbol{u}^0, p^0, T^0)$; // Initial guess
Assemble linear contribution $\boldsymbol{G}_N(\boldsymbol{\mu})$;
Assemble bilinear contributions $\boldsymbol{D}_N^{\boldsymbol{u}}$, $\boldsymbol{B}_N$, $\boldsymbol{F}_N$,$\boldsymbol{D}_N^T$;
repeat
  Assemble trilinear contributions: $\boldsymbol{C}_N(\boldsymbol{u}^{k-1}, \boldsymbol{\mu})$, $\boldsymbol{C}_N^*(\boldsymbol{u}^{k-1}, \boldsymbol{\mu})$;
  Assemble trilinear contributions: $\boldsymbol{A}(\boldsymbol{u}^{k-1}; \boldsymbol{\mu})$, $\boldsymbol{A}^*(T^{k-1}; \boldsymbol{\mu})$;
  Assemble $\boldsymbol{J}_N^{k-1} = \boldsymbol{J}_N(\boldsymbol{u}^{k-1}, p^{k-1}, T^{k-1}; \boldsymbol{\mu})$, see (7.8);
  Assemble $\boldsymbol{R}_N^{k-1} = \boldsymbol{R}(\boldsymbol{u}^{k-1}, p^{k-1}, T^{k-1}; \boldsymbol{\mu})$, see (7.12);
  Solve $\boldsymbol{J}_N^{k-1}\boldsymbol{H}^{k-1} = -\boldsymbol{R}_N^{k-1}$;
  $(\boldsymbol{u}^k, p^k, T^k) \leftarrow (\boldsymbol{u}^{k-1}, p^{k-1}, T^{k-1}) + \boldsymbol{H}^{k-1}$;
until $\|\boldsymbol{R}_N(\boldsymbol{u}^k, p^k, T^k; \boldsymbol{\mu})\|_2 \leq \delta_{tol}$;
return $(\boldsymbol{u}_N(\boldsymbol{\mu}), p_N(\boldsymbol{\mu}), T_N(\boldsymbol{\mu})) = (\boldsymbol{u}^k, p^k, T^k)$;

1.3 Treatment of the Stabilization Terms

We consider the system (2.2) and we add the stabilization terms. We obtain the new weak formulation: find $(\boldsymbol{u}_h(\boldsymbol{\mu}), p_h(\boldsymbol{\mu}), T_h(\boldsymbol{\mu})) \in V_h \times Q_h \times X_h$ such that

$$\begin{aligned} a_{\boldsymbol{u}}^*(\boldsymbol{u}_h(\boldsymbol{\mu}), p_h(\boldsymbol{\mu}), \boldsymbol{v}_h; \boldsymbol{\mu}) - b(\boldsymbol{v}_h, p_h; \boldsymbol{\mu}) + b(\boldsymbol{u}_h, q_h; \boldsymbol{\mu}) &= f_{\boldsymbol{u}}^*(T_h, \boldsymbol{v}_h; \boldsymbol{\mu}), \\ a_T^*(T_h, \boldsymbol{u}_h, S_h; \boldsymbol{\mu}) &= f_T^*(S_h; \boldsymbol{\mu}), \\ \forall (\boldsymbol{v}_h, q_h, S_h) \in V_h \times Q_h \times X_h, & \end{aligned} \tag{7.13}$$

whith

$$\begin{aligned} a_{\boldsymbol{u}}^*(\boldsymbol{u}_h, p_h, \boldsymbol{v}_h; \boldsymbol{\mu}) &= c(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{v}_h; \boldsymbol{\mu}) + d_u(\boldsymbol{u}_h, \boldsymbol{v}_h; \boldsymbol{\mu}) + a_{\boldsymbol{u}}^{\mathrm{SDM}}(\boldsymbol{u}_h, p_h, \boldsymbol{v}_h; \boldsymbol{u}_h, \boldsymbol{\mu}) \\ a_T^*(T_h, \boldsymbol{u}_h, S_h; \boldsymbol{\mu}) &= a(T_h, \boldsymbol{u}_h, S_h; \boldsymbol{\mu}) + d_T(T_h, S_h; \boldsymbol{\mu}) + a_T^{\mathrm{SDM}}(T_h, S_h; \boldsymbol{u}_h, \boldsymbol{\mu}) \\ f_{\boldsymbol{u}}^*(T_h, \boldsymbol{v}_h; \boldsymbol{\mu}) &= f(T_h, \boldsymbol{v}_h; \boldsymbol{\mu}) + f_{\boldsymbol{u}}^{\mathrm{SDM}}(T_h, \boldsymbol{v}_h; \boldsymbol{u}_h, \boldsymbol{\mu}) \\ f_T^*(S; \boldsymbol{\mu}) &= g(S_h; \boldsymbol{\mu}) + f_T^{\mathrm{SDM}}(T_h, S_h; \boldsymbol{u}_h, \boldsymbol{\mu}), \end{aligned} \tag{7.14}$$

where the forms c, d_u, a, d_T, f and g are defined in (7.3). The operators $a_{\boldsymbol{u}}^{\mathrm{SDM}}$, a_T^{SDM}, $f_{\boldsymbol{u}}^{\mathrm{SDM}}$ and f_T^{SDM} are the stabilization terms detailed in the chapter 3.

Those stabilization operators are non-linear and non-affine in parameters. By introducing these terms in the variational formulation, we can rewrite the Jacobian $\boldsymbol{J}_N$ and the residual $\boldsymbol{R}_N$ as

$$\begin{aligned} \boldsymbol{J}_N(\boldsymbol{u}, p, T; \boldsymbol{\mu}) &= \boldsymbol{J}_N^{\mathrm{tri}}(\boldsymbol{u}, p, T; \boldsymbol{\mu}) + \boldsymbol{J}_N^{\mathrm{SDM}}(\boldsymbol{u}, p, T; \boldsymbol{\mu}), \\ \boldsymbol{R}_N(\boldsymbol{u}, p, T; \boldsymbol{\mu}) &= \boldsymbol{R}_N^{\mathrm{tri}}(\boldsymbol{u}, p, T; \boldsymbol{\mu}) + \boldsymbol{R}_N^{\mathrm{SDM}}(\boldsymbol{u}, p, T; \boldsymbol{\mu}), \end{aligned} \tag{7.15}$$

where $\boldsymbol{J}_N^{\text{tri}}$ and $\boldsymbol{R}_N^{\text{tri}}$ are the contributions detailed respectively in (7.8) and (7.12).$\boldsymbol{J}_N^{\text{SDM}}$ and $\boldsymbol{R}_N^{\text{SDM}}$ are the contributions from the stabilization operators. Using the EIM for discrete operators introduced in the previous chapter, we can compute an affine approximation for these contributions,

$$\begin{aligned}\boldsymbol{J}_N^{\text{SDM}}(\boldsymbol{u},p,T;\boldsymbol{\mu}) &\approx \sum_{m=1}^{M^J}\theta_m^J(\boldsymbol{u},p,T,\boldsymbol{\mu})\boldsymbol{J}_{m,N}\\ \boldsymbol{R}_N^{\text{SDM}}(\boldsymbol{u},p,T;\boldsymbol{\mu}) &\approx \sum_{m=1}^{M^R}\theta_m^R(\boldsymbol{u},p,T,\boldsymbol{\mu})\boldsymbol{R}_{m,N}\end{aligned} \tag{7.16}$$

where the matrices $\boldsymbol{J}_{m,N}$ and the vectors $\boldsymbol{R}_{m,N}$ are pre-computed during the offline procedure. The online phase then consists in the evaluation of the $\theta_m^J(\boldsymbol{u},p,T,\boldsymbol{\mu})$ and $\theta_m^R(\boldsymbol{u},p,T,\boldsymbol{\mu})$ using the EIM algorithm. This allows to compute the Jacobian matrix and the Residual vector independently of the FE dimension.

Remark 12. *With very basic stabilization parameters we might retrieve a quadratic non-linearity for the stabilization terms. However, it is a particular case, and we will not detail this point. In such a situation, the stabilization terms can be integrated into the trilinear contribution of the aerothermal system.*

1.4 Reduction of a Turbulent Model

We had some difficulties with turbulent simulations and especially with the stabilization of the turbulent models. The final development of the full turbulent model is very recent, and then we had no time to start the implementation of the reduced version. However, we will present in this section how we planned to apply the RBM to RANS simulations. The presented method is only a theory and was not implemented.

We assume here that we have a fully operational turbulence model. This model allows to compute the turbulent viscosity $\nu_t(\boldsymbol{u},\boldsymbol{\mu})$. In RANS models, this turbulent viscosity is usually computed from scalar quantities, solutions of one (or several) PDE(s). For instance, the Spalart Allmaras model and the $k-\omega$ SST models presented in the chapter 4 require the resolution of one and two equations respectively.

A natural idea is to use the EIM to build an affine approximation of the turbulent viscosity $\nu_t(\boldsymbol{u},\boldsymbol{\mu})$. However, the EIM requires to know for any new $\boldsymbol{u}$ and $\boldsymbol{\mu}$, the exact value of $\nu_t(\boldsymbol{u},\boldsymbol{\mu})$ on the interpolation points. To obtain theses values, we have to build reduced basis spaces for the scalar quantities of the turbulence model ($\bar{\nu}(\boldsymbol{u},\boldsymbol{\mu})$ for Spalart Allmaras or $(k(\boldsymbol{u},\boldsymbol{\mu}),\omega(\boldsymbol{u},\boldsymbol{\mu}))$ for $k-\omega$ SST). To simplify the explanations in the following paragraphs, we suppose now that the viscosity ν_t is computed from one generic scalar quantity $\bar{\nu}(\boldsymbol{u},\boldsymbol{\mu})$, solution of one PDE. The following method is easily adaptable in the case of a model with multiple PDE.

In most of the RANS models, the scalar field $\bar{\nu}(\boldsymbol{u}, \boldsymbol{\mu})$ is expressed as the solution of a non-linear transport equation on the domain Ω. We will solve this PDE in a adapted FE space $P_h(\Omega)$. The generic weak formulation for this problem can be written as find $\bar{\nu}(\boldsymbol{u}, \boldsymbol{\mu}) \in P_h$, $\forall \boldsymbol{u}_h \in V_h$, $\forall \boldsymbol{\mu} \in \mathcal{D}$, such that

$$c_t(\bar{\nu}_h, \gamma_h, \boldsymbol{u}_h; \boldsymbol{\mu}) + a_t(\bar{\nu}_h, \gamma_h; \boldsymbol{\mu}, \boldsymbol{u}_h) = f_t(\gamma_h), \quad \forall \gamma_h \in P_h, \tag{7.17}$$

where, $\forall \boldsymbol{\mu} \in \mathcal{D}$

- $c_t : P_h \times P_h \times V_h \to \mathbb{R}$ is a continuous trilinear form,
- $a_t : P_h \times P_h \to \mathbb{R}$ is a continuous bilinear form,
- $f_t : P_h \to \mathbb{R}$ is a continuous linear form.

In practice, a_t is usually not bilinear and we suppose here that we use the EIM to obtain an affine approximation, $\forall \boldsymbol{\mu} \in \mathcal{D}$, $\forall \boldsymbol{u}_h \in V_h$

$$\begin{aligned} a_t(\bar{\nu}_h, \gamma_h; \boldsymbol{\mu}, \boldsymbol{u}_h, \bar{\nu}_h) &= \sum_{k=1}^{Q_{a_t}} \theta_k^{a_t}(\boldsymbol{\mu}, \boldsymbol{u}_h, \bar{\nu}_h) a_{t,k}(\bar{\nu}_h, \gamma_h) \\ f_t(\gamma_h; \boldsymbol{\mu}) &= \sum_{k=1}^{Q_{f_t}} \theta_k^{f_t}(\boldsymbol{\mu}) f_{t,k}(\gamma_h). \end{aligned} \tag{7.18}$$

We also suppose that the convection term c_t admits an affine decomposition

$$c_t(\bar{\nu}_h, \gamma_h, \boldsymbol{u}_h; \boldsymbol{\mu}) = \sum_{k=1}^{Q_{c_t}} \theta_k^{c_t}(\boldsymbol{\mu}) a_{t,k}(\bar{\nu}_h, \gamma_h, \boldsymbol{u}_h). \tag{7.19}$$

During the offline procedure, the reduced spaces V_N, Q_N, X_N are built as described in the previous section. We also construct the space P_N,

$$P_N = \text{span}\{\bar{\nu}_h(\boldsymbol{u}_h(\boldsymbol{\mu}_n), \boldsymbol{\mu}_n), 1 \leq n \leq N\} = \text{span}\{\psi_n, 1 \leq n \leq N\}, \tag{7.20}$$

where $(\psi_n)_{n=1}^N$ is an orthonormalized basis of P_N. During this phase, we will also compute the quantities

$$\begin{aligned} c_{t,k}(\psi_i, \psi_j, \boldsymbol{\zeta}_l), &\quad \forall 1 \leq i, j \leq N, \quad \forall 1 \leq l \leq N^{\boldsymbol{u}}, \\ a_{t,k}(\psi_i, \psi_j), &\quad \forall 1 \leq i, j \leq N, \\ f_{t,k}(\psi_i), &\quad \forall 1 \leq i \leq N. \end{aligned} \tag{7.21}$$

Then, during the online phase, we can evaluate the RB approximation $\bar{\nu}_N(\boldsymbol{u}_N, \boldsymbol{\mu})$, $\forall \boldsymbol{u}_N \in \mathbb{R}^{N^{\boldsymbol{u}}}$, $\forall \boldsymbol{\mu} \in \mathcal{D}$ as the solution of the reduced non-linear system

$$(\boldsymbol{C}_t(\boldsymbol{u}_N, \boldsymbol{\mu}) + \boldsymbol{A}(\boldsymbol{\mu}, \boldsymbol{u}_h, \bar{\nu}_N))\bar{\nu}_N = \boldsymbol{F}_t(\boldsymbol{\mu}), \tag{7.22}$$

with

$$
\begin{aligned}
(\boldsymbol{C}_t(\boldsymbol{u}_N, \boldsymbol{\mu}))_{i,j} &= \sum_{k=1}^{Q_{c_t}} \sum_{l=1}^{N^{\boldsymbol{u}}} \theta_k^{c_t}(\boldsymbol{\mu}) c_{t,k}(\psi_i, \psi_j, \boldsymbol{\zeta}_l), \\
(\boldsymbol{A}_t(\boldsymbol{\mu}, \boldsymbol{u}_N, \bar{\nu}_N))_{i,j} &= \sum_{k=1}^{Q_{a_t}} \theta_k^{a_t}(\boldsymbol{\mu}, \boldsymbol{u}_N, \bar{\nu}_N) a_{t,k}(\psi_i, \psi_j) \\
(\boldsymbol{F}_t(\boldsymbol{\mu}))_i &= \sum_{k=1}^{Q_{f_t}} \theta_k^{f_t}(\boldsymbol{\mu}) f_{t,k}(\psi_i).
\end{aligned}
\tag{7.23}
$$

Once this system is solved, the RB approximation $\bar{\nu}_N(\boldsymbol{u}_N, \boldsymbol{\mu})$ can be used to evaluate the turbulent viscosity $\nu_t(\boldsymbol{u}_N, \boldsymbol{\mu})$. It is then possible to iterate between the two reduced models with classical fixed point iterations.

The method described might theoretically work. However, it implies the use of multiple EIMs. The resolution of the turbulent PDE is usually not trivial and very sensitive. There is no guaranty that the many approximations made through the EIMs provide a stable solution.

2 OPUS Test-Case: Application to the Cooling of Electronic Components

We present in this section our numerical results on a test case proposed in 2009 by Michel Fouquembergh and Annabelle Le-Hyaric, both from Airbus Group.

2.1 Problem Description

We consider a 2D model representative of the neighboring of an electronic component submitted to a cooling air flow. It is described by four geometrical domains in $\mathbb{R}^2$ named Ω_i, $i = 1, 2, 3, 4$, see figure 7.1. We then want to compute the temperature field T, in the whole domain $\Omega = \cup_{i=1}^4 \Omega_i$, and the velocity/pressure $(\boldsymbol{u}, p)$ in the sub-domain Ω_4, as the solution of the aerothermal system of equations

$$
\begin{aligned}
\rho \boldsymbol{u} \cdot \nabla \boldsymbol{u} - 2\nabla \cdot (\mu \boldsymbol{S}(\boldsymbol{u})) + \nabla p &= 0, && \text{on } \Omega_4 \\
\nabla \cdot \boldsymbol{u} &= 0, && \text{on } \Omega_4 \\
\rho_i C_i \boldsymbol{v} \cdot \nabla T - \nabla \cdot (k_i \nabla T) &= Q_i, && \text{on } \Omega_i, \quad i = 1, 2, 3, 4
\end{aligned}
\tag{7.24}
$$

where, in each subdomain Ω_i, k_i is the thermal diffusivity and Q_i is a volumic heat dissipated. ρ and μ are respectively the density and the viscosity of the air. We do not consider natural convection in this case and so the coupling between the fluid and the thermal equations is only one way.

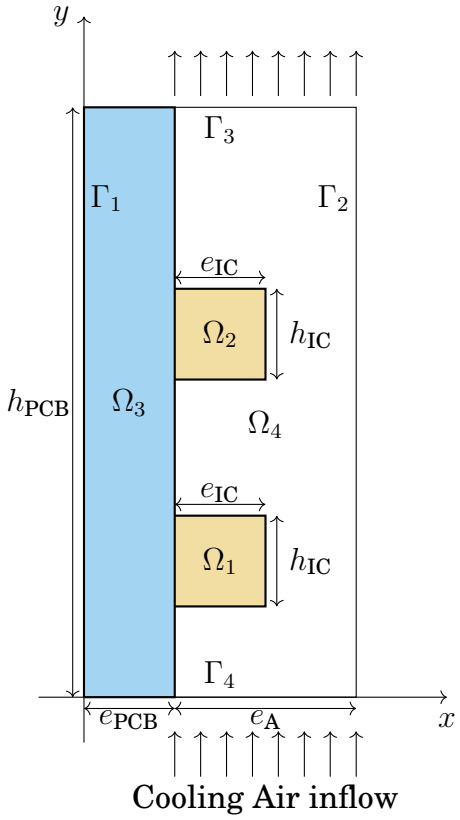


Figure 7.1 – Geometry of the problem

The subdomains Ω_1 and Ω_2 define two Integrated Circuits, respectively IC1 and IC2, simplified as two rectangles of height h_{IC} and width e_{IC}. These IC are soldered to the Printed Circuit Board (PCB) which is a rectangle of height h_{PCB} and width e_{PCB}. The Air is flowing along the PCB in the domain Ω_4, of width e_{A}.

The notation for the external boundaries of the domain Ω:

$$\begin{aligned}
\Gamma_1 &= \{ \quad x = 0, \quad 0 \leq y \leq h_{\text{PCB}}\}, \\
\Gamma_2 &= \{ \quad x = e_{\text{PCB}} + e_{\text{A}}, \quad 0 \leq y \leq h_{\text{PCB}}\}, \\
\Gamma_3 &= \{ \quad 0 \leq x \leq e_{\text{PCB}} + e_{\text{A}}, \quad y = h_{\text{PCB}}\}, \\
\Gamma_4 &= \{ \quad 0 \leq x \leq e_{\text{PCB}} + e_{\text{A}}, \quad y = 0\},
\end{aligned} \tag{7.25}$$

we also define the convenient notations: $\Gamma_I = (\Omega_3 \cup \Omega_2 \cup \Omega_1) \cap \Omega_4$ for the internal boundary between fluid and solid domains and $\Gamma_{ij} = \Gamma_i \cup \Omega_j$ for the sub-boundaries.

Boundary Conditions on the Temperature

- on $\Gamma_1 \cup \Gamma_2 \cup \Gamma_3$, a zero flux condition (Neumann)

$$-k\nabla T \cdot \boldsymbol{n} = 0, \tag{7.26}$$

- on Γ_4, imposed temperature (Dirichlet)

$$T = T_0; \tag{7.27}$$

- on all the internal boundaries, the continuity of the heat flux and temperature

$$T_i = T_j, \qquad k_i \nabla T \cdot \boldsymbol{n} = -k_j \nabla T \cdot \boldsymbol{n} \tag{7.28}$$

Boundary Conditions on the Fluid

- on Γ_3, a free stream condition

$$\mu \nabla \boldsymbol{u} \cdot \boldsymbol{n} - p\boldsymbol{n} = 0 \tag{7.29}$$

- on $\Gamma_2 \cup \Gamma_I$, a no-slip wall condition (Homogeneous Dirichlet)

$$\boldsymbol{u} = 0 \tag{7.30}$$

- on Γ_4, a Poiseuille Profile (Dirichlet)

$$\boldsymbol{u} = \boldsymbol{u}_P = \left(0, \frac{3}{2(e_{\text{A}} - e_{\text{IC}})} D \left(1 - \left(\frac{x - (\frac{e_{\text{A}}+e_{\text{IC}}}{2} + e_{\text{PCB}})}{\frac{e_{\text{A}}-e_{\text{IC}}}{2}} \right)^2 \right) \right) \tag{7.31}$$

Modifications of the Original Model

The boundary conditions presented above are slightly simplified compared to the conditions proposed in the original test case.

- The conditions of the temperature on Γ_1 and Γ_2 are supposed to be periodic conditions

$$\begin{aligned} T|_{x=0} &= T|_{x=e_{\text{PCB}}+e_{\text{A}}} \\ k_3 \nabla T|_{x=0} \cdot \boldsymbol{n}_3 &= -k_4 \nabla T|_{x=e_{\text{PCB}}+e_{\text{A}}} \cdot \boldsymbol{n}_4 \end{aligned} \tag{7.32}$$

- The conditions at the interfaces between ICi and the PCB are supposed to be a thermal contact conductance

$$-k_i \nabla T \cdot \boldsymbol{n}_i - k_3 \nabla T \cdot \boldsymbol{n}_3 = r_{i3}(T_{\partial\Omega_i} - T_{\partial\Omega_3}) \tag{7.33}$$

  where r_{i3} is a positive coefficient.

These conditions have been simplified into the respective conditions (7.26) and (7.28) because these kinds of conditions are not supported in Feel++.

Inputs of the Model

We present in table 7.1 the different parameters of this case. We indicate a nominal value and eventually a range. The nominal value is the default. The variables with a range are the parameters of the model.

Name	Description	Nominal Value	Range	Units
	IC			
$\rho_1 C_1 = \rho_2 C_2$	Heat Capacity	$1.4 \cdot 10^6$		$\mathrm{J.m^{-1}.K^{-1}}$
Q_1, Q_2	Heat source	10^6	$[0, 10^6]$	$\mathrm{W.m^{-3}}$
k_1, k_2	Thermal conductivity	2	$[0.2, 150]$	$\mathrm{W.m^{-1}.K^{-1}}$
e_{IC}	Thickness	$2 \cdot 10^{-3}$		m
h_{IC}	Height	$70 \cdot 10^{-3}$		m
	PCB			
$\rho_3 C_3$	Heat Capacity	$2 \cdot 10^6$		$\mathrm{J.m^{-1}.K^{-1}}$
Q_3	Heat source	0		$\mathrm{W.m^{-3}}$
k_3	Thermal conductivity	0.2		$\mathrm{W.m^{-1}.K^{-1}}$
e_{PCB}	Thickness	$2 \cdot 10^{-3}$		m
h_{PCB}	Height	$130 \cdot 10^{-3}$		m
	Air			
$\rho_4 C_4$	Heat Capacity	1100		$\mathrm{J.m^{-1}.K^{-1}}$
Q_4	Heat source	0		$\mathrm{W.m^{-3}}$
k_4	Thermal conductivity	$3 \cdot 10^{-2}$		$\mathrm{W.m^{-1}.K^{-1}}$
ρ	Density	1.0		$\mathrm{kg.m^{-3}}$
μ	Viscosity	$1.8 \cdot 10^{-5}$		$\mathrm{kg.m^{-1}.s^{-1}}$
e_{A}	Thickness	$14 \cdot 10^{-3}$		m
T_0	Inflow temperature	300		K
D	Inflow rate	7	$[0.5, 50]$	$10^{-3}\ \mathrm{m^2.s^{-1}}$

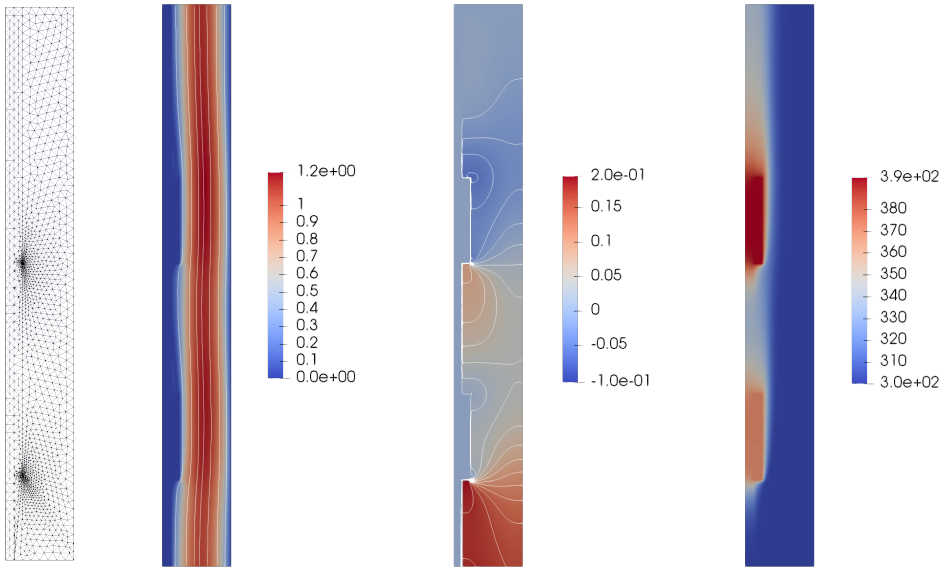
Table 7.1 – Opus Application: Parameters of the model

Remark 13. *A geometric parameter was initially introduced in the benchmark configuration. The thickness e_A was supposed to vary. It appeared that some shapes produced recirculations at the outlet and then the Newton solver was not able to converge anymore. We are still working on this issue, and we hope to propose the results with geometric parameter very soon.*

Output of the Model

We compute, $\forall \boldsymbol{\mu} \in \mathcal{D}$ the mean temperature on the second IC,

$$s(\boldsymbol{\mu}) = \int_{\Omega_2} T(\boldsymbol{\mu})\, d\Omega\ . \tag{7.34}$$



(a) Mesh (b) Velocity (c) Pressure (d) Temperature

Figure 7.2 – Opus Application: mesh and example of solution profiles (RB) for parameter $\boldsymbol{\mu} = (150, 150, 10^6, 10^6, 10^{-2})$. (a) coarser mesh used for the stabilized simulations, (b) velocity profile, in $\mathrm{m.s}^{-1}$, with streamlines, (c) pressure field, in Pa, with contour lines, (d) temperature field, in K

2.2 Numerical Results

The following results have been produced on a non-structured simplex mesh. The mesh is presented in figure 7.2 with an example of profiles for the temperature, velocity and the pressure ($\boldsymbol{\mu} = (150, 150, 10^6, 10^6, 10^{-2})$). The FE space is $\mathbb{TH}^1_{ch}(\Omega) \times \mathbb{L}^2_{ch}(\Omega)$.

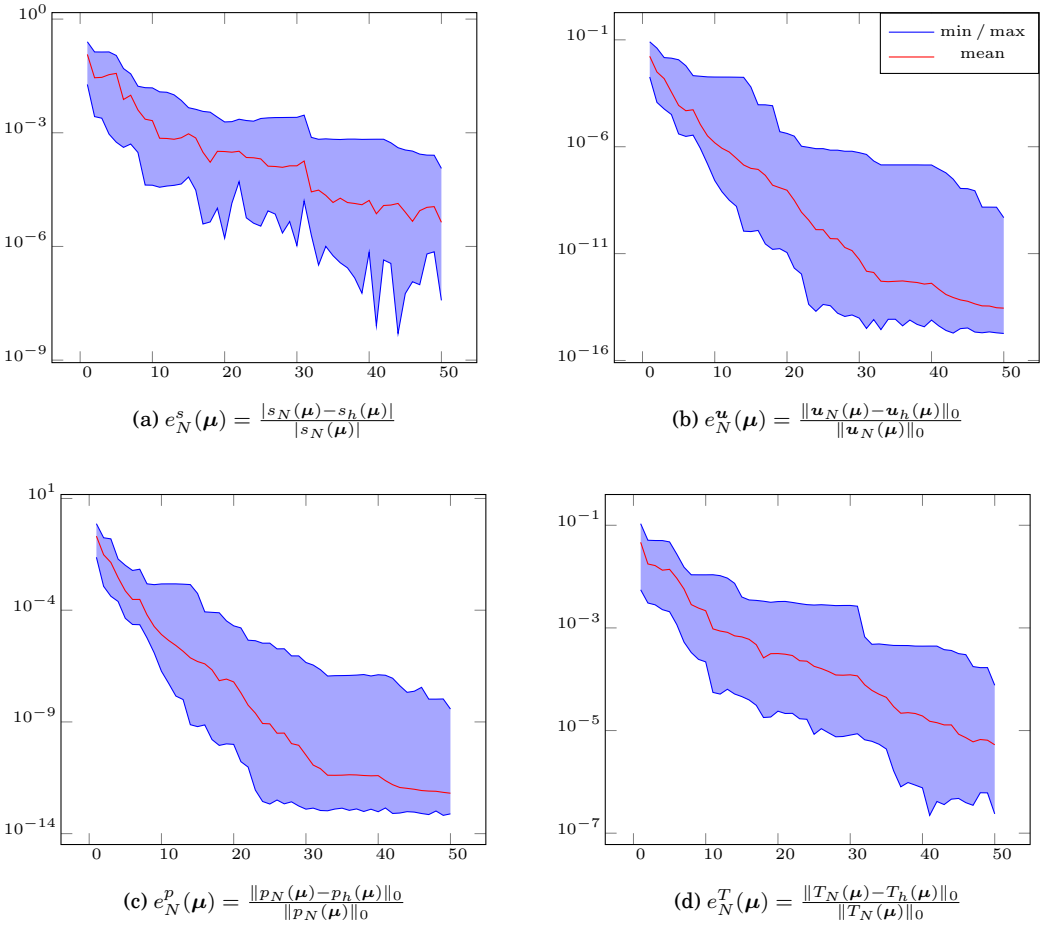
No Stabilization

We first present some results with no stabilization. In order to keep a stable solution, we do two small modifications. First, the mesh has been refined by splitting all the cells in the mesh presented in figure 7.2. Then, the range for the inflow rate has been reduced, and we consider for now $D \in [0.5 \cdot 10^{-3}, 10 \cdot 10^{-3}]\mathrm{m}^2.\mathrm{s}^{-1}$.

We built the reduced basis using $\delta_{tol} = 10^{-12}$ in the selective Gram-Schmidt algorithm 7, with 50 parameters in S_N. At the end of the offline procedure, the size of the spaces $V_N \times Q_N \times X_N$ is $89 \times 38 \times 50$. We present the convergence of the errors for s, $\boldsymbol{u}$, p and T on the figure 7.3. These results are very satisfying, considering the dimension of the parameter space. Of course, the convergence of the errors can still be improved by using an adapted error estimator in the greedy algorithm.

As expected, the time for the resolution of the reduced problem increases with the size of the reduced spaces. We present in figure 7.4 the mean runtime for

the reduced problem vs. the size N of S_N. To explain these runtimes, we also plotted (figure 7.5) the execution times for the main components of the resolution process. We observe that the assemblies of the reduced Jacobian and Residual become very expensive when N is growing. However, we figured out that the time for these operations depends on the dimension N but also on the max dimension $N_{\max}$. Indeed, if we compare the assembly times for different values of $N_{\max}$, see figure 7.6, we see a significant difference. This gap is probably the consequence of the block extraction methods used to assemble the reduced operators. The reduced matrices and vectors are stored using data structures from the C++ library Eigen. This library is very convenient for the extraction and the operations on submatrices. However, these extraction methods seem to depend on the size of the global matrix. This issue should be investigated in future works.



(a) $e_N^s(\boldsymbol{\mu}) = \frac{|s_N(\boldsymbol{\mu}) - s_h(\boldsymbol{\mu})|}{|s_N(\boldsymbol{\mu})|}$

(b) $e_N^{\boldsymbol{u}}(\boldsymbol{\mu}) = \frac{\|\boldsymbol{u}_N(\boldsymbol{\mu}) - \boldsymbol{u}_h(\boldsymbol{\mu})\|_0}{\|\boldsymbol{u}_N(\boldsymbol{\mu})\|_0}$

(c) $e_N^p(\boldsymbol{\mu}) = \frac{\|p_N(\boldsymbol{\mu}) - p_h(\boldsymbol{\mu})\|_0}{\|p_N(\boldsymbol{\mu})\|_0}$

(d) $e_N^T(\boldsymbol{\mu}) = \frac{\|T_N(\boldsymbol{\mu}) - T_h(\boldsymbol{\mu})\|_0}{\|T_N(\boldsymbol{\mu})\|_0}$

Figure 7.3 – Opus Application without Stabilization: convergence of the errors vs the size of N. Maximum minimum and average errors evaluated from 50 approximations. $\delta_{tol} = 10^{-10}$ in selective Gram-Schmidt algorithm 7, maximum size of the reduced spaces $89 \times 38 \times 50$

These first results with our RB aerothermal solver are very encouraging. However, the geometry was very basic, and thus there were no significant variations in the fluid flow. We are now interested in the validation of our solver on more complex problems and also for simulations with stabilization methods.

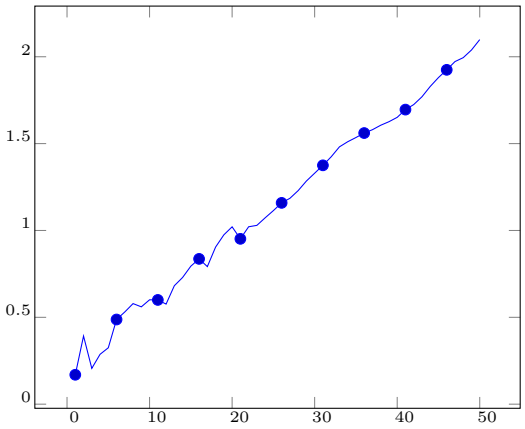


Figure 7.4 – Opus Application without Stabilization: average resolution time (in s) for the reduced problem vs N.

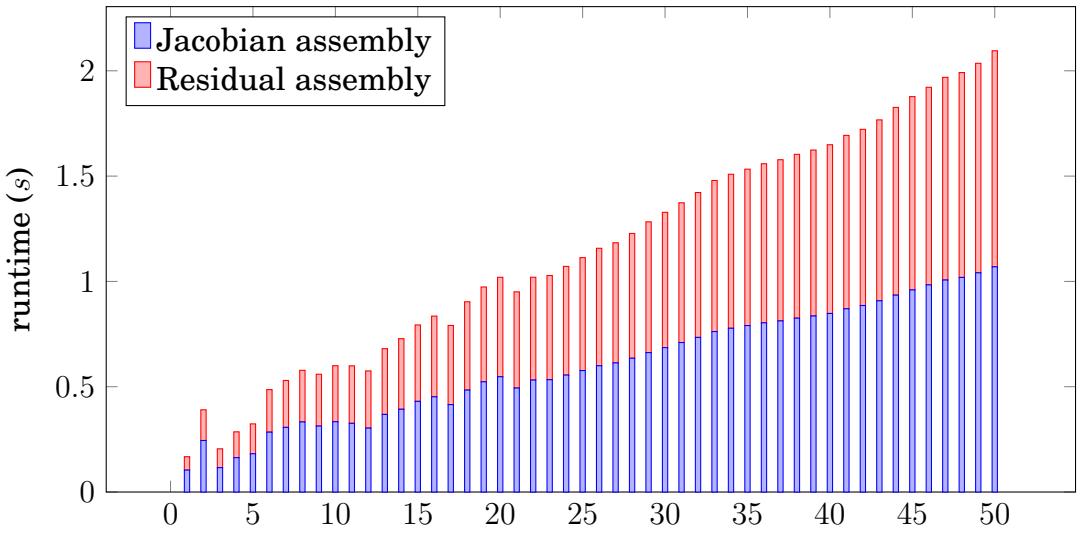


Figure 7.5 – Opus Application without Stabilization: Runtime of the main phases for the online resolution. Mean value for 50 runs. Other contributions are negligible

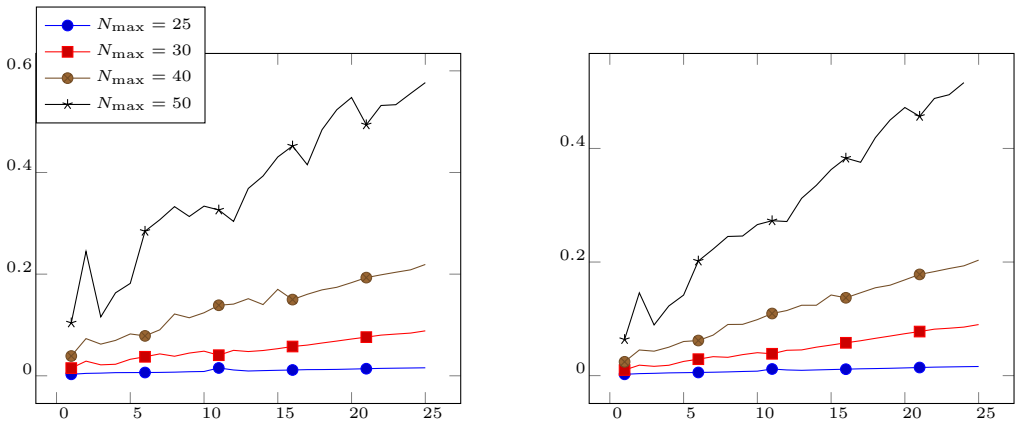


Figure 7.6 – Opus Application without Stabilization: assembly time for the reduced Jacobian (a) and the reduced Residual (b) for different values of $N_{\max}$. Mean values evaluated over 50 reduced resolutions.

Results with Stabilization

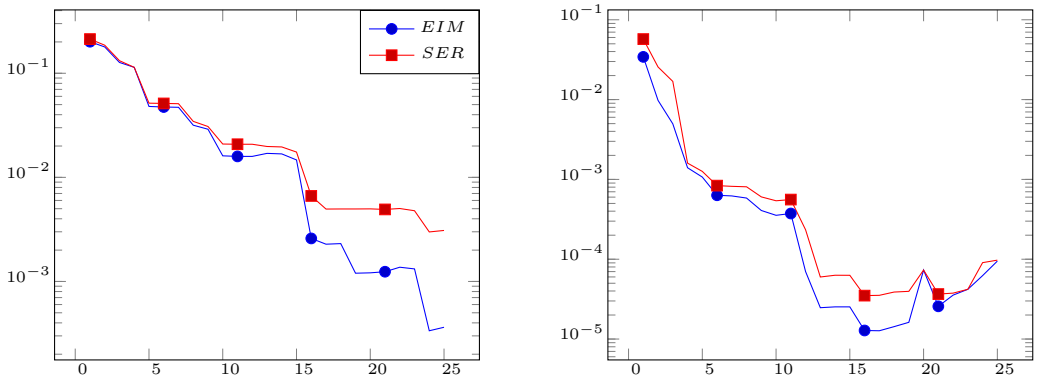
We now introduce stabilization operators. For this test case we choose the formulation GLS_1 for both the fluid and the temperature, see chapter3. We use the non-refined mesh presented in figure 7.2 and the full range for the inflow rate, $D \in [0.5 \cdot 10^{-3}, 50 \cdot 10^{-3}]$ $\text{m}^2.\text{s}^{-1}$. The tolerance in the selective Gram-Schmidt algorithm is set to $\delta_{tol} = 10^{-10}$.

There are two possibilities for the treatment of the stabilization. We can either use an EIM for the stabilization of the whole system or two EIMs, one for each physic. We initially wanted to compare the results of both versions. However, it appears that the results with the monolithic construction of the EIM approximation was not convincing and we finally abandoned this idea. It seems necessary to build one EIM for each physic.

We present on figure 7.7 a comparison to the errors for classical EIM and SER construction, both with the same offline sampling and the same EIM size $M = 25$. The convergence of SER is very satisfying in comparison of the classic EIM. It confirms the preliminary results of the chapter 6. We also propose a plot of the minimal, maximal and averaged values of the errors for the construction using SER on figure 7.8. Regarding the convergence of the errors for the velocity and the pressure fields, we observe a deterioration of the results after $N = 16$. For now, we have no explanation for this phenomena, and we are still investigating this issue.

With these simulations, we highlighted another problem with our implementation of the discrete EIM. The computation of the EIM coefficients is relatively expensive and is limiting the performance of the reduced model. Even if the assembly is made on a minimal mesh, it requires between 0.1 and $0.3s$ per assembly. With the four EIM of this model and the non-linear iterations, the time spent in computing the non-linear coefficients becomes particularly expensive. To illustrate this issue, we present on figure 7.9 the runtimes of the principal components of the online resolution: the assembly of the Jacobian, the assembly of the Residual and the evaluation of the nonlinear coefficients. This excessive computation time, in addition to the runtime issue already exposed with the previous test-case, became a problem for the study of simulations with stabilization. We started thinking about some optimizations to fix this issue but it requires some implementation efforts and the modification could not be made in the context of this thesis.

Even if the numerical results are encouraging for this test-case, it remains some work to do in order to be completely satisfied. We have to identify the cause of the bad accuracy for the pressure and the velocity fields and we still have to optimize the computation of the non-linear coefficients using EIM. We will now test our framework on more complex geometries with the following problem.



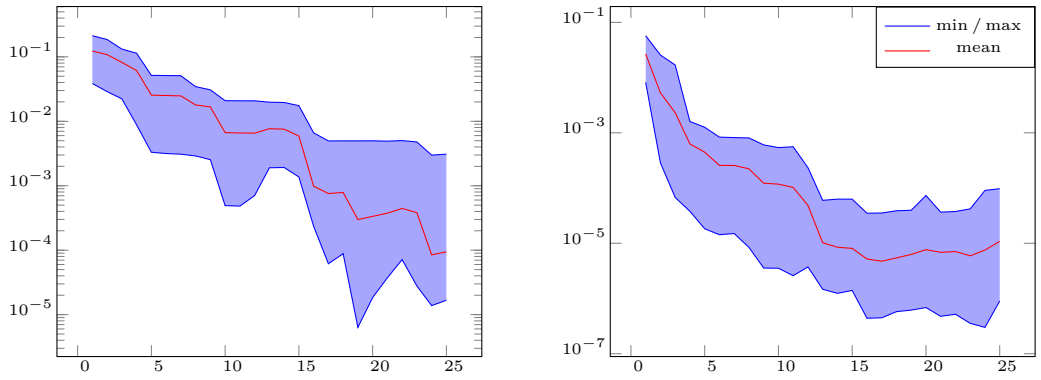
(a) $e_N^s(\boldsymbol{\mu}) = \frac{|s_N(\boldsymbol{\mu}) - s_h(\boldsymbol{\mu})|}{|s_N(\boldsymbol{\mu})|}$

(b) $e_N^{\boldsymbol{u}}(\boldsymbol{\mu}) = \frac{\|\boldsymbol{u}_N(\boldsymbol{\mu}) - \boldsymbol{u}_h(\boldsymbol{\mu})\|_0}{\|\boldsymbol{u}_N(\boldsymbol{\mu})\|_0}$

(c) $e_N^p(\boldsymbol{\mu}) = \frac{\|p_N(\boldsymbol{\mu}) - p_h(\boldsymbol{\mu})\|_0}{\|p_N(\boldsymbol{\mu})\|_0}$

(d) $e_N^T(\boldsymbol{\mu}) = \frac{\|T_N(\boldsymbol{\mu}) - T_h(\boldsymbol{\mu})\|_0}{\|T_N(\boldsymbol{\mu})\|_0}$

Figure 7.7 – Opus Application with Stabilization: convergence of the errors vs the size of N. Comparison of the maximum values for classical EIM and SER algorithm. Maximum size of the reduced spaces $50 \times 25 \times 25$



(a) $e_N^s(\boldsymbol{\mu}) = \frac{|s_N(\boldsymbol{\mu}) - s_h(\boldsymbol{\mu})|}{|s_N(\boldsymbol{\mu})|}$ (b) $e_N^{\boldsymbol{u}}(\boldsymbol{\mu}) = \frac{\|\boldsymbol{u}_N(\boldsymbol{\mu}) - \boldsymbol{u}_h(\boldsymbol{\mu})\|_0}{\|\boldsymbol{u}_N(\boldsymbol{\mu})\|_0}$

(c) $e_N^p(\boldsymbol{\mu}) = \frac{\|p_N(\boldsymbol{\mu}) - p_h(\boldsymbol{\mu})\|_0}{\|p_N(\boldsymbol{\mu})\|_0}$ (d) $e_N^T(\boldsymbol{\mu}) = \frac{\|T_N(\boldsymbol{\mu}) - T_h(\boldsymbol{\mu})\|_0}{\|T_N(\boldsymbol{\mu})\|_0}$

Figure 7.8 – Opus Application with Stabilization + SER: convergence of the errors vs N. Maximum minimum and average errors evaluated from 50 approximations. Maximum size of the reduced spaces: $50 \times 25 \times 25$. Number of basis vectors in EIM approximations $M = 25$

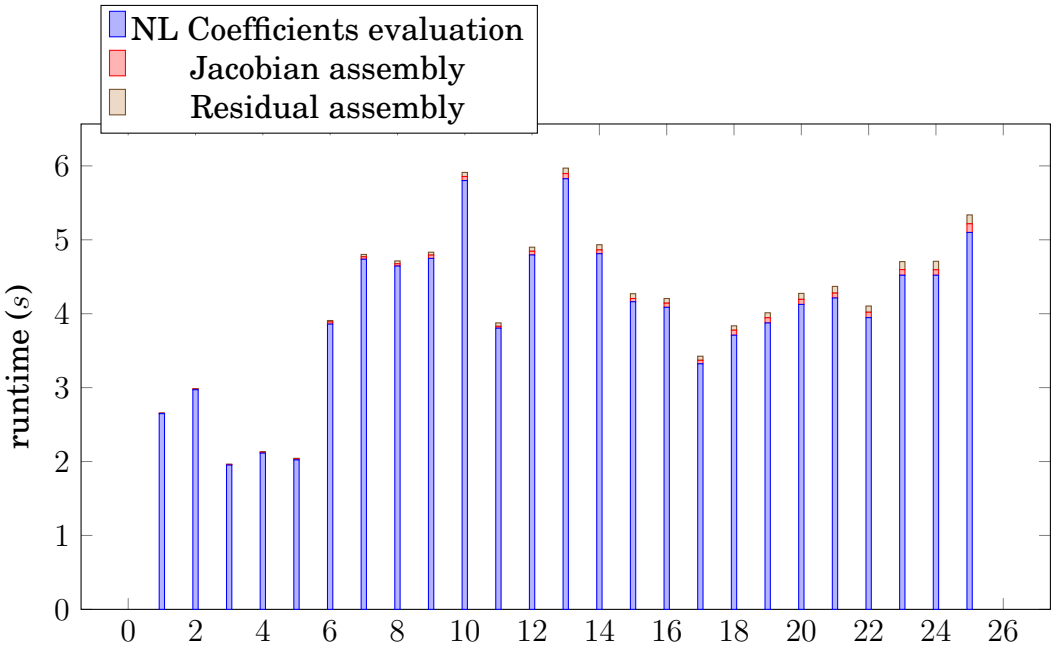


Figure 7.9 – Opus Application without Stabilization: Runtime of the main phases for the online resolution. Mean value for 50 runs. Other contributions are negligible

3 CHORUS Test-Case: Application to Aerothermal Simulations in an Airplane

We present in the test case proposed by Airbus Group in the context of the CHORUS project.

3.1 Problem Description

The geometry of the problem is a 2D slice of an aircraft, see 7.10. The top of the geometry is the Cabin Ω_C. In this part are six passengers P_i, $1 \leq i \leq 6$, one inlet I_C for the cooled air and two outlets O_{C1} and O_{C2}. The bottom part is the bay of the aircraft. It contains two electrical equipment E_1 and E_2, three bleed air ducts D_1, D_2 and D_3 and three electrical harnesses H_1, H_2 and H_3. There are also two inlets I_{B1} and I_{B2} and two outlets O_{B1} and O_{B2}. For any of these items X we denote by Γ_X its boundary. All the remaining boundaries are denoted by Γ_W. We finally note $\Omega = \Omega_C \cup \Omega_B$.

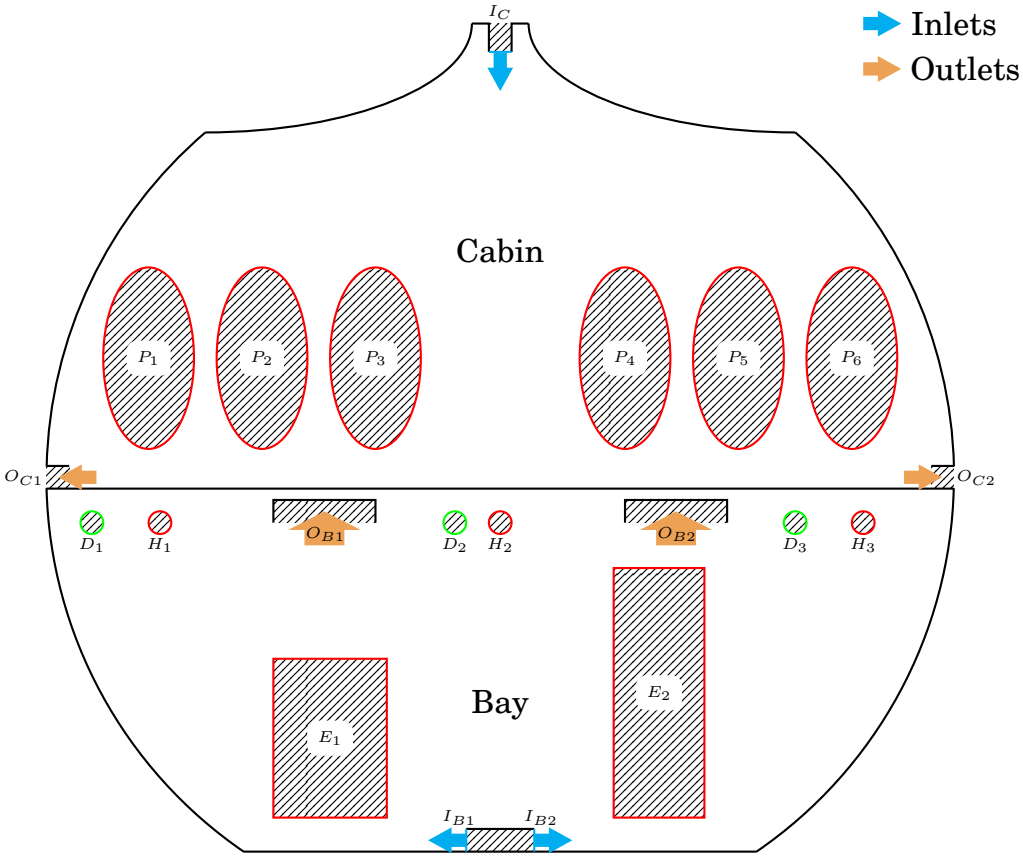


Figure 7.10 – Isometric 2D slice of the avionic bay

The aim of this test-case is the computation of the velocity field $\boldsymbol{u}$, the pressure

field p and the temperature field T, as solutions of the strong problem

$$\begin{aligned} \rho \boldsymbol{u} \cdot \nabla \boldsymbol{u} - 2\nabla \cdot (\mu \boldsymbol{S}(\boldsymbol{u})) + \nabla p &= -\rho\beta(T - T_0)\boldsymbol{g}, \\ \nabla \cdot \boldsymbol{u} &= 0, \\ \boldsymbol{v} \cdot \nabla T - \nabla \cdot (\kappa \nabla T) &= 0, \qquad \text{on } \Omega. \end{aligned} \tag{7.35}$$

The closing boundary conditions and the parameters of this system are detailed in the following paragraphs.

Boundary Conditions on the Temperature

- Passengers: on $\Gamma_P = \cup_{i=1}^{6} \Gamma_{P_i}$, a constant heat flux (Neumann condition)

$$-\kappa \nabla T \cdot \boldsymbol{n} = Q_{P_i}, \tag{7.36}$$

- Inlets: on $\Gamma_{I_k}, k = I_C, I_{B1}, I_{B2}$, a constant temperature (Dirichlet condition)

$$T = T_{I_k} \tag{7.37}$$

- Ducts: on $\Gamma_{D_i}, 1 \leq i \leq 3$, a constant temperature (Dirichlet condition)

$$T = T_{D_i} \tag{7.38}$$

- Harnesses: on $\Gamma_{H_i}, 1 \leq i \leq 3$, a constant flux (Neumann condition)

$$-\kappa \nabla T \cdot \boldsymbol{n} = Q_{H_i}, \tag{7.39}$$

- Equipment: on $\Gamma_{E_i}, 1 \leq i \leq 2$, a constant flux (Neumann condition), the same for both equipment

$$-\kappa \nabla T \cdot \boldsymbol{n} = Q_E, \tag{7.40}$$

- All the other solid walls are supposed to be insulated (Homogeneous Neumann condition)

$$\nabla T \cdot \boldsymbol{n} = 0 \tag{7.41}$$

Boundary Conditions on the Fluid

- Cabin Inlet: on Γ_{I_C}, a Poiseuille profile (Dirichlet)

$$\boldsymbol{u} = \boldsymbol{u}_{I_C} = (0, -D_{I_C} 600(0.05 - x)(x + 0.05)) \tag{7.42}$$

- Bay Inlets: on $\Gamma_{I_{B1}}$ and $\Gamma_{I_{B2}}$, a Poiseuille profile (Dirichlet)

$$\begin{aligned} \boldsymbol{u} = \boldsymbol{u}_{I_{B1}} &= -(D_{I_B} 600(y + 1.65)(-1.55 - y), 0), \quad \text{on } \Gamma_{I_{B1}} \\ \boldsymbol{u} = \boldsymbol{u}_{I_{B2}} &= (D_{I_B} 600(y + 1.65)(-1.55 - y), 0), \quad \text{on } \Gamma_{I_{B2}} \end{aligned} \tag{7.43}$$

- Outlets: on Γ_{Ok}, $k = C1, C2, B1, B2$, a free stream condition

$$\mu \nabla \boldsymbol{u} \cdot \boldsymbol{n} - p\boldsymbol{n} = 0 \tag{7.44}$$

- All the other solid walls have no-slip conditions (Homogeneous Dirichlet)

$$\boldsymbol{u} = (0, 0) \tag{7.45}$$

Modifications of the Original Model

The boundary conditions presented above are slightly simplified compared to the conditions proposed in the original test case.

- The floor between the two parts is not supposed to be insulated. We should model it as a solid part with a thickness of $0.05m$. This simplification allows decoupling the two resolutions. The modeling of the floor might be added in a future version of this application.

- The external walls of both the bay and the cabin are not supposed to be insulated. The walls should be modeled as solid parts with a thickness of $0.1m$. The outside temperature of the air is then a new parameter. This part might be added in a future version of the application.

- In the following numerical experiments, we will not consider the physical parameters of the air (μ and κ). Such parameters do not allow to reach a steady state without turbulence models. We will use larger viscosity and diffusivity to ensure the convergence of the solver. As soon as the turbulence models are fully operational, we will reconsider this simplification.

- The heat sources of the different passengers, harnesses and ducts are originally supposed to be independent. To keep the dimension of the parameter space reasonable, we suppose here that

$$\begin{aligned} T_{D_i} = T_{D_j} = T_D, &\quad \forall 1 \leq i, j \leq 3, \\ Q_{H_i} = Q_{H_j} = Q_H, &\quad \forall 1 \leq i, j \leq 3, \\ Q_{P_i} = Q_{P_j} = Q_P, &\quad \forall 1 \leq i, j \leq 6. \end{aligned} \tag{7.46}$$

Inputs of the Model

We present in table 7.2 the different parameters of this case. We indicate a nominal value and eventually a range. The nominal value is the default. The variables with a range are the parameters of the model.

Name	Description	Nominal Value	Range	Units
	Cabin			
Q_P	Passengers heat source	0.016	$[0, 0.016]$	$\mathrm{K.s^{-1}}$
T_{I_C}	Inflow temperature	288	$[273, 303]$	K
D_{I_C}	Inflow rate	0.5	$[0.001, 1]$	$\mathrm{m^2.s^{-1}}$
	Bay			
Q_H	Harnesses heat source	0.004	$[0, 0.0045]$	$\mathrm{K.s^{-1}}$
Q_E	Equipment heat source	0.05	$[0, 0.05]$	$\mathrm{K.s^{-1}}$
T_D	Ducts temperature	293	$[293, 373]$	K
T_{I_B}	Inflow temperature	288	$[273, 303]$	K
D_{I_B}	Inflow rate	0.5	$[0.001, 1]$	$\mathrm{m^2.s^{-1}}$
	Air			
κ	Thermal conductivity	$2.7 \cdot 10^{-3}$		$\mathrm{m^2.s^{-1}}$
ρ	Density	1.0		$\mathrm{kg.m^{-3}}$
μ	Viscosity	$1.8 \cdot 10^{-3}$		$\mathrm{kg.m^{-1}.s^{-1}}$

Table 7.2 – Chorus Application: Parameters of the model

Output of the Model

The original model was described with 15 quantities of interest. We chose here to focus on two of them: the mean temperature of the cabin and the mean temperature of the walls of the bay,

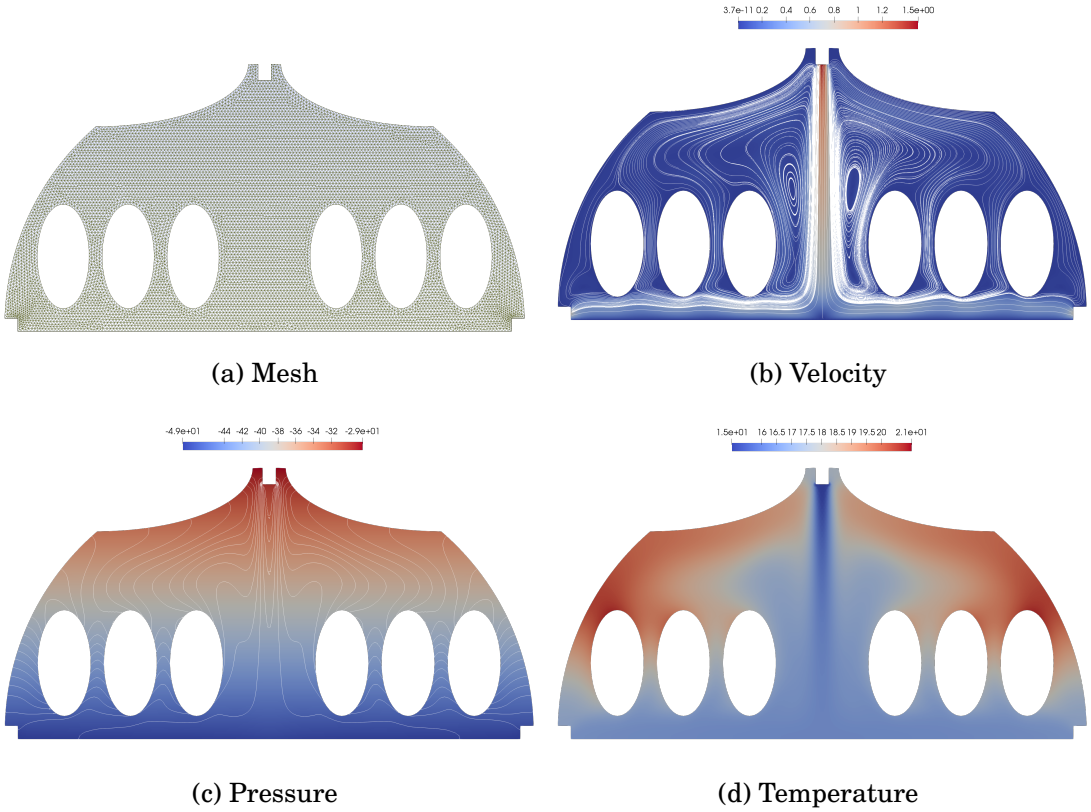
$$\begin{aligned} s_C(\boldsymbol{\mu}) &= \int_{\Omega_C} T(\boldsymbol{\mu})\, d\Omega \\ s_B(\boldsymbol{\mu}) &= \int_{\Gamma_{W_B}} T(\boldsymbol{\mu})\, d\Gamma \end{aligned} \tag{7.47}$$

3.2 Numerical Results

The following results have been produced on non-structured simplex meshes. The FE space is the same for both the cabin and the bay, $\mathbb{TH}^1_{ch}(\Omega) \times \mathbb{L}^2_{ch}(\Omega)$. The physical parameters presented previously guarantee to find a stationary state for the problem. It also ensures the stability of the numerical solution with a reasonable mesh. For that reason, we do not need to consider the stabilization operators for this test-case.

Results on the Cabin

The mesh used for the simulation on the cabin part, and an example of solution profiles are proposed in figure 7.11. We built the reduced basis using $\delta_{tol} = 10^{-10}$ in the selective Gram-Schmidt algorithm 7, with 50 parameters in S_N. At the end of



(a) Mesh (b) Velocity

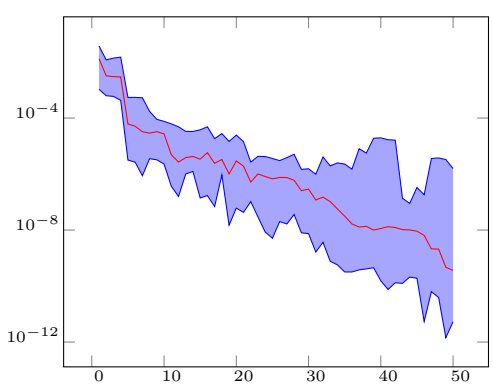
(c) Pressure (d) Temperature

Figure 7.11 – Chorus Application, Cabin Part: (a) meshes and example of solution profiles ($\boldsymbol{\mu} = (1.6 \cdot 10^{-2}, 288, 1)$) (b) velocity profile, in $\mathrm{m.s}^{-1}$, with streamlines, (c) pressure field, in Pa, with contour lines, (d) temperature field, in $^{\circ}\mathrm{C}$

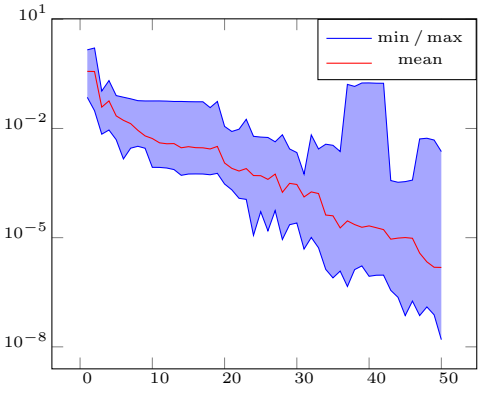
the offline procedure, the size of the spaces $V_N \times Q_N \times X_N$ is $97 \times 47 \times 50$. We present the convergence of the errors for s, $\boldsymbol{u}$, p and T on the figure 7.12.

We observe a small deterioration of the results (for the max value) after $N = 35$. We have no explanation for these error pikes however these inaccurate results only happened for one parameter value. This kind of wrong approximation would probably disappear with a suitable error estimator and a thinner super sampling Ξ. Regarding the mean value of the errors, the results are very satisfying.

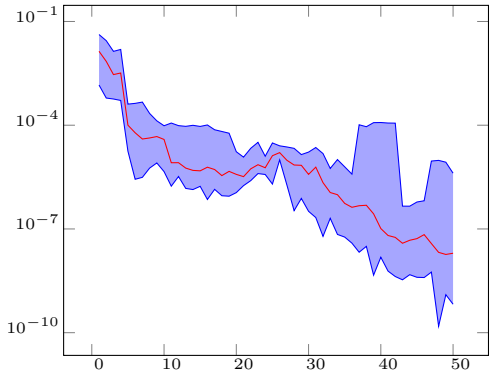
Again the resolution time for the online resolution grows very fast with N. For this test case, we also observed that the number of iterations in the non-linear solver is more consequent and fluctuating. We present the mean resolution times and number of iterations in figure 7.13 for $N_{\max} = 50$ and $N_{\max} = 25$ as a comparison. We observe the same gap between the runtime for $N_{\max} = 25$ and $N_{\max} = 50$, but the number of iterations is broadly the same. The evolution of the time required for the assemblies of the reduced Jacobian and Residual is also presented on figure 7.14. Considering the fluctuating number of iterations in the non-linear solver, we decided to plot the mean value per iteration to have a better idea of the actual evolution.



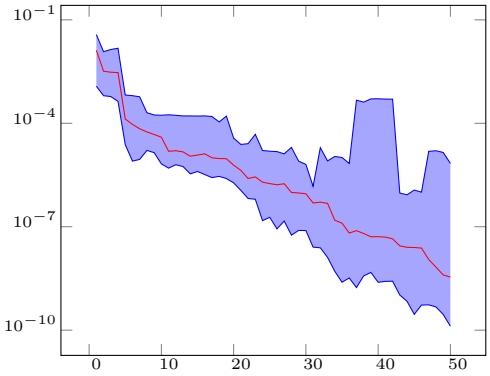
(a) $e_N^s(\boldsymbol{\mu}) = \frac{|s_N(\boldsymbol{\mu}) - s_h(\boldsymbol{\mu})|}{|s_N(\boldsymbol{\mu})|}$



(b) $e_N^{\boldsymbol{u}}(\boldsymbol{\mu}) = \frac{\|\boldsymbol{u}_N(\boldsymbol{\mu}) - \boldsymbol{u}_h(\boldsymbol{\mu})\|_0}{\|\boldsymbol{u}_N(\boldsymbol{\mu})\|_0}$

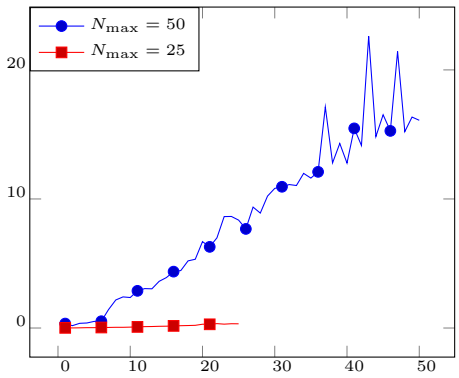


(c) $e_N^p(\boldsymbol{\mu}) = \frac{\|p_N(\boldsymbol{\mu}) - p_h(\boldsymbol{\mu})\|_0}{\|p_N(\boldsymbol{\mu})\|_0}$

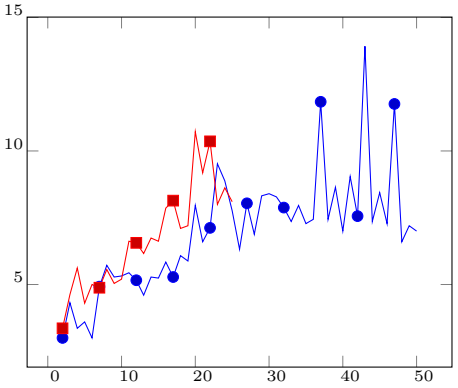


(d) $e_N^T(\boldsymbol{\mu}) = \frac{\|T_N(\boldsymbol{\mu}) - T_h(\boldsymbol{\mu})\|_0}{\|T_N(\boldsymbol{\mu})\|_0}$

Figure 7.12 – Chorus Application, Cabin Part: convergence of the errors vs the size of N. Maximum minimum and average errors evaluated from 50 approximations. $\delta_{tol} = 10^{-10}$ in selective Gram-Schmidt algorithm 7, maximum size of the reduced spaces $97 \times 47 \times 50$



(a) Resolution Time



(b) Number of Iteration

Figure 7.13 – Chorus Application, Cabin Part: average resolution time (in s) and number of iterations in the non-linear solver for the reduced problem vs N.

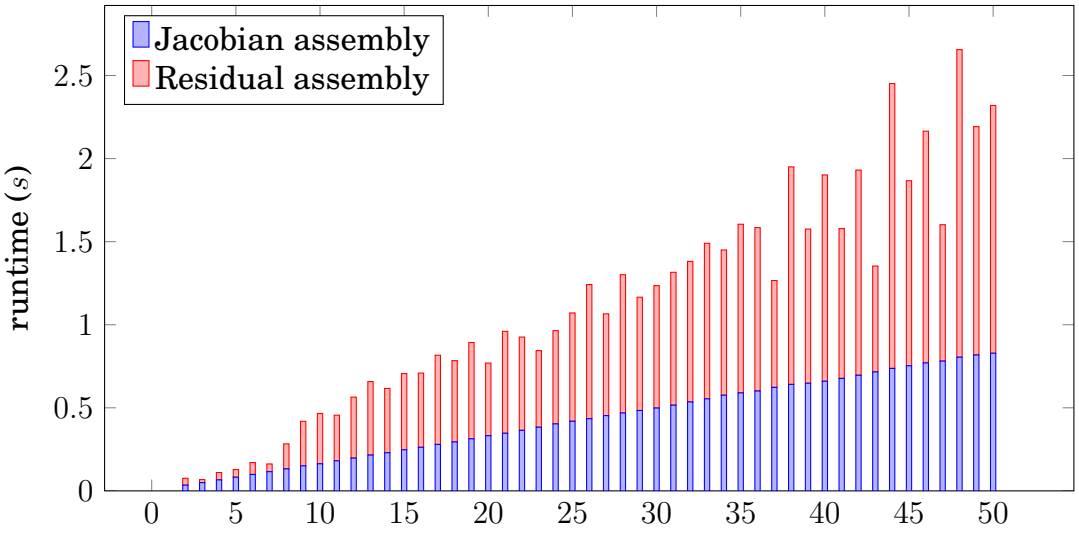


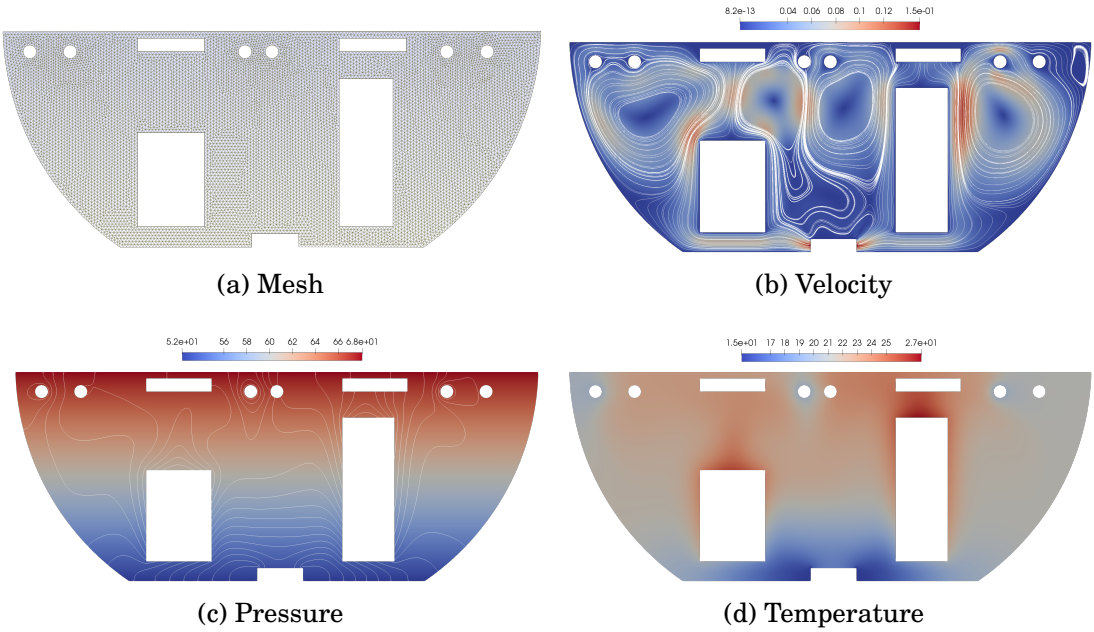
Figure 7.14 – Chorus Application, Cabin Part: Runtime of the main phases for the online resolution. Mean value for 50 runs. Other contributions are negligible. Jacobian and Residual assembly times per iteration.

Results on the Bay

This part of the problem is more complex than the cabin. First, the dimension of the parameter space is larger, with five independent inputs. Then geometry is more complex and so is the fluid flow. The convergence of our solver was complicated even for the FEM resolution, and we had to use both physical and Ψtc continuation, see chapter 2. We present the mesh of simulation and an example of solution profiles on the figure 7.15

Regarding the convergence of the errors in figure 7.16 we observe that the reduced model is not able to correctly approximate the truth solution, even for large values of N. This bad accuracy can be explained by the complexity of the geometry and the larger dimension of the parameter space. We hope that an appropriate error estimator would improve these convergence results.

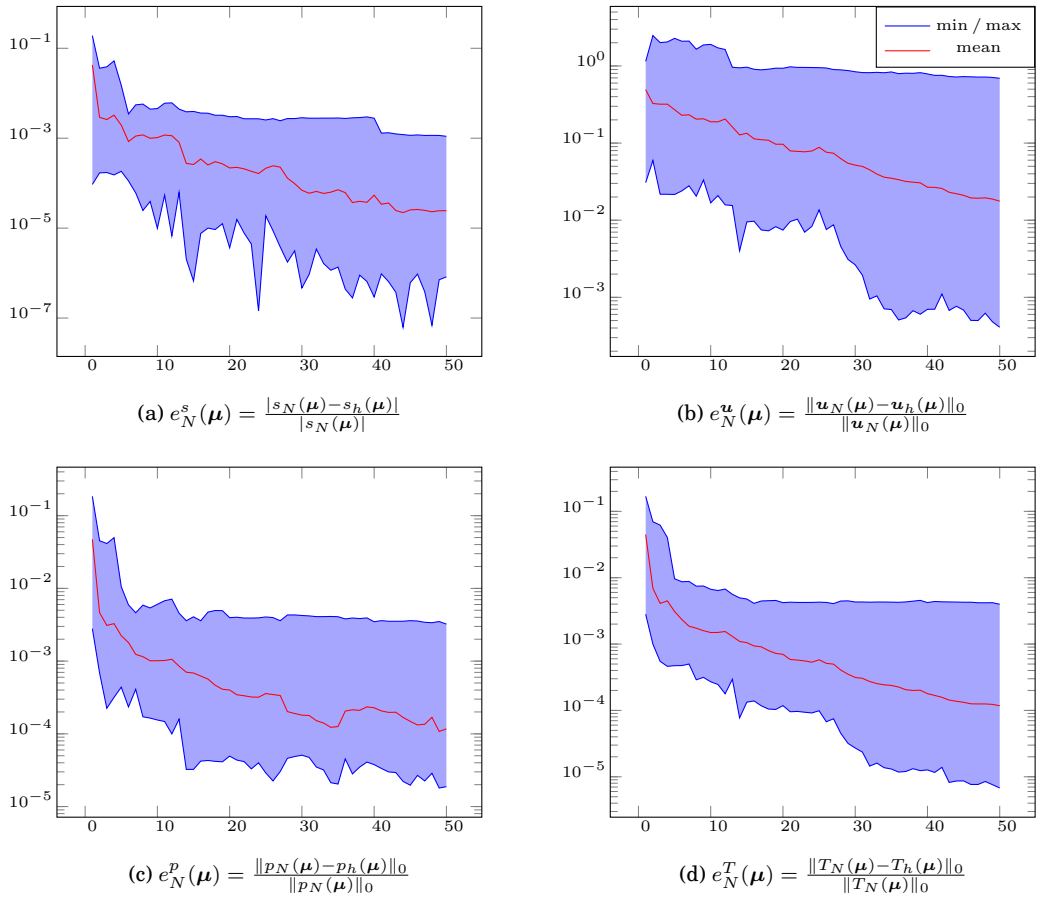
We did not plot any runtime information for this test-case, because these results are entirely similar to those of the cabin case.



(a) Mesh (b) Velocity

(c) Pressure (d) Temperature

Figure 7.15 – Chorus Application, Bay Part: (a) meshes and example of solution profiles ($\boldsymbol{\mu} = (0.004, 0.05, 293, 288, 0.5)$) (b) velocity profile, in $\mathrm{m.s}^{-1}$, with streamlines, (c) pressure field, in Pa, with contour lines, (d) temperature field, in $^{\circ}\mathrm{C}$



(a) $e_N^s(\boldsymbol{\mu}) = \frac{|s_N(\boldsymbol{\mu}) - s_h(\boldsymbol{\mu})|}{|s_N(\boldsymbol{\mu})|}$

(b) $e_N^{\boldsymbol{u}}(\boldsymbol{\mu}) = \frac{\|\boldsymbol{u}_N(\boldsymbol{\mu}) - \boldsymbol{u}_h(\boldsymbol{\mu})\|_0}{\|\boldsymbol{u}_N(\boldsymbol{\mu})\|_0}$

(c) $e_N^p(\boldsymbol{\mu}) = \frac{\|p_N(\boldsymbol{\mu}) - p_h(\boldsymbol{\mu})\|_0}{\|p_N(\boldsymbol{\mu})\|_0}$

(d) $e_N^T(\boldsymbol{\mu}) = \frac{\|T_N(\boldsymbol{\mu}) - T_h(\boldsymbol{\mu})\|_0}{\|T_N(\boldsymbol{\mu})\|_0}$

Figure 7.16 – Chorus Application, Bay Part: convergence of the errors vs the size of N. Maximum minimum and average errors evaluated from 50 approximations. $\delta_{tol} = 10^{-10}$ in selective Gram-Schmidt algorithm 7, maximum size of the reduced spaces $100 \times 50 \times 50$

Conclusion

We proposed in this chapter a method for the reduction of an aerothermal model, using the Reduced Basis Method. The reduced spaces are built for each field, $\boldsymbol{u}$, p, and T. The non-linearity is solved using a Newton algorithm and standard reduction methods for quadratic problems. We also described a method for the reduction of stabilized operators, based on the EIM applied to discrete operators. Finally, we proposed some ideas for the reduction of turbulent problems, this last point remaining theoretical.

The results presented in this chapter are the outcome of this thesis. The resolutions of the FEM problems required most of the techniques developed in the part I (non-linear resolution, stabilization, continuations). The reduction procedure also used the different advanced methods presented in previous chapters (RB for multi-physics and saddle-point problems, EIM and SER for stabilization operators). We obtained excellent results on the opus test case and the cabin. The results for the stabilized simulation and the bay are less accurate and probably require some optimizations.

We propose here some points to study in future works to improve the reduced model:

- The error estimators are missing. We saw in the applications that the convergence of the errors might be difficult, especially when the dimension of the parameter space is growing. The development of an accurate error estimator would significantly improve this convergence. In a first time, we could imagine implementing the dual norm of the residual for aerothermal problems, as an error indicator.
- The theoretical study of the problem is missing too. For now we validated our framework with numerical experiments. It would be interesting to have also a theoretical study of the well-posedness of the reduced problem.
- The reduction of the turbulent models presented in the chapter 4.

We also highlighted two technical issues in the current implementation:

- The data structure used to store the reduced operators is not optimized. The operations on the reduced matrices and vectors become very expensive when the dimension of the reduced space is growing.
- The online computation of the EIM coefficients is still too slow. This part has already been optimized by assembling the online tensors only on the required degrees of freedom, see section 2. However, the cost of these evaluations is a limiting factor, especially when we use multiple EIM. It remains some

optimization to do, for instance, it is possible to store the evaluations of the FE basis functions for all the quadrature points of the interpolation mesh. This kind of technical improvements requires consequent implementation works and could not be achieved in the context of this thesis.

Part III

Contribution to Feel++ Library

CHAPTER 8

Aerothermal Framework

Contents

The implementation of the aerothermal framework evolved a lot during the thesis. As explained in the first part of this report, we tried many strategies for the resolution of the coupled problems. When we eventually fixed the global idea of our solver, we decided to design a very generic solver to use it for many different applications. That kind of "black box" solvers already exists in Feel++ library and are called *toolboxes*. An aerothermal toolbox has been implemented since the beginning of this thesis, however, it does not have all the features required for our simulations. Thus, we started to implement our own toolbox with the advanced features presented in part I. These specific tools are now progressively adapted, optimized and integrated into the Feel++ aerothermal toolbox.

In this chapter, we present the main components of these aerothermal library. The models corresponding to these developments are detailed in the chapters 2, 3 and 4.

1 The Navier Stokes Solver

The resolution of the Navier-Stokes equations is managed by the class `NS`

```
template <typename meshType, int Order, int LMOrder, int TOrder>
class NS
```

where

- `meshType` is the type of mesh. We can change the shape (simplex or hypercube), the dimension and the geometrical order.
  Eventually, we used `Mesh<Simplex<2,1>` in almost all our applications.
- `Order` is the polynomial order k of the Taylor Hood FE space used for the resolution : $V_h \times Q_h = \mathbb{TH}^k_{ch}$, see chapter 2 for the definition.
- `LMOrder` is the polynomial order of the Lagrange Multiplier space used to impose the slip condition when needed. This template argument was used to test the implementation, but the results are only concluding with order 0.
- `TOrder` is the polynomial order k used for the temperature field when the aerothermal option is activated, $X_h = \mathbb{L}^k_{ch}$

This class has been designed to be very generic. Once the polynomial orders are chosen through the template parameters, we can set

- the viscosity μ of the fluid (as a constant or a scalar field),
- the density ρ of the fluid,
- the thermal diffusivity κ of the fluid,
- the thermal expansion coefficient β,
- the source term of the NS equation.

Then we can choose some of the characteristics of the solver through the command line options:

- the solver type: Newton, Picard or Stokes resolution,
- the formulation (symmetric or not),
- the stabilization method used (if needed),
- the continuation method used (if needed),
- the turbulence method used (if needed).

Listing 8.1 – Minimal application solving the Turek 2D benchmark, using the NS class

```
1  int main( int argc, char** argv )
2  {
3      typedef Mesh<Simplex<2,1>> mesh_t;          // define the mesh type
4
5      Environment env( _argc=argc, _argv=argv ); // create Feel environment
6
7      auto mesh = loadMesh( _mesh=new mesh_t );
8
9      auto ns_solver = NS<mesh_t>( mesh );       // instantiate the solver
10     ns_solver.setMu( 0.001 );                  // set the viscosity
11
12     auto U = ns_solver.solve();                // solve the system
13 }
```

We propose an elementary example of code in the snippet 8.1. This piece of code is a minimal application to solve the Turek benchmark 4.1. Once the `solve()` function is called (l.12), the solution `U` can be exported or used to compute quantities of interest.

The `solve()` method is the main function of the class. It may be called with the fixed point (Picard) solver or the Newton solver. However, the most recent features are not implemented in the Picard version and we will focus on the Newton algorithm. The global resolution routine is detailed in the algorithm 10. We also detail the crucial steps of the update of the Jacobian $\boldsymbol{J}$ and the Residual $\boldsymbol{R}$ in the respective algorithms 11 and 12. Note that most of the operators are updated with the Jacobian and reused in the Residual.

In the NS class, the number of blocks in the matrices/vectors can be set dynamically. By default, only the block velocity and pressure are built. If the problem presents slip conditions, we add blocks for the Lagrange multiplier space. Also, if we want to solve the temperature field, we can add the blocks dynamically for the temperature. This feature is particularly convenient and allows working with a very generic solver. This dynamic construction for the matrix is presented in the snippet 8.2, where `Xh`, `Lh` and `Th` are respectively the fluid, the Lagrange multiplier and the temperature FE spaces.

2 The Advection Diffusion Solver

In addition to the Navier-Stokes solver presented in the previous section, we needed a similar tool for the advection-diffusion equations. This class was initially implemented to test the different versions of the stabilization method. Later it has been used for the resolution of the turbulence model. To this purpose, we eventually added the possibility to have a reaction term in the equation.

The design of this class is very similar to the Navier-Stokes solver. We will only present the main features in this section. The prototype of the class is

```
template <typename MeshType, int Order=1>
class AD
```

Listing 8.2 – Example of Dynamic Block Construction for Matrices in Feel++

```
void initMatrix( sparse_matrix_ptrt& M )
{
    if ( M_slip_bc || M_aero )
    {
        BlocksBaseGraphCSR myblockGraph(M_nblocks,M_nblocks);
        myblockGraph(0,0) = stencil(_test=Xh,_trial=Xh,)->graph();

        if ( M_slip_bc )
        {
            myblockGraph(0,M_nblocks-1) =
               stencil(_test=Xh,_trial=Lh)->graph();
            myblockGraph(M_nblocks-1,0) =
               stencil(_test=Lh,_trial=Xh)->graph();
            myblockGraph(M_nblocks-1,M_nblocks-1) =
               stencil(_test=Lh,_trial=Lh)->graph();
        }

        if ( M_aero )
        {
            myblockGraph(0,1) = stencil(_test=Xh,_trial=Th)->graph();
            myblockGraph(1,0) = stencil(_test=Th,_trial=Xh)->graph();
            myblockGraph(1,1) = stencil(_test=Th,_trial=Th)->graph();
        }
        M = backend()->newBlockMatrix(_block=myblockGraph);
    }
    else
        M = backend()->newMatrix(Xh,Xh);
}
```

Algorithm 10: Resolution algorithm in the NS class using Newton method: SOLVE(μ, κ)

Assemble diffusion operators $\boldsymbol{M}_D^u = \mu \nabla \boldsymbol{u}_h : \nabla \boldsymbol{v}_h$;
Assemble null divergence operators $\boldsymbol{M}_B - p_h(\nabla \cdot \boldsymbol{v}_h) + (\nabla \cdot \boldsymbol{u}_h) q_h$;
if *Temperature Field* **then**
    Assemble diffusion operators $\boldsymbol{M}_D^T = \kappa \nabla T_h \cdot \nabla S_h$;
end
if *Initialize with Stokes* **then**
    Solve Stokes problem;
end
repeat
    $\boldsymbol{J} \leftarrow$UPDATEJACOBIAN$(U)$; // see algorithm 11
    $\boldsymbol{R} \leftarrow$UPDATERESIDUAL$(U)$; // see algorithm 12
    Solve $\boldsymbol{J}\Delta = -R$;
    $U \leftarrow U + \Delta$;
until $\frac{\|\boldsymbol{R}(U^k)\|}{\|\boldsymbol{R}(U^0)\|} \leq \delta_{Rtol}$ **or** $\|\boldsymbol{R}(U^k)\| \leq \delta_{Atol}$;
return U;

Algorithm 11: Updating step for the Jacobian, see resolution algorithm 10: `UPDATEJACOBIAN(U)`

$\boldsymbol{J} \leftarrow 0$;
Assemble trilinear contribution of $\boldsymbol{J}$;
$\boldsymbol{J} \leftarrow \boldsymbol{J} + M_B$; `// null divergence operators`
if *Turbulence model* **then**
  Update μ_t with the model; `// Spalart Almaras or` $k-\omega$
  Re-assemble Diffusion operator $\boldsymbol{M}_D^u(\mu + \mu_t)\nabla \boldsymbol{u}_h : \nabla \boldsymbol{v}_h$;
  if *Temperature Field* **then**
    Re-assemble Diffusion operator $\boldsymbol{M}_D^T = (\kappa + \frac{\mu_t}{\mathrm{Pr}})\nabla T_h \cdot \nabla S_h$;
  end
end
$\boldsymbol{J} \leftarrow \boldsymbol{J} + \boldsymbol{M}_D^u$;
if *Temperature Field* **then**
  $\boldsymbol{J} \leftarrow \boldsymbol{J} + \boldsymbol{M}_D^T$;
end
if *Use* Ψtc **then**
  Update pseudo time step;
  Update continuation matrix $\boldsymbol{M}_M$; `// See chapter 2`
  $\boldsymbol{J} \leftarrow \boldsymbol{J} + \boldsymbol{M}_M$;
end
if *Use Stabilization* **then** `// Detailed in chapter 3`
  Update Stabilization coefficient;
  Assemble Stabilization operators $\boldsymbol{M}_S^u$ and $\boldsymbol{M}_S^T$;
  $\boldsymbol{J} \leftarrow \boldsymbol{J} + \boldsymbol{M}_S^u + \boldsymbol{M}_S^T$;
end
Edit $\boldsymbol{J}$ to set boundary conditions;
return $\boldsymbol{J}$;

Algorithm 12: Updating step for the Residual, see resolution algorithm 10: `UPDATERESIDUAL(U)`

$\boldsymbol{R} \leftarrow 0$;
Assemble trilinear contribution of $\boldsymbol{R}$;
Update right hand side $\boldsymbol{F}_R$; `// If needed`
$\boldsymbol{R} \leftarrow \boldsymbol{R} + \boldsymbol{F}_R$;
$\boldsymbol{R} \leftarrow \boldsymbol{R} + \boldsymbol{M}_B U$;
$\boldsymbol{R} \leftarrow \boldsymbol{R} + \boldsymbol{M}_D^u U$;
if *Temperature Field* **then**
  $\boldsymbol{R} \leftarrow \boldsymbol{R} + \boldsymbol{M}_D^T U$;
end
if *Use Stabilization* **then** `// Detailed in chapter 3`
  Update Stabilization coefficient;
  Assemble Stabilization operators $\boldsymbol{F}_S^u$ and $\boldsymbol{F}_S^T$;
  $\boldsymbol{R} \leftarrow \boldsymbol{R} + \boldsymbol{F}_S^u + \boldsymbol{F}_S^T$;
end
Edit $\boldsymbol{R}$ to set boundary conditions;
return $\boldsymbol{R}$;

where

- `meshType` is the type of mesh.
- `Order` is the polynomial order k of the Lagrange FE space used for the resolution : $X_h = \mathbb{L}^k_{ch}$.

Once the solver is initialized, we can set:

- the advection field,
- the diffusion field or constant,
- the reaction field or constant,

and again we can customize the solver with command line options to choose

- the stabilization method (if needed)
- the shock capturing method (if needed)

We propose a short example in the snippet 8.3. This application was used to resolve the thermal layer test case 4.1. The `solveSc()` method allows solving the system using shock capturing methods. The non-linearity of the model is then treated with a fixed point iterative method.

Listing 8.3 – Minimal application solving the Thermal Layer test case, using the AD class

```
int main( int argc, char** argv )
{
    typedef Mesh<Simplex<2,1>> mesh_t;

    Environment env( _argc=argc, _argv=argv );

    double sigma = doption("sigma");       // read from command line options
    double mu = doption("mu");             // read from command line options

    auto mesh = loadMesh( _mesh=new mesh_t );
    auto ad_solver = AD<mesh_t,2>( mesh, "tlayer" );

    ad_solver.setDiff( mu );               // set the diffusivity

    // set the advection field v=(2*y,0)
    ad_solver.adcField().on( elements(mesh), vec(2*Py(),cst(0.) ) );

    if ( sigma!=0 )                        // set the reaction/mass constant
        ad_solver.setMass( sigma );

    auto u = ad_solver.solveSc();          // solve the system
}
```

Listing 8.4 – Spalart Allmaras model: resolution loop using the AD class

```
M_solver = boost::make_shared<solver_type>(mesh);
M_solver->adcField().on( elements(mesh), idv(u) );

do
{
    // update RHS and diffusion coefficient
    M_solver->diffField().on( elements(mesh), 1./sigma*(cst(mu)+idv(mut)) );
    M_solver->rhsField().on( elements(mesh),
                             cb2/sigma*gradv(mut)*trans(gradv(mut))
                             + cb1*(cst(1.)-ft2)*S*idv(mut)
                             - (cw1*fw - cb1/k2*ft2)*idv(mut)*idv(mut)/d2 );

    /// solve the equation
    auto results = M_solver->solve( 3*mu);

    // check the norm of the increment
    double norm2 = normL2( elements(mesh), idv(mut) );
    diff = normL2( elements(mesh), idv(mut)-idv(results) )/norm2;

    // iterate
    mut.on( elements(mesh), max(idv(results), cst(0.) ) );
    i++;
} while ( i<ioption(_name="sa.maxit" ) && doption( _name="sa.tol")<diff );
```

3 Turbulence Models

A direct application of the `AD` class is the resolution of the equations of the turbulence models presented in the chapter 4. We implemented two class `TurbulenceSA` and `TurbulenceKOmega` both inherited from a base class

```
template <typename FluidSpaceType, int Order>
class TurbulenceBase
```

where `FluidSpaceType` is the type of the fluid FE element space and `Order` is the desired polynomial order for the FE space used to solve the turbulent equations.

Both models presented in this chapter involved the resolution of transport equations. Our advection diffusion solver is used within the turbulence class to solve these equations. The snippet 8.4 presents the resolution loop in our Spalart Allmaras class.

Besides the resolution of the transport equation, we also need to compute the

Listing 8.5 – Fast marching algorithm used to compute the field of the distance to wall

```
auto Yh = Pch<1>( mesh );
auto Yh0 = Pdh<0>(mesh);

auto thefms = fms( Yh );
auto phio = Yh->element();
auto mark = Yh0->element();

// the first cell of the mesh on walls is initialized with the distance hmin
for ( int i=0 ; i<wall_marker.size() ; i++ )
    phio.on( _range=elementsWithMarkedFaces( mesh, wall_marker[i]),
             _expr= hMin() );

// all the wall boundary are set to distance 0
for ( int i=0 ; i<wall_marker.size() ; i++ )
    phio.on( markedfaces( mesh, wall_marker[i]), cst(0) );

// we mark the cells which are already initialized
for ( int i=0 ; i<wall_marker.size() ; i++ )
    mark += project( _space=Yh0,
                     _range= elementsWithMarkedFaces(mesh,wall_marker[i]),
                     _expr=cst(1)  );
mesh->updateMarker2( mark );

// the fast marching compute the distance field
M_distance = thefms->march( phio, true);
```

distance of each point of Ω to the nearest point of the wall. The evaluation of this distance to the wall can be particularly challenging - or at least expensive - for complex meshes. We used the fast marching feature of the Feel++ library to compute this distance. This algorithm is entirely parallel and allows fast and accurate computation of the distance to the wall. More details on this feature and its implementation are proposed in [30]. Snippet 8.5 presents an example of using the fast marching to recover the distance to the wall.

3.1 The Stabilization Parameters

The final important tool of our framework is the class `StabCoeff`. It allows computing the stabilization parameter from the advection field and the diffusion field (or constant). The version of the stabilization parameter τ and the version of the characteristic size h can be chosen with command line options. This class is used in both the `NS` and the `AD` class. Once the class is initialized, the coefficient τ is computed by calling

```
auto tau = M_stab->update( idv(u), cst(mu) );
```

where `u` is the convection field and `mu` the diffusion (can be a field).

We detail some interesting points of this class in the next paragraphs.

Initialization: computation of λ_K

As explained in the chapter 3, the coefficient λ_K is an essential ingredient of efficient stabilization parameters. This coefficient is used to scale the local Reynolds number with respect to the polynomial order of the FE space. In practice, λ_K is

computed as the solution of the local eigenvalue problem

$$\lambda_K = \max_{v \in (\mathbb{P}^k(K)\setminus\mathbb{R})} \frac{\|\Delta \boldsymbol{v}\|_K^2}{\|\nabla \boldsymbol{v}\|_K^2}. \tag{8.1}$$

To solve this eigen value problem, we first assemble the laplacian and the gradient operators on each cell, in a global matrix. For that we use the Feel++ language,

```
auto fA = form2( _trial=Xh, _test=Xh, _matrix=matA );
fA = integrate( _range=elements(mesh),
                _expr=inner(laplacian(w),laplaciant(w)) );
auto fB = form2( _trial=Xh, _test=Xh, _matrix=matB );
fB = integrate( _range=elements(mesh),
                _expr=inner(grad(w),gradt(w)) );
```

As specified in (8.1) we have to exclude the constants from the solutions. We chose to add a penalisation term to take care of this constraint.

```
fB += integrate( _range=elements(mesh),
                 _expr=penal*inner(id(w),idt(w)) );
```

Once the operators are assembled, we can extract the local matrices associated to each cell.

```
for ( int i=0;i<nLocalIndices;++i )
{
    for ( int j=0;j<nLocalIndices;++j )
    {
        localMatA(i,j) = matA(extractIndices[i],extractIndices[j]);
        localMatB(i,j) = matB(extractIndices[i],extractIndices[j]);
    }
}
```

We use an Eigen solver to recover the eigen values of the generalized eigen values problem associated to these two matrices. The value of λ_K on each cell is the max of this eigen values.

```
GeneralizedSelfAdjointEigenSolver<MatrixXd> es( localMatA,localMatB );
VectorXd eigen_vls= es.eigenvalues();

double lambda = eigen_vls.maxCoeff();
lambdaK.on( idedelements( mesh,elt.id() ), cst(lambda) );
```

Evaluation of the Characteristic Length h

We defined ans studied three different definition of the characteristic length in the section 2.2.

- h_m is the length of the smallest edge of the cell,
- $h_d = \frac{|\boldsymbol{a}|_p}{|\tilde{\boldsymbol{a}}|_p}\tilde{h}$ with $\tilde{\boldsymbol{a}} = \boldsymbol{M}\boldsymbol{J}^{-1}\boldsymbol{a}$, is the length of the cell in the direction of the current convection field $\boldsymbol{a}$.
- h_h is a definition proposed by Harari and Hughes, evaluated from the measure of the cell and the vertex-to-centroid distance.

The evaluation of h_m is already implemented in Feel++ and did not required any development.

The evaluation of h_d was not very painful, thanks to Feel++ language facilities. As explain in the stabilization chapter, the evaluation of h_d is made by comparing the convection field $\boldsymbol{a}$ with the convection field $\tilde{\boldsymbol{a}}$ on a reference cell. To evaluate $\tilde{\boldsymbol{a}}$ we first define the matrix $\boldsymbol{M}$

```
auto M = mat<2,2>( cst(1.), cst(1./2), cst(0) , cst(std::sqrt(0.75)) );
```

then we can evaluate h_d using the Feel++ keyword `JinvT()` which compute the transposed inverse inverse of the jacobian of the geometric transformation associated to the current cell,

```
h_d.on( elements(mesh), 2*sqrt(inner(a)/inner(M*trans(JinvT())*a)) );
```

The evaluation of h_h requires some more work. The first issue is that the definition depends on the shape of the cells. We will use the 2D simplex as an illustration. For this particular case, h_h can be expressed as

$$h_h = \frac{4S}{\sqrt{3\sum_{a=1}^{3} |x_i - x_c|^2}}. \tag{8.2}$$

This quantity has to be evaluated on each cell. The following snippet shows how we compute h_h using Feel++ language and a loop over the elements of the mesh. Thanks to Feel++ parallel implementation, this evaluation is split using the domain decomposition on each process.

```
// loop over the elements of the mesh
for ( auto const& eltWrap : elements(mesh) )
{
    auto const& elt = unwrap_ref( eltWrap );

    auto b = elt.barycenter();
    double s = 0;

    // loop over the vertices of the cell
    for ( int k=0; k<n_vertices; k++ )
    {
        auto const& c = elt.point(k).node();
        // evaluate the vertex-to-centroid distance
        s += math::pow(c[0]-b[0],2) + math::pow(c[1]-b[1],2);
    }

    h_h.on( idedelements( mesh,elt.id() ), 4*meas()/math::sqrt( 3*s ) );
}
```

Stabilization Parameter on a Reference Mesh

When using the Reduced Basis Method with geometric parameters, we rewrite the weak formulation on a reference domain, using the composition rules for differentiation operators. However, we also have to adapt the stabilization parameter. It has to be evaluated as if it was on the parametrized domain. In the following paragraphs, we denote by $\tilde{\boldsymbol{J}}$ the Jacobian of the mapping from the parametrized to the reference domain.

The evaluation of h_d on the reference domain is particularly convenient as long as we know $\tilde{\boldsymbol{J}}$. We simply have to compose with $\tilde{\boldsymbol{J}}^{-1}$ in the classical formula.

```
h_d.on( elements(mesh), 2*sqrt(inner(a)/inner(M*trans(JinvT())*inv(M_J)*a)) );
```

where M_J represents $\tilde{\boldsymbol{J}}$.

The computation of h_h on a reference domain was a bit more complex since we have to recompute the vertex-to-centroid distance for each new parametrized domain. To optimize this procedure, we first store discontinuous vector fields M_di containing for each cell the vector $B\vec{V}_i$, where B is the barycenter of the cell, and V_i is the ith vertices of the cell.

```
for ( auto const& eltWrap : elements(mesh) )
{
    auto const& elt = unwrap_ref( eltWrap );
    auto b = elt.barycenter();

    auto p0 = elt.point(0).node();
    auto p1 = elt.point(1).node();
    auto p2 = elt.point(2).node();

    M_d1.on( idedelements(mesh,elt.id()),
             (p0[0]-b[0])*oneX() + (p0[1]-b[1])*oneY() );
    M_d2.on( idedelements(mesh,elt.id()),
             (p1[0]-b[0])*oneX() + (p1[1]-b[1])*oneY() );
    M_d3.on( idedelements(mesh,elt.id()),
             (p2[0]-b[0])*oneX() + (p2[1]-b[1])*oneY() );
}
```

Then we can used these vectors fields to compute the vertex-to-centroid distance, using the image of this vector by the jacobian matrix $\tilde{\boldsymbol{J}}$.

```
h_h.on( elements(mesh),
        4*meas()*det(idv(M_J))/sqrt( 3*(inner(idv(M_J)*idv(M_d1))
        +inner(idv(M_J)*idv(M_d2))
        +inner(idv(M_J)*idv(M_d3))) ) );
```

We finally can evaluate the coefficient λ_K on the parametrized domain, using the relation $\lambda_K h_{\tilde{h}}^2 = \text{cst}$. Once we have the value of $\tilde{h}_h$ on the parametrized domain, we can evaluate $\tilde{\lambda}_K = \lambda_K \frac{h_h}{\tilde{h}_h}$. With this last ingredient, we can compute any version of the stabilization parameter τ on a reference domain, knowing only the Jacobian matrix $\tilde{\boldsymbol{J}}$. We can set this matrix in the StabCoeff class, using the method $J()$

```
M_stab->J().on( elements(mesh), mat<2,2>(cst(1.),cst(0),cst(0),cst(1.)) );
M_stab->J().on( markedelements(mesh,"air4"),
                mat<2,2>( cst((ea-eic)/(ear-eic)),cst(0),cst(0),cst(1.)) );
auto tau = M_stab->update( rhoC_a*conv, kappa_a );
```

CHAPTER 9

Reduced Basis Framework

Contents

In the previous chapter, we detailed the developments of our aerothermal library. This library was not fully integrated into Feel++ yet. Hereafter, we will present implementation works made directly in the Feel++ library. We begin, in this chapter, with the development of the RB basis framework.

At the beginning of this thesis work, the RB framework was already well developed in Feel++ library, see [105]. The main features are:

- CRBM for elliptic and parabolic linear coercive problems
  - Primal/Dual resolution
  - Coercivity bounds evaluation through min-θ or SCM
  - Offline/Online residual evaluation
  - Efficient error bounds on the output

- CRBM approximation for non-affine linear coercive problems
  - EIM and error estimators
- RBM for non-linear and multiphysic problems
  - EIM + SER
  - RB for composite spaces (monolithic resolution)

The multiphysics problems were already treated in the framework, see [24], but the reduced basis spaces were always built monolithically. The implementation of composite reduced spaces was the first contribution we needed in Feel++ RB framework. This feature is essential, for instance, with saddle point Stokes problems when we need to build a velocity space with higher dimension than the pressure space, see chapter 5. We eventually implemented a *block* RB framework which allows building composite RB spaces with a block-structured resolution. In addition, we also implemented the tools necessary to deal with trilinear and especially aerothermal problems.

To keep the framework coherent and easy to maintain, it was essential to integrate the new features into the existing code. Thanks to the C++ inheritance paradigm, it was very convenient to derive new models from the classic CRB implementation. To have a better understanding of our choice, we first present an overview of this classic CRB framework on figure 9.1. 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 characteristic 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` is used to recover an affine approximation for non-affine or non-linear problems, see chapter 6.

The classes `ModelCrbBase` and `EIM` are not impacted by the implementation of our new features. We will essentially focus on the modifications in `CRB` and `CRBModel` which are derived respectively in `CRBBlock` and `CRBModelBlock` and later in `CRBAero` and `CRBModelAero`.

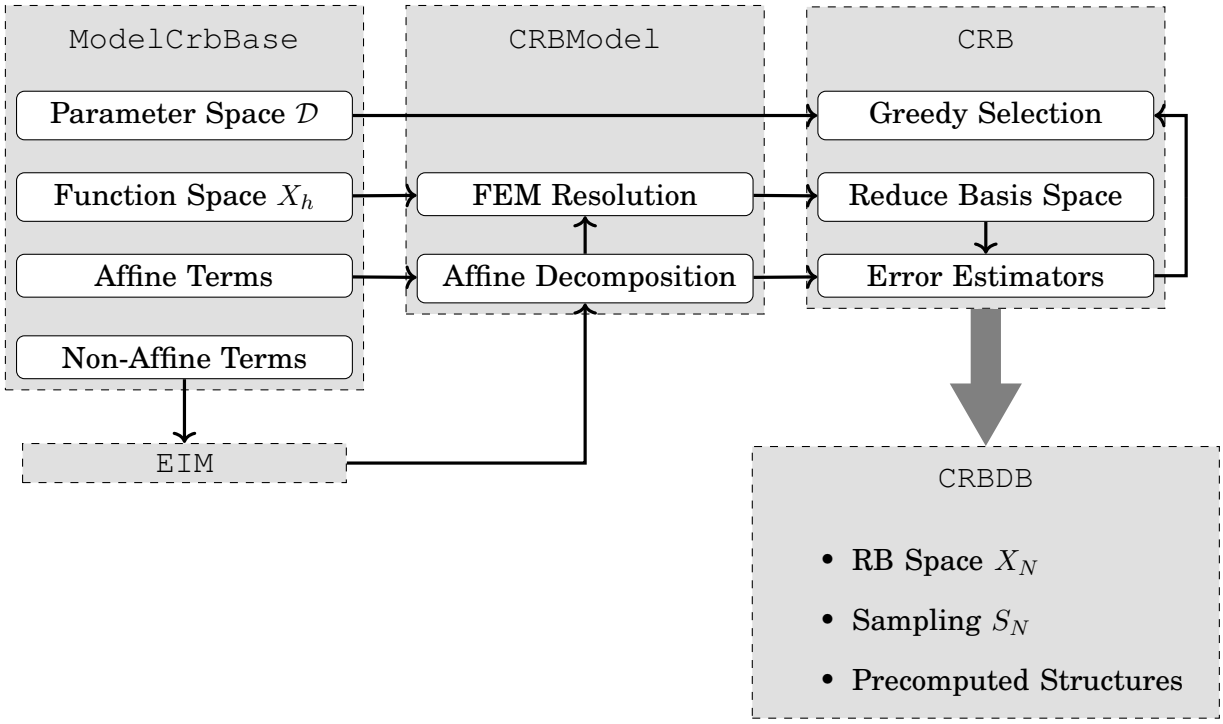


Figure 9.1 – Main classes in Feel++ RB framework

1 Reduced Basis by Block

Most of the modification needed for the block construction of the reduced spaces have been done in the new class

```
template <typename TruthModelType>
class CRBBlock : public CRB<TruthModelType>
```

Only a few functions have been rewritten in the inherited class. We detail the most important modifications in this section.

1.1 Offline Procedure

The offline construction of the reduced basis is entirely managed in the `offline()` method of `CRB`. We keep this algorithm unchanged, but we use virtual functions for the steps which are impacted by the block structure. In algorithm 13 we present a very generic version of the offline routine. The steps in blue have been rewritten in the `CRBBlock` class and will be detailed in further paragraphs.

Remark 14. *We chose to implement a block version of the RB algorithm where the number of blocks is not fixed. It depends on the number of subspaces in the product spaces. To implement this class, we had to use meta-programming. In Feel++, a composite space is a class, mostly defined by template arguments. This construction is very powerful, but it might also provide some additional constraints.*

Algorithm 13: Offline Construction of the Reduced Basis Space

Generate the super sampling $\Xi \subset \mathcal{D}$;
if *Use predefined sampling* **then**
  Generate Sampling W;
end
Chose $\boldsymbol{\mu}_1 \in \Xi$;
repeat
  $N \leftarrow N + 1$;
  $S_N \leftarrow S_{N-1} \cup \{\boldsymbol{\mu}_N\}$;
  $U(\boldsymbol{\mu}_N) \leftarrow$ Solve the FE problem with $\boldsymbol{\mu}_N$;
  Add new vector in the Basis;
  Orthonormalize the basis;
  Precompute the RB matrix and vectors;
  if *Use predefined sampling* **then**
    $\boldsymbol{\mu}_{N+1} \leftarrow W[N+1]$;
  else
    for $\mu \in \Xi$ **do**
      Compute error indicator, $\Delta_N(\mu)$;
    end
    $\mu_{N+1} = \arg\max_{\mu \in \Xi} \Delta_N(\mu)$;
  end
until $\Delta_N(\boldsymbol{\mu}_{N+1}) < \delta_{tol}$ **or** $N \geq N_{max}$;

Listing 9.1 – Minimal example of a loop over a template range, using meta-programming

```
static const int n_block = space_type::nSpaces;
typedef typename mpl::range_c< int, 0, n_block > rangespace_type;

template <typename T>
struct doSomethingOverEachSubSpacesByBlock
{
  template <typename T>
  void operator()( T const& t )const
  {
    // access a subspace using template value
    auto subXh = Xh->template functionSpace<T::value>();
    // now I can use this subspace...
    ...
  }
};

void doSomethingOverEachSubSpaces()
{
  // instantiate the template struct
  doSomethingOverEachSubSpacesByBlock my_struct();
  // instantiate an object of type range
  rangespace_type range;
  // loop on the structure
  boost::fusion::for_each( range, my_struct );
}
```

In the algorithms presented in this section, we regularly iterate over the subspaces of a composite space. This operation cannot be done in Feel++ with a classical loop. One solution is to iterate, using tool like `boost::fusion::for_each`*, over templated structures. We propose a minimal example of this construction in the snippet 9.1. Note that all the loops over the subspaces, in the following paragraphs, have been implemented using this kind of structures.*

We first introduce some notations for the product space and the block structure. We denote by $X_h = X_{h1} \times \cdots \times X_{hR}$ a composite FE space. X_h can be naturally decomposed (in the sense of the physic) as the product of R subspaces noted $X_{hr}, 1 \leq r \leq R$. We also assume for each block the affine decomposition: $\forall 1 \leq r, c \leq R$,

$$\begin{aligned} a^{rc}(u_r, u_c; \boldsymbol{\mu}) &= \sum_{q=1}^{Q^a} \theta_q^a(\boldsymbol{\mu}) a_q^{rc}(u_r, u_c) \\ f^r(u_r; \boldsymbol{\mu}) &= \sum_{q=1}^{Q^f} \theta_q^f(\boldsymbol{\mu}) f_q^r(u_r) \end{aligned} \tag{9.1}$$

where $a^{rc}(\cdot, \cdot; \boldsymbol{\mu}) : X_{hr} \times X_{hc} \times \mathcal{D} \to \mathbb{R}$, $a^{rc}(\cdot, \cdot) : X_{hr} \times X_{hc} \to \mathbb{R}$ and $f^r(\cdot; \boldsymbol{\mu}) : X_{hr} \times \mathcal{D} \to \mathbb{R}$, $f^r(\cdot) : X_{hr} \to \mathbb{R}$. Since we are detailing discrete algorithms, it is convenient to identify bilinear/linear forms and matrices/vectors. So we will transparently switch between a_q^{rc}, f_q^r and A_q^{rc}, F_q^r. The underlying block structure is such that

$$A_q = \begin{bmatrix} A_q^{11} & \cdots & A_q^{1R} \\ \vdots & & \vdots \\ A_q^{R1} & \cdots & A_q^{RR} \end{bmatrix}, \quad F_q = \begin{bmatrix} F_q^1 \\ \vdots \\ F_q^R \end{bmatrix}, \quad U = \begin{bmatrix} u_1 \\ \vdots \\ u_R \end{bmatrix} \tag{9.2}$$

We will not detail the steps Add new vector in the Basis; and Orthonormalize the basis of the algorithm 13. In the first one, we simply add each subfield u_r in

the corresponding reduced subspace X_{Nr}. In the second one, we orthonormalize (if necessary) the basis vectors of each reduced sub-space X_{Nr}.

The assembly of the reduced matrices and vectors is a bit more technical than the previous steps. The aim is to construct, $\forall 1 \leq r, c \leq R$ the matrices and vectors

$$\begin{aligned} A^{rc}_{Nq} &= a^{rc}_q(u_r, u_c) = u_r^\intercal A^{rc}_q u_c, \quad \forall 1 \leq q \leq Q^a, \\ F^{r}_{Nq} &= f^{r}_q(u_r) = u_r^\intercal F^{r}_q, \qquad\quad \forall 1 \leq q \leq Q^f. \end{aligned} \tag{9.3}$$

As presented in algorithm 14, it essentially consists in many nested loops. In this algorithm, n_r and n_c denote the number of basis vectors added respectively in X_{Nr} and X_{Nc}.

Algorithm 14: Block construction of the reduced matrices and vectors. cf l.12 in algorithm 13

```
1  for 1 ≤ r ≤ R do                                          // Loop on the Row
2      for 1 ≤ c ≤ R do                                      // Loop on the Columns
3          for 1 ≤ q ≤ Q^a do                                // Loop on the affine decomposition
               /* Update the last rows of A^rc_Nq.  */
4              for N_r − n_r ≤ i ≤ N_r do                    // Loop on the first basis
5                  for 1 ≤ j ≤ N_c do                        // Loop on the second basis
6                      ξ_i ← i-th basis in X_Nr;
7                      ξ_j ← j-th basis in X_Nc;
8                      (A^rc_Nq)_ij = ξ_i^⊺ A^rc_q x_j;
9                  end
10             end
               /* Update the last columns of A^rc_Nq.  */
11             for 1 ≤ i ≤ N_r do                            // Loop on the first basis
12                 for N_c − n_c ≤ j ≤ N_c do                // Loop on the second basis
13                     ξ_i ← i-th basis in X_Nr;
14                     ξ_j ← j-th basis in X_Nc;
15                     (A^rc_Nq)_ij = ξ_i^⊺ A^rc_q x_j;
16                 end
17             end
18         end
19     end
20     for 1 ≤ q ≤ Q^f do                                    // Loop on the affine decomposition
           /* Update the last rows of F^r_Nq.  */
21         for N_r − n_r ≤ i ≤ N_r do                        // Loop on the basis
22             ξ_i ← i-th basis in X_Nr;
23             (F^r_Nq)_i = ξ_i^⊺ F^r_q;
24         end
25     end
26 end
```

Error Estimator

Using an efficient error bound requires to have two main components: bounds for the coercivity/inf-sup constants and dual norm of the residual. The first ingredient is problem dependent and cannot be implemented generically for the multiphysics problems. We already have solutions (min-theta algorithm, SCM) for the evaluation of the coercivity constant in the classic `CRB` framework but these methods have not been extended to the block structure due to lack of time.

The evaluation of the residual has been implemented but only in the particular case of 2 subspaces. The offline precomputation of the different structures can be very expensive since it requires the resolution of many linear systems to evaluate the Riesz representations, see appendix 3. In the original `CRB` class, these Riesz representations were reevaluated multiple times because it was inconceivable to store all these vectors in the memory. However, these multiple reevaluations are very costly and especially for block construction. We finally decided to store the Riesz representations on the disk and to reread them each time they are needed. The dual norm of the residual is then used as an error indicator for certain problems, see Stokes test case 3.

The implementation of the residual evaluation for an arbitrary number of subspaces is on the todo list.

1.2 Online Procedure

The online phase of the reduced basis methodology essentially consists in the resolution of the reduced problem. For now the `CRBBlock` online integrates a fixed point algorithm, but a Newton method has been implemented in `CRBAero` and will be detailed in a further section.

The fixed point procedure is detailed in the algorithm 15. We simplified the algorithm by assuming that all the reduced subspaces have the same dimension N. In practice, with the selective Gram-Schmidt orthonomalization7 and the addition of the supremizer, the dimension of the sub-spaces may vary. The only remarkable thing is the assembly of the matrix $\boldsymbol{A}$ and the vector $\boldsymbol{F}$. In practice, this step is realized using the block operators of the Eigen library which simplifies the treatment of the blocks. Here is an example of an addition is the first block of the matrix `A` of size `N0`$\times$`N0`

```
A.block(0,0,N0,N0) += betaAqm[q][m]*blockAqm[0][0][q][m].block(0,0,N0,N0);
```

Algorithm 15: Online fixed point procedure for block RB.

Input: N, $\boldsymbol{\mu}$
Create matrix $\boldsymbol{A}$ of size $NR \times NR$;
Create vector $\boldsymbol{F}$ of size NR;
Compute $\theta_q^a(\boldsymbol{\mu})$ and $\theta_q^f(\boldsymbol{\mu})$;
repeat
  for $1 \leq r \leq R$ **do**
    for $1 \leq c \leq R$ **do**
      for $1 \leq q \leq Q^a$ **do**
        for $(r-1)N \leq i < rN$ **do**
          for $(c-1)N \leq j < cN$ **do**
            $(\boldsymbol{A})_{ij} \leftarrow (\boldsymbol{A})_{ij} + \theta_q^a(\boldsymbol{\mu}) A_{Nq}^{rc}$;
          end
        end
      end
    end
    for $1 \leq q \leq Q^a$ **do**
      for $(r-1)N \leq i < rN$ **do**
        $(\boldsymbol{F})_i \leftarrow (\boldsymbol{F})_i + \theta_q^f(\boldsymbol{\mu}) F_{Nq}^r$;
      end
    end
  end
  $U^* \leftarrow U$;
  $U \leftarrow$Solution of $AU = F$;
  is_finished $\leftarrow$ is_linear **or** $\|U - U^*\|_2 \leq \delta_{\text{tol}}$;
until is_finished;
return U;

2 Reduced Basis for Aerothermal Problems

The reduced basis construction for aerothermal problems, presented in the chapter 7, is made in a decoupled way. We construct three reduced sub-spaces: for the velocity, the pressure and the temperature. The `CRBBlock` class, introduced in the previous section, provides the tools to construct these reduced sub-spaces. In this section, we present some details of a new inherited class

```
template <typename TruthModelType>
class CRBAero : public CRBBlock<TruthModelType>
```

This class contains the algorithm necessary for the construction of the reduced basis and for the online evaluation of reduced solution.

2.1 Offline Procedure

We reuse the algorithm 13 of the previous section and we add in red, in algorithm 16, the steps which have to be modified for the RB aerothermal framework. The

Algorithm 16: Offline Construction of the Reduced Basis Space for Aerothermal RB

Generate the super sampling $\Xi \subset \mathcal{D}$;
if *Use predefined sampling* **then**
  Generate Sampling W;
end
Chose $\boldsymbol{\mu}_1 \in \Xi$;
repeat
  $N \leftarrow N + 1$;
  $S_N \leftarrow S_{N-1} \cup \{\boldsymbol{\mu}_N\}$;
  $U(\boldsymbol{\mu}_N) \leftarrow$ Solve the FE problem with $\boldsymbol{\mu}_N$;
  Add new vector in the Basis;
  Orthonormalize the basis;
  Precompute the RB matrix and vectors;
  Precompute the reduced trilinear contributions;
  if *Use predefined sampling* **then**
    $\boldsymbol{\mu}_{N+1} \leftarrow W[N+1]$;
  else
    for $\mu \in \Xi$ **do**
      Compute error indicator, $\Delta_N(\mu)$;
    end
    $\mu_{N+1} = \arg\max_{\mu \in \Xi} \Delta_N(\mu)$;
  end
until $\Delta_N(\boldsymbol{\mu}_{N+1}) < \delta_{tol}$ **or** $N \geq N_{max}$;

resolution of the finite element problem is particular since it implies quadratic non-linearity. This part is modified in the class `CRBModelAero`. The resolution uses a Newton iterative method and the model provides the affine decomposition of the trilinear, bilinear and linear forms.

The precomputation of the reduced structures now integrates the evaluation of the order-3 tensors, as explained in the chapter 7, in equation (7.11). In the particular context of the aerothermal problems, we know that we only have quadratic contributions in certain blocks. We will only assemble the order-3 tensors $\boldsymbol{T}_q^1$ and $\boldsymbol{T}_q^2$ for the convection operators respectively of the fluid and temperature equations. In addition of the construction 14, the class `CRBAero` will also assemble these tensors, following the algorithm 17, where $N_{\boldsymbol{u}}$ and N_T are the current dimension of respectively the velocity and the temperature spaces, $n_{\boldsymbol{u}}$ and n_T are the number of basis vectors added in these spaces.

Algorithm 17: Precomputation of the reduced order-3 tensors

for $1 \leq q \leq Q^c$ **do**
  for $1 \leq i \leq N_{\boldsymbol{u}}$ **do**
    for $1 \leq j \leq N_{\boldsymbol{u}}$ **do**
      for $1 \leq k \leq N_{\boldsymbol{u}}$ **do**
        if $i \geq N_{\boldsymbol{u}} - n_{\boldsymbol{u}}$ **or** $j \geq N_{\boldsymbol{u}} - n_{\boldsymbol{u}}$ **or** $k \geq N_{\boldsymbol{u}} - n_{\boldsymbol{u}}$ **then**
          $(\boldsymbol{T}_q^1)_{ijk} \leftarrow c_q(\boldsymbol{\zeta}_i, \boldsymbol{\zeta}_j, \boldsymbol{\zeta}_k)$;
        end
      end
    end
  end
end
for $1 \leq q \leq Q^a$ **do**
  for $1 \leq i \leq N_T$ **do**
    for $1 \leq j \leq N_{\boldsymbol{u}}$ **do**
      for $1 \leq k \leq N_T$ **do**
        if $i \geq N_T - n_T$ **or** $j \geq N_{\boldsymbol{u}} - n_u$ **or** $k \geq N_T - n_T$ **then**
          $(T_q^2)_{ijk} \leftarrow a_q(\xi_i, \boldsymbol{\zeta}_j, \xi_k)$;
        end
      end
    end
  end
end

2.2 Online Procedure

The online procedure for the aerothermal framework is already well detailed in the chapter 7. We will not rewrite these algorithms here, but we decided to illustrate them with a detailed explanation of the code.

In the class `CRBAero`, the virtual method `onlineSolve` has been totally rewritten. Its prototype is

```
typename CRBAero<TruthModelType>::matrix_info_tuple
onlineSolve(  size_type N, parameter_type const& mu, std::vector< vectorN_type > & uN,
              std::vector< vectorN_type > & uNdu, std::vector<vectorN_type> & uNold,
              std::vector<vectorN_type> & uNduold, std::vector< double > & output_vector,
              int K, bool print_rb_matrix, bool computeOutput ) const;
```

Once the different variables are initialized, we start with the assembly of the bilinear contributions for the Jacobian

```
for ( int q=0; q<model->sizeOfBilinearJ(); q++ )
{
    for ( int m=0; m<model->mMaxA(q); m++ )
    {
        // first row
        M_Jbil.block( 0,      0, N0, N0) += betaJqm[q][m]*blockAqm(0,0)[q][m].block(0,0,N0,N0);
        M_Jbil.block( 0,     N0, N0, N1) += betaJqm[q][m]*blockAqm(0,1)[q][m].block(0,0,N0,N1);
        M_Jbil.block( 0, N0+N1, N0, N2) += betaJqm[q][m]*blockAqm(0,2)[q][m].block(0,0,N0,N2);
        // second row
        M_Jbil.block(N0,      0, N1, N0) += betaJqm[q][m]*blockAqm(1,0)[q][m].block(0,0,N1,N0);
        M_Jbil.block(N0,     N0, N1, N1) += betaJqm[q][m]*blockAqm(1,1)[q][m].block(0,0,N1,N1);
        M_Jbil.block(N0, N0+N1, N1, N2) += betaJqm[q][m]*blockAqm(1,2)[q][m].block(0,0,N1,N2);
        // third row
        M_Jbil.block(N0+N1,     0,N2,N0) += betaJqm[q][m]*blockAqm(2,0)[q][m].block(0,0,N2,N0);
        M_Jbil.block(N0+N1,    N0,N2,N1) += betaJqm[q][m]*blockAqm(2,1)[q][m].block(0,0,N2,N1);
        M_Jbil.block(N0+N1,N0+N1,N2,N2) += betaJqm[q][m]*blockAqm(2,2)[q][m].block(0,0,N2,N2);
    }
}
```

and for the Residual

```
for ( int q=0; q<model->sizeOfLinearR(); q++ )
{
    for ( int m=0; m<model->mMaxF(0,q); m++ )
    {
        M_Rli.segment(    0,N0) += betaRqm[0][q][m]*blockFqm(0)[q][m].head(N0);
        M_Rli.segment(   N0,N1) += betaRqm[0][q][m]*blockFqm(1)[q][m].head(N1);
        M_Rli.segment(N0+N1,N2) += betaRqm[0][q][m]*blockFqm(2)[q][m].head(N2);
    }
}
```

then, the non-linear solver is called and the rest of the work will be done in the functions

```
void
updateJacobianOnline( const map_dense_vector_type& X, map_dense_matrix_type& J,
                      parameter_type const& mu , int N ) const
```

and

```
void
updateResidualOnline( const map_dense_vector_type& X, map_dense_vector_type& R,
                      parameter_type const& mu , int N ) const
```

The Jacobian is updated in three steps. First we add the bilinear contributions, already assembled before the resolution loop

```
J += M_Jbil;
```

then we assemble the trilinear contributions

```
auto betaTri = this->M_model->computeBetaTri( mu );
for ( int q=0; q<model->QTri(); q++ )
{
    for ( int k=0; k<N0; k++ )
    {
        for ( int i=0; i<N0; i++ )
        {
            J(k,i) += betaTri[q]*(blockTriqm(0,0)[q][k].row(i).head(N)).dot(X.segment(0,N0));
            J(k,i) += betaTri[q]*(blockTriqm(0,0)[q][k].col(i).head(N)).dot(X.segment(0,N0));
        }
    }
    for ( int k=0; k<N2; k++ )
    {
        for( int i=0; i<N2; i++ )
        {
            J(N0+N1+k,N0+N1+i) += betaTri[q]*(blockTriqm(2,0)[q][k].row(i)).dot(X.segment(0,N0));
        }

        for( int j=0; j<N0; j++ )
        {
            J(N0+N1+k,j) += betaTri[q]*(blockTriqm(2,0)[q][k].col(j)).dot(X.segment(N0+N1,N2));
        }
    }
}
```

and finally we assemble the non linear contributions (if needed), with coefficients given by the EIM

```
for ( int q=model->sizeOfBilinearJ(); q< model->Qa(); q++ )
{
    for ( int m=0; m<model->mMaxA(q); m++ )
    {
        // first row
        J.block( 0,     0, N0, N0) += betaJqm[q][m]*blockAqm(0,0)[q][m].block(0,0,N0,N0);
        J.block( 0,    N0, N0, N1) += betaJqm[q][m]*blockAqm(0,1)[q][m].block(0,0,N0,N1);
        J.block( 0, N0+N1, N0, N2) += betaJqm[q][m]*blockAqm(0,2)[q][m].block(0,0,N0,N2);
        // second row
        J.block( N0,     0, N1, N0) += betaJqm[q][m]*blockAqm(1,0)[q][m].block(0,0,N1,N0);
        J.block( N0,    N0, N1, N1) += betaJqm[q][m]*blockAqm(1,1)[q][m].block(0,0,N1,N1);
        J.block( N0, N0+N1, N1, N2) += betaJqm[q][m]*blockAqm(1,2)[q][m].block(0,0,N1,N2);
        // third row
        J.block( N0+N1,     0, N2, N0) += betaJqm[q][m]*blockAqm(2,0)[q][m].block(0,0,N2,N0);
        J.block( N0+N1,    N0, N2, N1) += betaJqm[q][m]*blockAqm(2,1)[q][m].block(0,0,N2,N1);
        J.block( N0+N1, N0+N1, N2, N2) += betaJqm[q][m]*blockAqm(2,2)[q][m].block(0,0,N2,N2);
    }
}
```

We use the same idea for the update of the Residual. We start by adding the bilinear an linear contributions

```
R += M_Rli;
R += M_Jbil*X;
```

then the trilinear part

```
auto betaTri = this->M_model->computeBetaTri( mu );
for ( int q=0; q<model->QTri(); q++ )
{
    for ( int k=0; k<N0; k++ )
        for ( int i=0; i<N0; i++ )
            temp(k,i) = (blockTriqm(0,0)[q][k].row(i)).dot(X.segment(0,N0));
    for ( int k=0; k<N2; k++ )
        for ( int i=0; i<N2; i++ )
            temp(N0+N1+k,N0+N1+i) = (blockTriqm(2,0)[q][k].row(i)).dot(X.segment(0,N0));
    R += betaTri[q]*temp*X;
}
```

and finally the non-linear terms

```
for ( int q=model->sizeOfLinearR(); q<model->Ql(0); q++ )
{
    for ( int m=0; m<model->mMaxF(0,q); m++ )
    {
        R.segment(    0,N0) += betaRqm[0][q][m]*blockFqm(0)[q][m].head(N0);
        R.segment(   N0,N1) += betaRqm[0][q][m]*blockFqm(1)[q][m].head(N1);
        R.segment(N0+N1,N2) += betaRqm[0][q][m]*blockFqm(2)[q][m].head(N2);
    }
}
```

CHAPTER 10

EIM Framework for Discrete Operators

Contents

The work on the EIM algorithm presented in the section 2 represent the most important implementation effort. This feature has been entirely coded and integrated into the Feel++ RB framework during this thesis. We present in this chapter some interesting details and advanced C++ techniques used in this implementation.

The Feel++ library already had an EIM framework when we started this implementation. This initial version uses analytic expressions and their evaluation on the quadrature points. Also, this version was not adapted to the treatment of discrete operators. To distinguish this version from the discrete one, we used the abusive notation DEIM for our new implementation. To remain consistent with the code we will keep using this denomination in this chapter.

1 Design of the Implementation

The DEIM framework have been split into four classes, with the inheritance relations detailed on the figure 10.1.

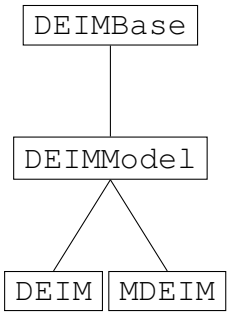


Figure 10.1 – Inheritance Diagram of the Classes in the DEIM Framework

The class

```
template <typename ParameterSpaceType, typename SpaceType, typename TensorType>
class DEIMBase : public CRBDB
```

contains the essence of the DEIM algorithm. This class is managing all the algebraic operations which do not require an access to the RB model:

- The greedy algorithm
- The research of maximum entry in the tensors
- The evaluation of a specific entry in a tensor
- The computation of the $\beta_m(u, \boldsymbol{\mu})$

The class is templated by the type of the tensor. It allowed to write only one class for both matrices and vectors. The class also inherits from the class `CRBDB` which manages to save the essential information in a database.

The class

```
template <typename ModelType, typename TensorType>
class DEIMModel :
        public DEIMBase<typename ModelType::parameterspace_type,
                        typename ModelType::space_type, TensorType>
```

is the interface with the RB model. This class will call the assembly functions of the model and possibly the resolution functions for non-linear problems. It also contains the mapping for the degrees of freedom (DoFs) between the interpolation space $X_{\mathcal{I}_M}$ and the FE space X_h. This point is detailed in the section 2.

The class

```
template <typename ModelType>
class DEIM :
      public DEIMModel<ModelType,typename Backend<typename ModelType::value_type>::vector_type>
```

and

```
template <typename ModelType>
class MDEIM :
      public DEIMModel<ModelType,
                       typename Backend<typename ModelType::value_type>::sparse_matrix_type>
```

are specific to the type of tensors. `DEIM` is the instensiation for the vectors and `MDEIM` the instantiation for the matrices. These classes also provide methods for the construction of the interpolation mesh $\mathcal{T}_{\mathcal{I}_M}$

The global structure of the DEIM algorithm is basic and did not present many difficulties for the implementation. The main challenge of the implementation was the creation of the interpolation mesh and space, required for the optimized online evaluation of the $\beta_m(u, \boldsymbol{\mu})$.

2 Construction of the Interpolation Space

At the end of the DEIM greedy algorithm, we obtain a set of indices $\mathcal{I}_M$. To optimized the online computation of the $\beta_m(u, \boldsymbol{\mu})$, we have to assemble reduced tensors $T_{\mathcal{I}_M}(u, \boldsymbol{\mu})$ only on the associated DoFs. This procedure is detailed in the section 2. We will describe here some interesting details.

Remark 15. *In the following paragraphs, we only present the construction for a single FE space. The construction for product spaces was also implemented and is basically a loop of the following method on each space.*

The Interpolation Mesh and Space

The first step is to extract the interpolation mesh $T_{\mathcal{I}_M}$ from the full mesh $\mathcal{T}_h$. From the `index_list`, we have to create a list of the elements in $\mathcal{T}_h$ that contains the indices in `index_list`

```
for ( auto index : index_list )
{
    auto searchGpDof = Xh->dof()->searchGlobalProcessDof( index );
    if ( boost::get<0>( searchGpDof ) )
    {
        size_type gpdof = boost::get<1>( searchGpDof );
        for ( auto const& dof : Xh->dof()->globalDof( gpdof ) )
        {
            size_type eltId = dof.second.elementId();
            if ( mesh->element( eltId ).isGhostCell() )
                continue;
            this->M_elts_ids.insert( eltId );
        }
    }
}
```

After this loop, each processor has a set of elements, `M_elts_ids`, containing the local elements to extract. The extraction function is then provided by the Feel++ library

```
auto submesh = createSubmesh( mesh, idelements(mesh,this->M_elts_ids.begin(),
                              this->M_elts_ids.end()) );
```

At this point, each processor contains its local part of the extracted `submesh`. To gather all the contributions, we will write them on the disk and then read the complete extracted mesh with each processor:

```
saveGMSHMesh( _mesh=submesh, _filename=this->name(true)+"-submesh.msh" );
Environment::worldComm().barrier();
auto seqmesh = loadMesh( _mesh=new mesh_type,
                         _filename=this->name(true)+"-submesh.msh",
                         _worldcomm= Environment::worldCommSeq() );
```

Here the mpi barrier, `Environment::worldComm().barrier();`, is essential to ensure that all the processors finished writing there contributions before the `seqmesh` is read. From this sequential mesh we can now build the interpolation space

```
Rh = this->newInterpolationSpace(seqmesh);
```

Once this space is generated, it will be used in the online model to assemble the reduced tensors $T_{\mathcal{I}_M}(u, \boldsymbol{\mu})$. However, we are not interested in all the value in this reduced tensor. We only need the entry corresponding to the index in $\mathcal{I}_M$. Here the issue is that the DoFs have entirely different indices in $X_{\mathcal{I}_M}$ and X_h. To deal with that problem, we implemented a specific class to build the mapping between a parallel space and a sequential extracted space.

Sequential/Parallel Mapping

In Feel++, once a submesh is extracted, all the elements are renumbered. The only data remaining unchanged is the numbering of the vertex of the mesh. We will use this property to build the mapping between the two spaces.

We denote by X_p and $\mathcal{T}_p$ the parallel space and mesh. These data are split between all the processors. We also note X_s and $\mathcal{T}_s$ the sequential space and mesh. All the processors have the entirety of these data. We assume that we have the relation $\mathcal{T}_s \subset \mathcal{T}_p$. The map between the two spaces has to contain the two-way correspondence $\text{DoF}_s \longleftrightarrow (\text{n}_p, \text{DoF}_p)$ where n_p is number of the processor having the parallel DoF DoF_p corresponding to the sequential DoF DoF_s.

On each processor, we first store for each element in $\mathcal{T}_s$ the numbering of its vertices

```
for ( auto const& eltWrap : elements(Xs->mesh()) )
{
    auto const& elt = unwrap_ref( eltWrap );
    std::set<int> pts_id;
    for ( int p=0; p<elt.nPoints(); p++ )
        pts_id.insert( elt.point(p).id() );
    elts_map_s[pts_id] = elt.id();
}
```

Then, each processor will look if its part of $\mathcal{T}_p$ has elements in commun with $\mathcal{T}_s$. For each commun element, the map is then constructed by identifying the DoF associated in X_p and X_s. This procedure is detailed in the snippet 10.1.

Once the map is built, we have access to useful methods to do the correspondence between the two spaces:

Listing 10.1 – Construction of the map between the DoF of the sequential and the parallel space

```
// loop on the elements of the full mesh splited between all procs
for ( auto const& eltWrap : elements(Xp->mesh()) )
{
    auto const& elt = unwrap_ref( eltWrap );
    std::set<int> pts_id;
    for ( int p=0; p<elt.nPoints(); p++ )
        pts_id.insert( elt.point(p).id() + M_shift );

    // check if this elements is in the submesh of the sequential space
    auto map_it = elts_map_s.find( pts_id );
    if ( map_it!=elts_map_s.end() ) // the element exists in the seq mesh
    {
        int eid_s = map_it->second;

        // loop on each ldof of the element : get the globaldof id associated
        // and put it in the maps
        for ( auto const& ldof : Xp->dof()->localDof(elt.id()) )
        {
            int gdof_s = Xs->dof()->localToGlobalId( eid_s, ldof.first.localDof() );
            int gdof_p = ldof.second.index() ;

            auto s_to_p_it = M_s_to_p.find( gdof_s );
            if ( s_to_p_it==M_s_to_p.end() )
            {
                M_s_to_p[gdof_s] = std::make_pair( Xp->worldComm().globalRank(),
                                                   gdof_p );
                M_p_to_s[gdof_p] = gdof_s;
            }
        }
    }
}
```

- `int parallelToSequential( size_type const& p_dof )`: returns the sequential DoF corresponding to the parallel Dof `p_dof`.
- `std::pair<int,int> sequentialToParallel( size_type const& s_dof )`: returns the couple (n_p,DoF_p) corresponding to the sequential Dof `s_dof`.
- `void project( element_type& us, element_type const& up )`: projects a parallel element from X_p on X_s. This function also gathers the contributions from all processors, this procedure is detailed in the snippet 10.2

This map and those methods are used in DEIM to extract the values of $T_{\mathcal{I}_M}(u, \boldsymbol{\mu})$ and to project the reduced basis on the interpolation space (for non-linear problems only).

Listing 10.2 – Gathering of the contributions from all processors after the projection of a parallel element on the sequential space.

```
1  // loop on the elements of the full mesh splited between all procs
2  template <typename ElementType>
3  void gather( ElementType& us )
4  {
5      int world_size = Environment::worldComm().globalSize();
6      auto Rh = us.functionSpace();
7      auto ut = Rh->element();
8      std::vector<ElementType> all_u( world_size, ut );
9
10     mpi::all_gather( Environment::worldComm().globalComm(),
11                      us,
12                      all_u );
13
14     for ( auto ut : all_u )
15     {
16         for ( int j=0; j<us.size(); j++ )
17         {
18             if ( ut(j)!=0 )
19                 us(j) = ut(j);
20         }
21     }
22 }
```

Conclusion and Outlook

Conclusion

We started this thesis with the ambition to develop a reduced model for aerothermal simulations. We divided this work in two natural parts.

In the first part, we exposed the numerical methods implemented for modeling aerothermal flows. We considered the incompressible Navier-Stokes equations coupled with an energy equation in the Boussinesq approximation. We used a standard Finite Element discretization with a Galerkin projection to numerically solve this system of PDEs. We introduced a Newton method to solve the non-linear discrete system. To improve the convergence capability of this iterative method, we also introduced a physical and a pseudo-transient continuation. These tools allow to efficiently reach the stationary state of the system if it exists.

We also implemented different versions of the Streamline Diffusion Method (SDM) to stabilize the numerical resolution. These methods have two aims: the first one is to deal with non-physical oscillations appearing in the resolution of convection dominated problems. The second objective is the stabilization of the numerical solution in the regions of anisotropic meshing. This kind of mesh is used to model the boundary layers of turbulent flows. To fulfill these two goals, we compared the efficiency of two SDM formulations: the Streamline-Upwind Petrov Galerkin (SUPG) and the Galerkin Least Square (GLS). These methods have been studied with different expressions for the stabilization parameter. We paid a special attention to the evaluation of the characteristic size of the cells for anisotropic meshes and we proposed an efficient formulation for simplex with large aspect ratios. The implemented methods were validated on multiple numerical applications.

The objective of the second part of this thesis was the reduction of the model described previously.

The model order reduction method studied was the Reduced Basis Method (RBM). This method is particularly adapted for the reduction of problems discretized with the FEM. It provides an efficient offline/online decomposition and a rapid convergence when used with efficient error estimators. We presented in this report our implementation of the RBM for saddle-point and trilinear problems. The numerical results are very encouraging for the reduced aerothermal model. We obtained a nice convergence of the reduced solutions although we did not have any error estimator. However, we noticed the limit of our model for complex geometries with important variation of the flow. For these kind of problems, our reduced model was not able to provide accurate solutions.

In order to reduce the stabilization operators presented in the first part, we proposed to use the Empirical Interpolation Method (EIM). We implemented a discrete version of this algorithm, allowing to recover an affine decomposition of parametrized matrices and vectors. This method was successfully applied to obtain an affine approximation of the Jacobian and the Residual used for the resolution of stabilized aerothermal problems with a Newton algorithm. We also presented a discrete version of the Simultaneous EIM and RB (SER) algorithm using this new discrete version of EIM. Using SER for the reduction of non-linear problems greatly reduced the cost of the offline procedure by replacing FEM approximations with RB approximation in the EIM greedy algorithm.

The reduction of the turbulence models, described in the first part, was not tackled by lack of time. Nevertheless, we exposed some ideas for the development of reduced RANS model involving the construction of a reduced space for the turbulent quantities.

We eventually provided a reduced aerothermal model. However, this model still needs important improvements.

Outlook

Apart from the issue with the turbulent model, the aerothermal framework is finally robust and efficient. Most of the future developments should concern the reduction method.

The first priority would concern the optimization of the online resolution. We highlighted two major issues with our numerical results. First, the assembly of the reduced operators becomes abnormally expensive when the dimension of the reduced spaces is growing. This issue leads to excessive online runtimes. This issue should be fixed by optimizing the data structure used for storing the reduced

operators. The second important issue concerns the assembly of the reduced tensors for the discrete EIM. Despite the use of a reduced interpolation space, this procedure is relatively expensive and is limiting the capabilities of the reduced models. Some optimizations are still possible. For instance, we could store the evaluation of the FE basis functions on all the quadrature points of the interpolation mesh.

Another important potential improvement is the implementation of error estimators for the aerothermal model. We observed some convergence issues in the numerical results which could be solved with a better selection of the parameters in the greedy algorithm. However, this point remains non-trivial and requires important theoretical and numerical efforts.

Finally, the ultimate outlook would be the complete development of a reduced turbulent model. This research subject is still very open in the community of the model order reduction and we hope that the contribution in this thesis can help somehow to advance on this problem.

Appendices

1 Adaptive Time Stepping and Benchmarks

As an alternative to the standard BDF time discretization, we implemented an adaptive time-stepping algorithm proposed by Kay and Al. in [54, 38]. We recall in this section the main ingredients of the methods and we give some results.

Considering the results on the 2D Turek Benchmark and the difficulty to scale our PCD preconditioner with this formulation, the method does not actually seem to be adapted to our problem. We nevertheless dedicated a section to this algorithm which was very interesting to study and implement.

We consider the problem described in chapter 2. We will not detail the formulation for the whole system but only for the Navier-Stokes part. The method is similar for the energy equation and we give the final equations in section 1.1

We introduce a discretization of the time interval $[0, t_f]$, $\{t_n\}_{n=0,\dots,N}$ with $t_0 = 0$ and $t_N = t_f$. We then define the time steps $\{k_n\}_{n=1,\dots,N}$ by $k_{n+1} = t_{n+1} - t_n$. We denote by $(\boldsymbol{u}^n, p^n, T^n)$ the velocity, pressure and temperature fields at iteration n.

1.1 Detailed formulation for Navier-Stokes

We consider the fluid equation in the system (6.25). We first decide to linearise the equation using an Oseen formulation with extrapolation of the convection velocity

$$\mathbf{w}^{n+1} = \left(1 + \frac{k_{n+1}}{k_n}\right)\mathbf{u}^n - \frac{k_{n+1}}{k_n}\mathbf{u}^{n-1}. \tag{1}$$

We choose a Crank Nicholson scheme for the time discretization. We find different propositions for the choice of the scheme in the literature but Crank Nicholson seams to produce the best results with this kind of time stepping algorithm. And so the weak formulation becomes

$$\begin{aligned} 2\rho(\frac{\boldsymbol{u}^{n+1}}{k_{n+1}}, \boldsymbol{v}) + \rho(\boldsymbol{w}^{n+1}\cdot\nabla\boldsymbol{u}^{n+1}, \boldsymbol{v}) + \mu(\nabla\boldsymbol{u}^{n+1}, \nabla\boldsymbol{v}) - (p^{n+1}, \nabla\cdot\boldsymbol{v}) - (q, \nabla\cdot\mathbf{u}^{n+1}) = \\ 2\rho(\frac{\boldsymbol{u}^{n}}{k_{n+1}}, v) + (\frac{\partial\mathbf{u}^n}{\partial t}, v) + (\mathbf{f^n}, v) \end{aligned} \tag{2}$$

where $f^n = \beta(T^n - T_0)\boldsymbol{g}$.

We introduce now the discrete acceleration given by $\mathbf{d}^n = \frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{k_{n+1}}$, $\mathbf{u}^{n+1} = \mathbf{u}^n + k_{n+1}\mathbf{d}^n$ and we rewrite the equation (2) with this new quantity,

$$\rho(\mathbf{d}^n, \boldsymbol{v}) + \rho k_{n+1}(\boldsymbol{w}^{n+1} \cdot \nabla \boldsymbol{d}^n, \boldsymbol{v}) + \mu k_{n+1}(\nabla \boldsymbol{d}^n, \nabla \boldsymbol{v}) - (p^{n+1}, \nabla \cdot \boldsymbol{v}) - (q, \nabla \cdot \boldsymbol{d}^n) =$$
$$\mu(\nabla u^n, \nabla \boldsymbol{v}) - \rho(\mathbf{w}^{n+1} \cdot \nabla u^n, \boldsymbol{v}) + \frac{1}{k_{n+1}}(q, \nabla \cdot \mathbf{u}^n) + (\frac{\partial \mathbf{u}^n}{\partial t}, \boldsymbol{v}) + (f, \boldsymbol{v}). \quad (3)$$

We recognize a convection diffusion equation very close to the usual Navier-Stokes weak formulation. But now we want to solve this equation in order to find a couple $(\boldsymbol{d}^n, p^{n+1}) \in V_h$. The solution strategy and especially the preconditioner choice will be discussed later.

Formulation for the Aerothermal System

With the same method we can rewrite our full system (6.25), introducing the rate of change of the temperature field $D^n = \frac{T^{n+1} - T^n}{k_{n+1}}$, the full aerothermal system now becomes, find $(d^n, p^{n+1}, D^n) \in X_h$ such that

$$\rho(\mathbf{d}^n, \boldsymbol{v}) + \rho k_{n+1}(\boldsymbol{w}^{n+1} \cdot \nabla \boldsymbol{d}^n, \boldsymbol{v}) + \mu k_{n+1}(\nabla \boldsymbol{d}^n, \nabla \boldsymbol{v}) - (p^{n+1}, \nabla \cdot \boldsymbol{v}) - (q, \nabla \cdot \boldsymbol{d}^n) =$$
$$(f^n, \boldsymbol{v}) - \mu(\nabla u^n, \nabla \boldsymbol{v}) - \rho(\mathbf{w}^{n+1} \cdot \nabla u^n, \boldsymbol{v}) + \frac{1}{k_{n+1}}(q, \nabla \cdot \mathbf{u}^n) + (\frac{\partial \mathbf{u}^n}{\partial t}, \boldsymbol{v})$$

$$(D^n, S) + \kappa k_{n+1}(\nabla D^n, \nabla S) + k_{n+1}(\boldsymbol{w}^{n+1} D^n, S) = -\kappa(\nabla T^n, \nabla S) - (\boldsymbol{w}^{n+1} T^n, S) + (\frac{\partial T^n}{\partial t}, S),$$

$$\forall (\boldsymbol{v}, q, S) \in X_h \quad (4)$$

Once we have solved the system (4) we can evaluate the different quantities required for the next step :

$$\mathbf{u}^{n+1} = \mathbf{u}^n + k_{n+1}\mathbf{d}^n \quad (5)$$
$$\frac{\partial \mathbf{u}^{n+1}}{\partial t} = 2\mathbf{d}^n - \frac{\partial u^n}{\partial t} \quad (6)$$
$$T^{n+1} = T^n + k_{n+1} D^n \quad (7)$$
$$\frac{\partial T^{n+1}}{\partial t} = 2D^n - \frac{\partial T^n}{\partial t} \quad (8)$$

1.2 Time step computation

The choice of the time step is based on comparison of two solution : the first one given by our resolution of (4) after the update (5), the second solution is given

by an explicit evaluation using a second order Adams-Baschforth scheme. This comparison is made for both velocity and temperature field.

First the AB solution are computed

$$\mathbf{u}_{AB2}^{n+1} = \mathbf{u}^n + \frac{k_{n+1}}{2}\left[(1+k_{n+1}/k_n)\frac{\partial \boldsymbol{u}^n}{\partial t} + \frac{k_{n+1}}{k_n}\frac{\partial \boldsymbol{u}^{n-1}}{\partial t}\right], \tag{9}$$

$$T_{AB2}^{n+1} = T^n + \frac{k_{n+1}}{2}\left[(1+k_{n+1}/k_n)\frac{\partial T^n}{\partial t} + \frac{k_{n+1}}{k_n}\frac{\partial T^{n-1}}{\partial t}\right]. \tag{10}$$

The errors between the solution and the AB extrapolation are computed using the formula

$$e_u^{n+1} = \frac{||\mathbf{u}^{n+1} - \mathbf{u}_{AB2}^{n+1}||}{3(1+k_{n+1}/k_n)}, \qquad e_T^{n+1} = \frac{||T^{n+1} - T_{AB2}^{n+1}||}{3(1+k_{n+1}/k_n)} \tag{11}$$

and then used to evaluate the next time step, taking in account the error on both velocity and temperature fields.

$$k_{n+2} = k_{n+1}\left(\frac{\varepsilon}{\sqrt{(e_T^{n+1})^2 + (e_T^{n+1})^2}}\right)^{1/3}. \tag{12}$$

Here the parameter ε is a tolerance used to act on the time step evolution. If this tolerance is too big the time step will grow too quickly and one will lose in accuracy. But with too small tolerance, the time step will not take off and then one loses the interest of this algorithm. We typically use a tolerance ε between 10^{-2} and 10^{-6}, this choice greatly depends of the physics of your problem.

Initialization of the algorithm

Since the algorithm is not self-starting, one has to compute acceptable initial state for the different fields of interest $(\boldsymbol{u}, p, T)$, for the discrete acceleration and for rate of change.

The first time step is computed by the resolution of at time $t_1 = k_1$. In practice we use very small value of k_1 (10^{-8} for instance). We start by the computation of $(\boldsymbol{u}^1, p^1, T^1)$ such that

$$\begin{aligned} \mu(\nabla \boldsymbol{u}^1, \nabla \boldsymbol{v}) - (p^1, \nabla \cdot \boldsymbol{v}) - (q, \nabla \cdot \boldsymbol{u}^1) &= 0, \\ \kappa(\nabla T^1, \nabla S) &= 0, \quad \forall (\boldsymbol{v}, q, T) \in V_h \times X_h. \end{aligned} \tag{13}$$

These initial solutions are then used to compute initial acceleration d^1 and rate

of change D^1 such that

$$\begin{aligned}\rho(\boldsymbol{d}^1,\boldsymbol{v})-(p^1,\nabla\cdot\boldsymbol{v})-(q,\nabla\cdot\boldsymbol{d}^1)&=-\mu(\nabla\boldsymbol{u}^1,\nabla\boldsymbol{v})-\rho(\boldsymbol{u}^1\cdot\nabla\boldsymbol{u}^1,\boldsymbol{v})\\(\frac{\partial D^1}{\partial t},S)&=-\kappa(\nabla T^1,S)-(\boldsymbol{u}^1\cdot\nabla T^1,S),\qquad\forall(\boldsymbol{v},q,S)\in V_h\times X_h\end{aligned}\tag{14}$$

For the second time step we first have to find (u^2,p^2) and T^2 such that

$$\begin{aligned}\frac{2}{k_2}(\boldsymbol{u}^2,\boldsymbol{v})+\mu(\nabla\boldsymbol{u}^2,\nabla\boldsymbol{v})+(\boldsymbol{w}^2\cdot\nabla\boldsymbol{u}^2,\boldsymbol{v})-(p^2,\nabla\cdot\boldsymbol{v})-(q,\nabla\cdot\boldsymbol{u}^2)&=\frac{2}{k_2}(\boldsymbol{u}^1,\boldsymbol{v})+(\boldsymbol{d}^1,\boldsymbol{v}),\\\frac{2}{k_2}(T^2,S)+\kappa(\nabla T^2,\nabla S)+(\boldsymbol{w}^2\cdot\nabla T^2,S)&=\frac{2}{k_2}(T^1,S)+(D^1,S),\\&\forall(\boldsymbol{v},q,S)\in V_h\times X_h.\end{aligned}\tag{15}$$

This allows to evaluate d^2 and D^2 with the formula

$$\begin{aligned}d^2&=\frac{2}{k_2}(\boldsymbol{u}^2-\boldsymbol{u}^1)-\boldsymbol{d}^1\\D^2&=\frac{2}{k_2}(T^2-T^1)-D^1.\end{aligned}\tag{16}$$

Since we now have the acceleration and the rate of change at time $t^2=2k_1$, we can start the algorithm. The value of $\boldsymbol{u}^2$ and T^2 will be replaced by using the update (4). And then the algorithm can begin with the resolution of (5) for the next time step.

Averaging algorithm

The method introduced in this section is prone to "ringing" effects especially with large tolerance ε or when we are close to a steady state. More details about this behavior can be found in [54, 38].

To avoid these oscillations effects, we chose to implement an averaging algorithm as proposed in [54]. This algorithm is invoked periodically, every n^* steps. For such a step, we first compute the next time step k_{n+1}, using the method described in section 1.2. We also compute the current velocity $\boldsymbol{u}^n$ and the current acceleration $\frac{\partial\mathbf{u}^n}{\partial t}$ with the formula (4).

The time step is then modified as follows

$$t_n=\frac{t_{n-1}+t^*}{2},\qquad\qquad k_n=\frac{k^*}{2}\tag{17}$$

and we can compute the new value for the fields $\boldsymbol{u}^n$ and $\frac{\partial\boldsymbol{u}}{\partial t}^n$

$$\boldsymbol{u}^n=\frac{\boldsymbol{u}^{n-1}+\boldsymbol{u}^*}{2},\qquad\qquad\frac{\partial\boldsymbol{u}}{\partial t}^n=\frac{1}{2}(\frac{\partial\boldsymbol{u}}{\partial t}^{n-1}+\frac{\partial\boldsymbol{u}}{\partial t}^*).\tag{18}$$

with

$$t^* = t_n, \qquad k^* = k_n, \qquad \boldsymbol{u}^* = \boldsymbol{u}^n, \qquad \frac{\partial \mathbf{u}^*}{\partial t} = \frac{\partial \mathbf{u}^n}{\partial t}. \tag{19}$$

The value of k_{n+1} choose previously keep unchanged and we can solve the next time step at $t_{n+1} = t_n + k_{n+1}$. We gave here the details of the averaging for the fluid part but the process is exactly the same for the thermal part of the system.

1.3 Turek Benchmarks

These benchmarks are proposed on http://www.featflow.de. Here we briefly recall the main parameters of the model. We then present our results obtained with our implementation of the framework presented in chapter 2.

The results on these two benchmarks were significant to evaluate the efficiency of our implementation.

Configuration

The parameters of the fluid are $\rho = 1.0$ and $\mu = 0.001$. The geometry of the model is a pipe without a circular cylinder of radius $r = 0.05$, as described in figure 2

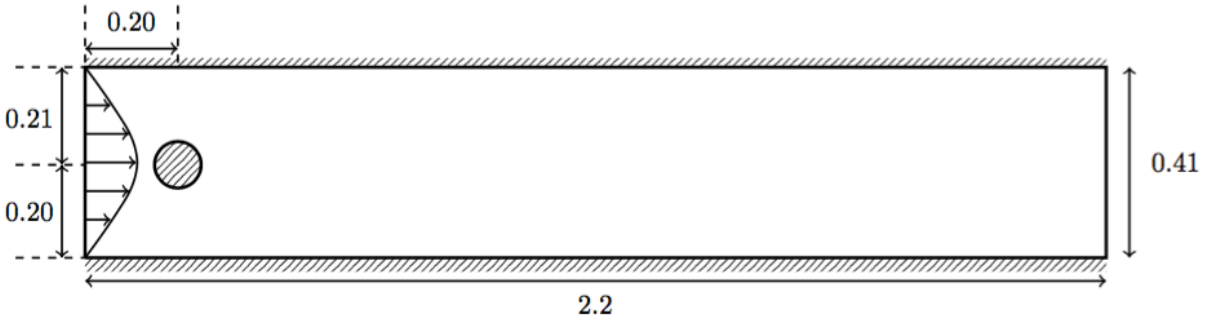


Figure 2 – Geometry of the Problem

No-slip boundary conditions are imposed on upper and lower walls and on the boundary of the cylinder. The left edge is the input of the pipe where a Poiseuille profile is prescribed,

$$u(0, y) = \left(\frac{4U_y(0.41 - y)}{0.41^2}, 0\right) \tag{20}$$

with the maximum amplitude of the velocity $U_y = 0.3$. The right edge is the output of the pipe and with a do-nothing condition

$$\bar{\sigma} = 0. \tag{21}$$

With this configuration, the Reynolds number is $\mathrm{Re} = 20$ and the system turns into stationary state.

Quantities of interest

In this benchmark we are interested in three quantities. All this quantities are measured in the steady state.

First we want to evaluate the drag and lift coefficients, respectively C_D and C_L, defined by

$$C_D = \frac{2}{\bar{U}^2 L} F_D \qquad\qquad C_L = \frac{2}{\bar{U}^2 L} F_L \tag{22}$$

with $\bar{U} = 0.2$ the mean velocity at the input, $L = 0.1$ the characteristic length of the flow configuration and (F_D, F_L) such that

$$(F_D, F_L) = \int_{\Gamma_C} \bar{\sigma} \boldsymbol{n} d\Gamma \tag{23}$$

where Γ_C is the boundary of the cylinder and $\boldsymbol{n}$ is the outer normal vector of the circle.

The last quantitity of interest is the pressure difference $\Delta_p = p(a_1) - p(a_2)$ between the points $a_1 = (0.15, 0.2)$ and $a_2 = (0.25, 0.2)$.

We will compare our results with the reference which gives $C_D = 5.57953523384$, $C_L = 0.010618948146$ and $\Delta_p = 0.11752016697$.

Mesh Convergence

We first try to find the best compromise between accuracy and computation cost for our mesh. We chose to use a mesh with 2 characteristic sizes: a default size h for the whole domain and a smaller size h_c applied on the cylinder. It results in a mesh particularly refined around the cylinder. An example of such a mesh is proposed in figure 3.

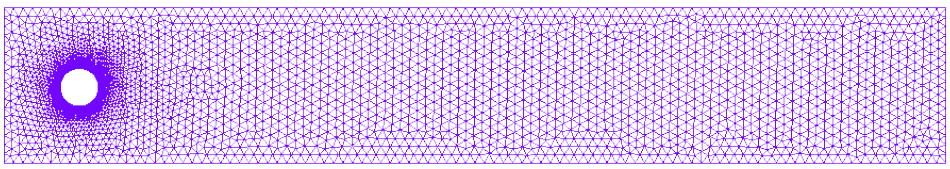


Figure 3 – Mesh used for the simulation

To find the best values for h and h_c we ran many simulation and compare the results for C_D, C_L and Δ_p with the reference values for different value of (h, h_c). The errors in percent, with respect to the reference results, are reported in tables 4, 5 and 6.

$h \backslash h_c$	0.1	0.05	0.025	0.01	0.005	0.00025	0.001
0.1	9.72	9.72	5.14	1.69	0.60	0.26	0.17
0.05		9.20	4.94	1.51	0.48	0.17	0.06
0.025			4.51	1.54	9.47	0.16	0.05

Figure 4 – C_D error for different values of h and h_c (in %)

$h \backslash h_c$	0.1	0.05	0.025	0.01	0.005	0.00025	0.001
0.1	407.17	407.17	71.95	149.20	63.77	11.58	77.98
0.05		2118.51	160.67	1.00	12.28	1.83	3.15
0.025			71.02	13.18	12.12	0.43	0.09

Figure 5 – C_L error for different values of h and h_c (in %)

$h \backslash h_c$	0.1	0.05	0.025	0.01	0.005	0.00025	0.001
0.1	1.33	1.33	0.21	1.07	0.88	0.71	0.61
0.05		0.74	0.28	0.10	0.14	0.06	0.05
0.025			1.04	0.03	0.02	0.02	0.01

Figure 6 – Δ_p error for different values of h and h_c (in %)

Reference Case

We are not only interested in the validation of our algorithms for the steady state but also for the transient state. For that we start by running a simulation with a very small time step. The result will be our reference to know if we are accurate enough in the transient state.

This reference simulation is ran on the mesh described in section 1.3 with a time step $\delta t = 0.01$ and an order 2 BDF formulation.

The results over time for C_D, C_L and Δ_p are presented in the figure 7. We consider that the system is in a steady state once the difference $\frac{||u^{n+1}-u^n||}{||u^n||}$ is smaller than a chosen tolerance $\varepsilon_{\text{steady}}$. For this simulation we choose $\varepsilon_{\text{steady}} = 10^{-5}$ since we got very good results for all the quantities of interest with this tolerance.

Considering these results and the results from the mesh convergence study, it appears that the lift coefficient (C_L) is the more sensitive and so we decided to focus on this quantity to find the optimal time-step (BDF) and tolerance (CNAB2) for this simulation.

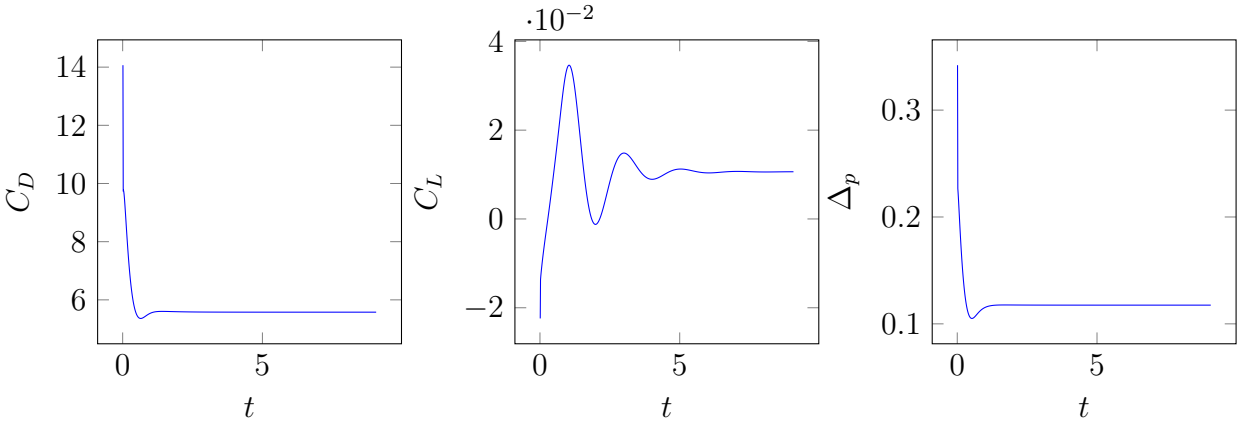


Figure 7 – Evolution of the 3 quantities of interest over time for the reference case

BDF : time step choice

We tried 3 time steps for BDF algorithm, $\delta t = 0.1, 0.25, 0.5$. The results are summarized in figure 8 . If both time step $\delta t = 0.05$ and $\delta t = 0.1$ give acceptable results, it is clear that the transient state is not well modeled with bigger time step. However we note that the system still converge to the steady state with $\delta t = 0.25$ but it takes much more time.

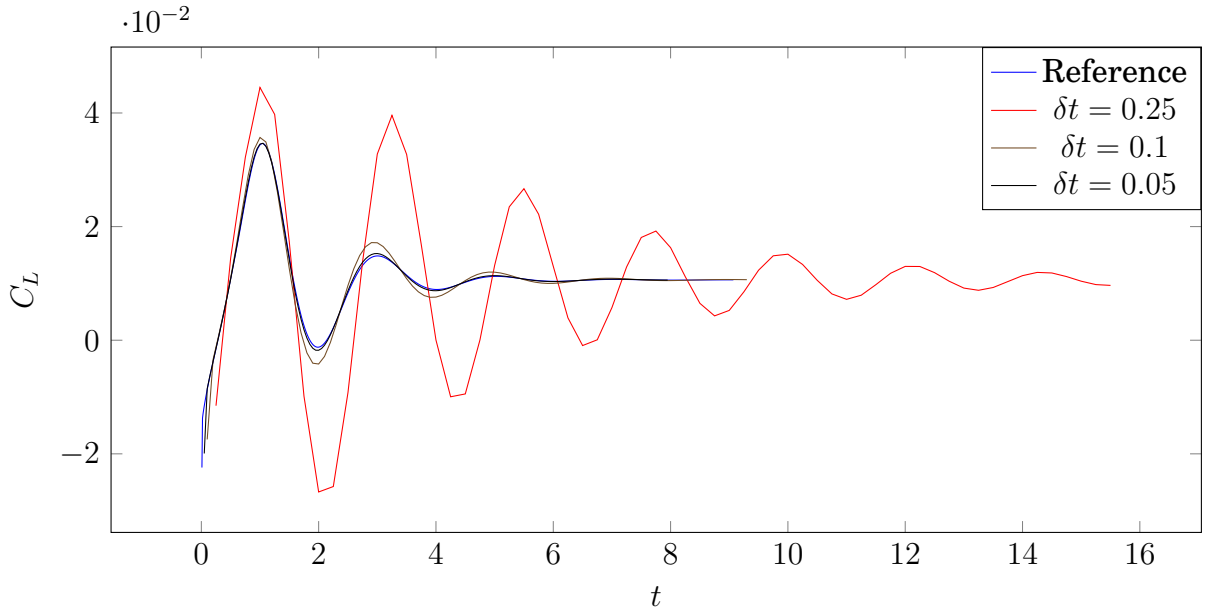


Figure 8 – Comparison of the lift coeeficient for 3 diffent time steps

CNAB2 : configuration choice

For CNAB2 algorithm, we tried 3 different tolerances, $\varepsilon = 10^{-5}, Re^{-5}, 10^{-4}$. But we had some issue with the growing time step and the CN scheme. When time step

becomes to big (typically when you are close to the steady state), the scheme loses in accuracy and generates some non-physics oscillations. To avoid theses oscillations we had to use a fixed algorithm in the temporal iterations. We iterate until the quantity $\frac{||u^{n+1,i+1}-u^{n+1,i}||}{||u^{n+1,i}||}$ is smaller than the tolerance ε_{pic} where $u^{n+1,i}$ is the solution after the i-th fixed point iteration. We fix $u^{n+1,0} = w^{n+1}$ as defined in (1).

We tried 2 values for this tolerance, $\varepsilon_{\text{pic}} = 10^{-4}, 10^{-5}$. The results are presented figures 9 and 10

We finally chose to keep the configuration with $\varepsilon = 10^{-4}$ which was the most efficient in terms of simulation time and number of resolutions.

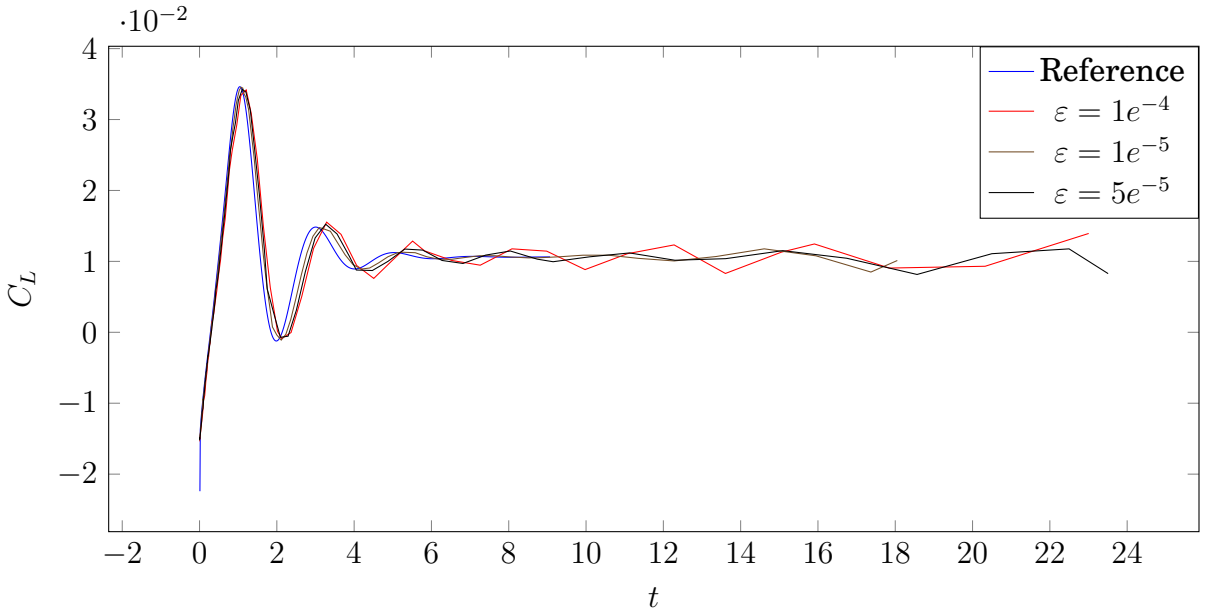


Figure 9 – Comparison of the lift coefficient for 3 different tolerances ε with a Picard tolerance $\varepsilon = 10^{-4}$

Numerical Results

In the table 11 are our results and the error compared to the reference. We also registered the total time of the simulation t_{simu}, the number of time-steps n_{steps} and the number of resolutions of the Naviers-Stokes system n_{NS}.

We ran all theses simulations on 4 cores. The number of DoF is 37797×4862. The BDF case correspond to a usual second order backward differentiation formula with a constant time step chosen as described in section 1.3. The CNAB2 case corresponds to the adaptive time stepping algorithm presented in chapter 2 with the configuration chosen previously, in section 1.3, with $n^* = 10$.

In both cases we iterate in time until the quantity Δ_u becomes smaller than a

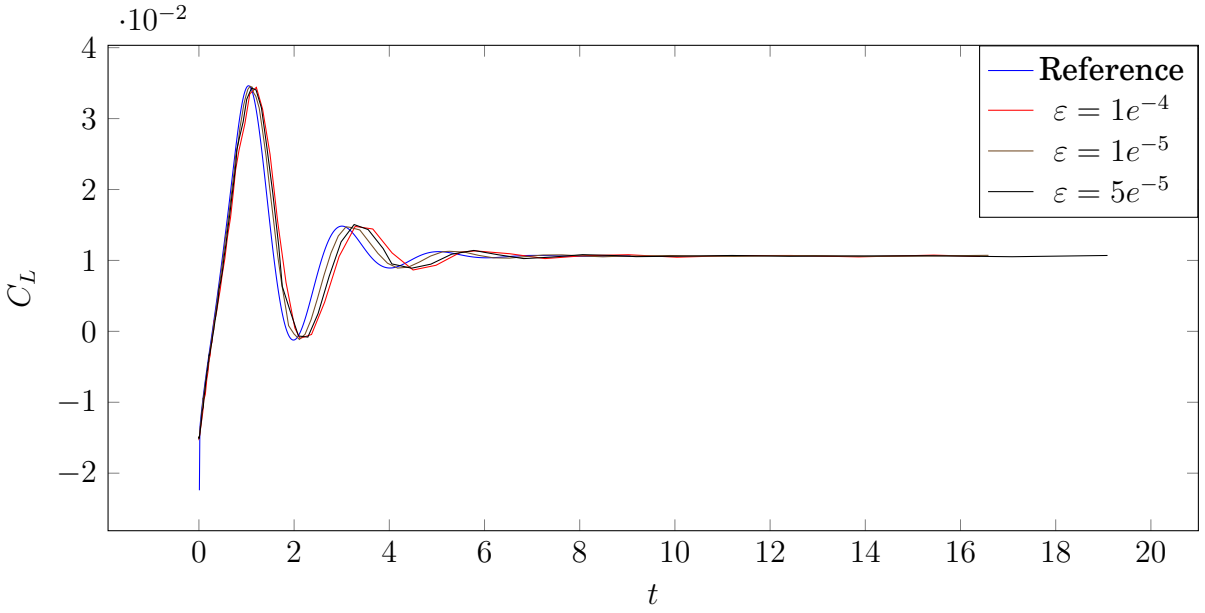


Figure 10 – Comparison of the lift coefficient for 3 different tolerances ε with a picard tolerance $\varepsilon = 10^{-5}$

tolerance $\varepsilon_{\text{steady}}$. In practice we choose $\varepsilon_{\text{steady}} = 10^{-4}$ and Δ_u is defined as

$$\Delta_u = \frac{||u^{n+1} - u^n||}{||u^n||}. \tag{24}$$

When using iterative solver (PCD), we typically set the tolerance for the relative error between two KSP iterations to rtol$= 10^{-9}$.

	BDF				CNAB2			
	LU		PCD		LU		PCD	
C_D	5.57694	[0.047]	5.57694	[0.047]	5.57694	[0.047]	5.56076	[0.33]
C_L	0.0106564	[0.35]	0.0106567	[0.35]	0.01056	[0.35]	0.0109929	[3.52]
Δ_p	0.117512	[0.007]	0.117512	[0.007]	0.117512	[0.007]	0.11711	[0.34]
t_{simu}	124s		242s		184s		546s	
n_{step}	93		93		61		61	
n_{NS}	93		93		147		152	

Figure 11 – Summary of benchmark results for different configurations, [error w.r.t. reference in %]

Turek 2D, $Re = 100$

The benchmark configuration is exactly the same than in section 1.3. We consider the same equations, fluid parameters and geometry, see figure 2.

The only parameter which is different in this configuration is the inflow velocity

profile. We still consider a parabolic profile,

$$u(0,y) = (\frac{4U_y(0.41-y)}{0.41^2}, 0) \tag{25}$$

but impose this time a maximum velocity $U_y = 1.5$. and so the mean velocity is $\bar{U} = 1$ and it leads to a Reynolds number $\mathrm{Re} = 100$.

For this Reynolds number, the flow turns into a time-periodic behavior with a vortex shedding behind the cylinder. See http://www.featflow.de for more details.

Data Measurement

The measure of the following data is made during a fully developed time-periodic solution. For that we wait for 25 seconds simulation, at this time the flow is fully developed. Then we take an arbitrary cycle between 25 and 30 seconds of simulation times to measure numerical data.

We are first interested in the value of C_L, C_D and Δ_p as defined in section 4.1. For these quantities we will compare the max, the min, the amplitude and the mean value against the reference results.

We also define f the frequency of the periodic flow. If a cycle starts at time t_0 where the lift coefficient C_L is the smallest, it ends at time $t1 = t0 + 1/f$ when C_L is smallest again. $1/f$ is then the length of the cycle. With this definition of the frequency, we can define the Strouhal number

$$\mathrm{St} = \frac{Lf}{\bar{U}} \tag{26}$$

Our measure for f and St will also be compared with the reference data.

For this benchmark we only used direct solver with LU resolution. PCD may be validated later on 3D benchmark. Our results are presented in table 12 for different meshes, time step or tolerance. In red are the non acceptable values of the measure with respect to the reference bounds recalled in table 13.

	BDF					CNAB2		
	$\delta t = 0.01$			0.005	0.0025	$\varepsilon = 10^{-4}$	10^{-5}	10^{-6}
h	0.025	0.01	0.005	0.01	0.01	0.01	0.01	0.01
$C_{D\max}$	3.2699	3.2722	3.2724	3.2389	3.2302	3.2333	3.2273	3.2268
$C_{L\max}$	1.0655	1.0666	1.0666	1.0071	0.99173	0.99391	0.98453	0.98473
Δ_p	2.50229	2.50718	2.5074	2.49234	2.48609	2.43213	2.48715	2.48501
St	0.31250	0.30303	0.30303	0.30303	0.30303	0.30239	0.30120	0.30130
n_{iter}	3000	3000	3000	6000	12000	6504	14218	30925
t_{simu}		10100s		19800s		28000s		135000s

Figure 12 – Results for turek 2D benchmark, $\mathrm{Re} = 100$

$C_{D\max}$	3.22 to 3.24
$C_{L\max}$	0.99 to 1.01
Δ_p	2.46 to 2.5
St	0.295 to 0.305

Figure 13 – Reference Bounds for the quantities of interest

2 Nitsche's Method for Slip Condition

We consider the system

$$\begin{cases} \boldsymbol{u}\cdot\nabla\boldsymbol{u}+\nabla p-\nabla\cdot(\mu\nabla\boldsymbol{u})=f, \\ \nabla\cdot\boldsymbol{u}=0, \\ \boldsymbol{u}\cdot\boldsymbol{n}=g_{\boldsymbol{n}}, \quad \text{on } \Gamma \\ (\boldsymbol{T}(p,\boldsymbol{u})\boldsymbol{n})_{\boldsymbol{\tau}}=\boldsymbol{g}_{\boldsymbol{\tau}}, \quad \text{on } \Gamma \end{cases} \tag{27}$$

For any vector $\boldsymbol{u}$ we denote by $(\boldsymbol{u})_{\boldsymbol{n}}=(\boldsymbol{u}\cdot\boldsymbol{n})\boldsymbol{n}$ the component of $\boldsymbol{u}$ orthogonal to Γ and $(\boldsymbol{u})_{\boldsymbol{\tau}}=\boldsymbol{u}-(\boldsymbol{u})_{\boldsymbol{n}}$ the component of $\boldsymbol{u}$ tangential to Γ.

We apply the usual method, detailed in section 2, to recover the following weak discrete formulation,

$$\begin{aligned} \int_\Omega(\boldsymbol{u}_h\cdot\nabla)\boldsymbol{u}_h\cdot\boldsymbol{v}_h\,d\Omega+\int_\Omega\mu\nabla\boldsymbol{u}_h:\nabla\boldsymbol{v}_h\,d\Omega-\int_\Omega p_h\nabla\cdot\boldsymbol{v}_h\,d\Omega-\alpha\int_\Omega q_h\nabla\cdot\boldsymbol{u}_h\,d\Omega \\ -\int_\Gamma\boldsymbol{T}(\boldsymbol{u}_h,p_h)\boldsymbol{n}\cdot\boldsymbol{v}_h\,d\Gamma=\int_\Omega\boldsymbol{f}\cdot\boldsymbol{v}_h, \quad \forall(q,\boldsymbol{v}_h)\in V_h\times Q_h. \end{aligned} \tag{28}$$

We rewrite the term on Γ using the normal/tangential decomposition of the vectors

$$\int_\Gamma\boldsymbol{T}(\boldsymbol{u}_h,p_h)\boldsymbol{n}\cdot\boldsymbol{v}_h\,d\Gamma=\int_\Gamma(\boldsymbol{T}(\boldsymbol{u}_h,p_h)\boldsymbol{n})_{\boldsymbol{n}}\cdot(\boldsymbol{v}_h)_{\boldsymbol{n}}\,d\Gamma+\int_\Gamma(\boldsymbol{T}(\boldsymbol{u}_h,p_h)\boldsymbol{n})_{\boldsymbol{\tau}}\cdot(\boldsymbol{v}_h)_{\boldsymbol{\tau}}\,d\Gamma\,, \tag{29}$$

and we can now apply the boundary condition on $(\boldsymbol{T}(\boldsymbol{u}_h,p_h))_{\boldsymbol{\tau}}$ given by (27). We also add the symmetric/antisymmetric term for the normal component (depending of the value of δ) and we finally add a penalization terms to control the value of $\boldsymbol{u}_h\cdot\boldsymbol{n}$,

$$\begin{aligned} \int_\Gamma\boldsymbol{T}(\boldsymbol{u}_h,p_h)\boldsymbol{n}\cdot\boldsymbol{v}_h\,d\Gamma=\int_\Gamma(\boldsymbol{T}(\boldsymbol{u}_h,p_h)\boldsymbol{n})_{\boldsymbol{n}}\cdot(\boldsymbol{v}_h)_{\boldsymbol{n}}\,d\Gamma+\delta\int_\Gamma(\boldsymbol{T}(\boldsymbol{v}_h,q_h)\boldsymbol{n})_{\boldsymbol{n}}\cdot(\boldsymbol{u}_h\cdot\boldsymbol{n}-g_{\boldsymbol{n}})\boldsymbol{n}\,d\Gamma \\ -\gamma\int_\Gamma\frac{1}{h_S}(\boldsymbol{u}_h\cdot\boldsymbol{n}-g_n)\boldsymbol{v}_h\cdot\boldsymbol{n}\,d\Gamma+\int_\Gamma\boldsymbol{g}_{\boldsymbol{\tau}}\cdot(\boldsymbol{v}_h)_{\boldsymbol{\tau}}\,d\Gamma\,, \end{aligned} \tag{30}$$

where $\delta=\pm1$, h_S is the measure of the surface of the current element and γ is a positive constant. The choice of the different constants (α,δ,γ) and its consequences will be discussed in the following subsection.

Introducing the expression (30) in equation (28), we obtain the new weak formulation

$$\begin{aligned} &\int_\Omega(\boldsymbol{u}_h\cdot\nabla)\boldsymbol{u}_h\cdot\boldsymbol{v}_h\,d\Omega+\int_\Omega\mu\nabla\boldsymbol{u}_h:\nabla\boldsymbol{v}_h\,d\Omega-\int_\Omega p_h\nabla\cdot\boldsymbol{v}_h\,d\Omega-\alpha\int_\Omega q_h\nabla\cdot\boldsymbol{u}_h\,d\Omega \\ &-\int_\Gamma(\boldsymbol{T}(\boldsymbol{u}_h,p_h)\boldsymbol{n})_{\boldsymbol{n}}\cdot(\boldsymbol{v}_h)_{\boldsymbol{n}}\,d\Gamma-\delta\int_\Gamma(\boldsymbol{T}(\boldsymbol{v}_h,q_h)\boldsymbol{n})_{\boldsymbol{n}}\cdot(\boldsymbol{u}_h)_{\boldsymbol{n}}\,d\Gamma+\gamma\int_\Gamma\frac{1}{h_S}(\boldsymbol{u}_h\cdot\boldsymbol{n})(\boldsymbol{v}_h\cdot\boldsymbol{n})\,d\Gamma \\ &=-\delta\int_\Gamma(\boldsymbol{T}(\boldsymbol{v}_h,q_h)\boldsymbol{n})_{\boldsymbol{n}}\cdot g_{\boldsymbol{n}}\boldsymbol{n}\,d\Gamma+\gamma\int_\Gamma\frac{1}{h_S}g_n\boldsymbol{v}_h\cdot\boldsymbol{n}\,d\Gamma+\int_\Gamma\boldsymbol{g}_{\boldsymbol{\tau}}\cdot(\boldsymbol{v}_h)_{\boldsymbol{\tau}}\,d\Gamma+\int_\Omega\boldsymbol{f}\cdot\boldsymbol{v}_h, \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\forall(q,\boldsymbol{v}_h)\in V_h\times Q_h. \end{aligned} \tag{31}$$

2.1 Validation

Convergence Study with Stokes Equations

We first consider a square $\Omega = [-1, 1] \times [-1, 1]$ with a non structured triangular mesh of characteristic size h. On this domain, with impose an exact solution to the Stokes equations with $\mu = 1$, on all the boundary Γ. As a comparison we first impose this condition with strong Dirichlet (DIR) condition and then using the penalization method for slip condition (SLIP), as written in (31). The exact solution imposed is $\boldsymbol{u}_e = (2y(1 - x^2), -2x(1 - y^2))$. With this solution we have

$$(\boldsymbol{T}(\boldsymbol{u}_e, p_e)\boldsymbol{n})_{\boldsymbol{\tau}} = \boldsymbol{u}_e = (2y(1 - x^2), -2x(1 - y^2)) \tag{32}$$

on the boundary of the domain. We then use the formulation (31) with

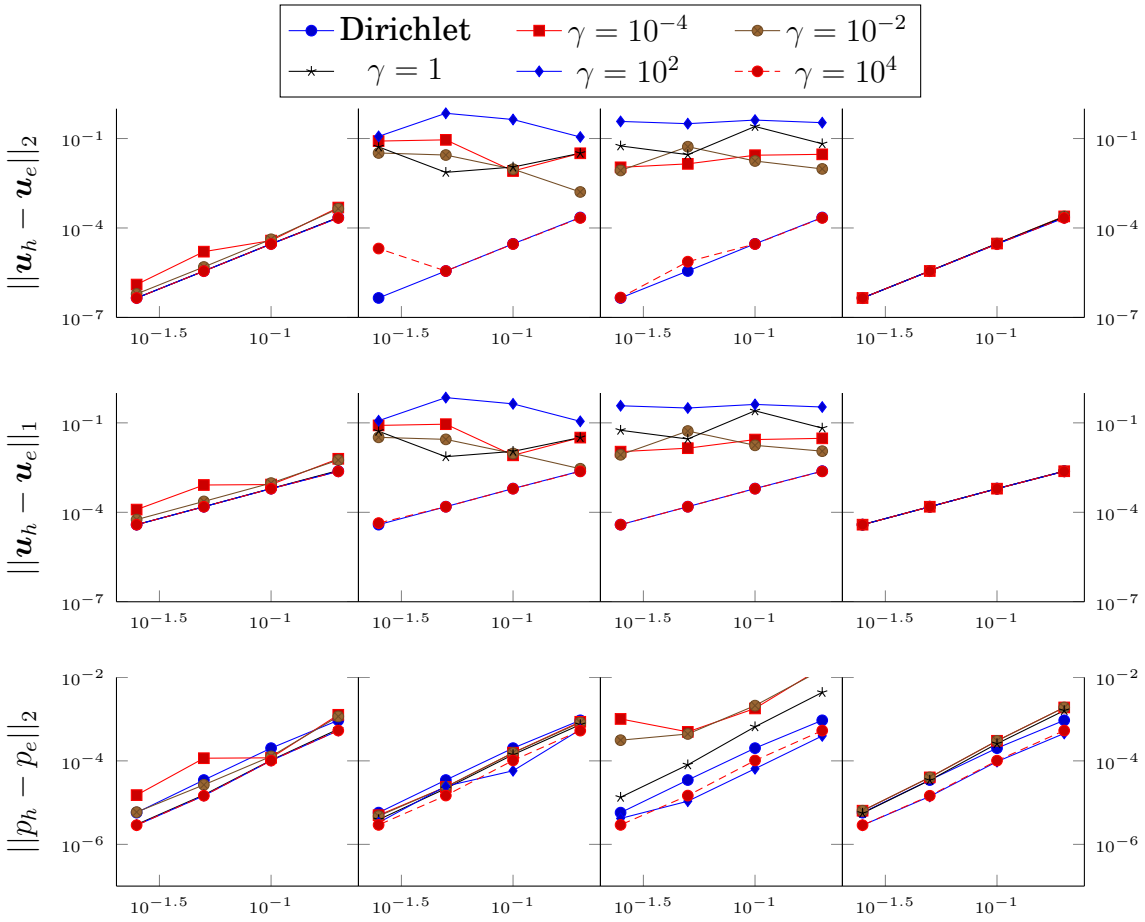
$$g_{\boldsymbol{n}} = 0, \qquad \boldsymbol{g}_\tau = (2y(1 - x^2), -2x(1 - y^2)), \tag{33}$$

and the source term f is chosen in order to have $p_2 = 0$. The discrete solution is computed for different values of $h = 0.2, 0.1, 0.05, 0.025$ and for the 4 different possibilities for the couple (α, δ). We also tested different value of γ. These results are summarized in the figure 14. According to these results, only two configurations have the good convergence rates : $\alpha = 1, \delta = 1$ and $\alpha = -1, \delta = -1$, with a slight accuracy advantage for this last case. In the following convergence studies we will only consider the configurations $\alpha = \delta$.

The second convergence test is made on a ring, the domain is defined by $1 \leq x^2 + y^2 \leq 4$. The exact solution $u_e = (-y\sqrt{x^2 + y^2}, x\sqrt{x^2 + y^2})$ is imposed strongly on the inside boundary ($x^2 + y^2 = 1$) and with slip conditions on the outside boundary ($x^2 + y^2 = 4$). With this exact solution we can evaluate

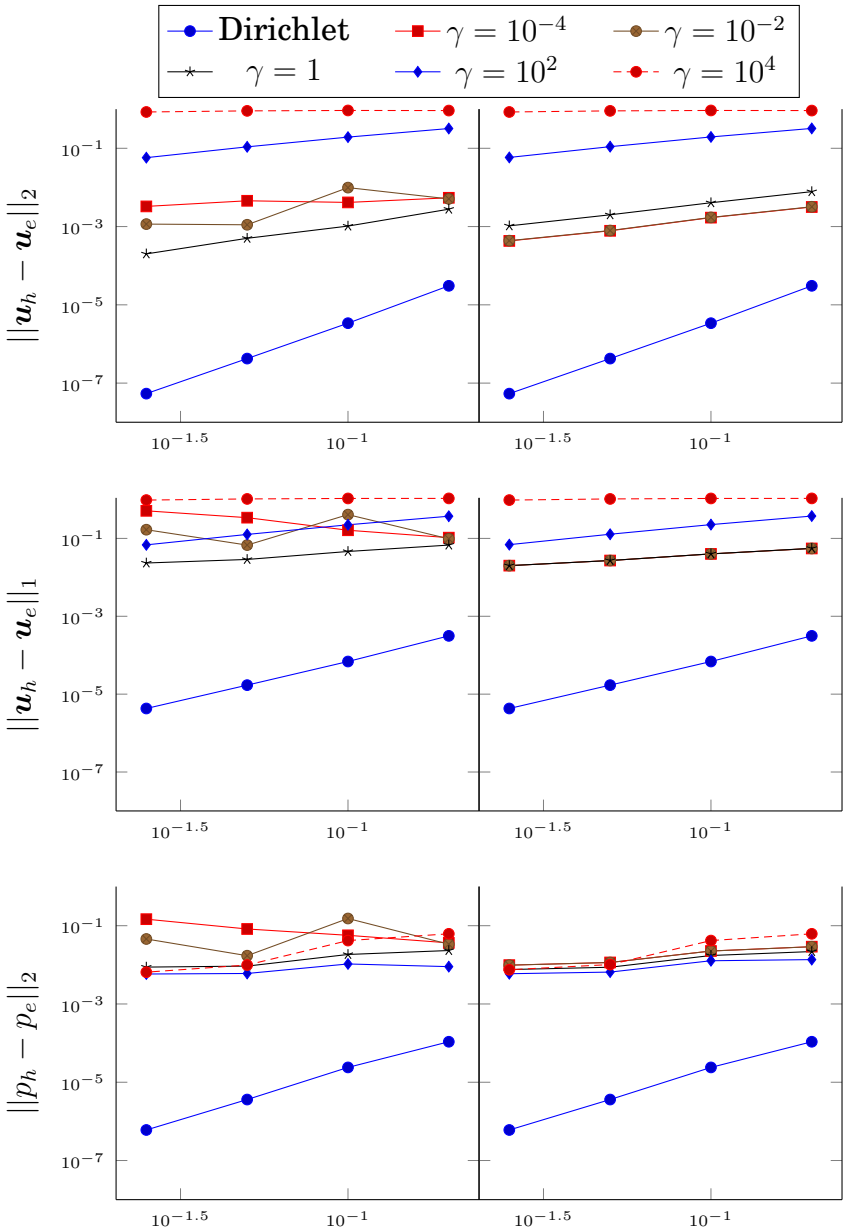
$$g_{\boldsymbol{n}} = 0, \qquad \boldsymbol{g}_\tau = (\boldsymbol{T}(\boldsymbol{u}_e, p_e)\boldsymbol{n})_{\boldsymbol{\tau}} = (-2y, 2x). \tag{34}$$

Again we choose $p = 0$ and the source term is computed to fit with these parameters.



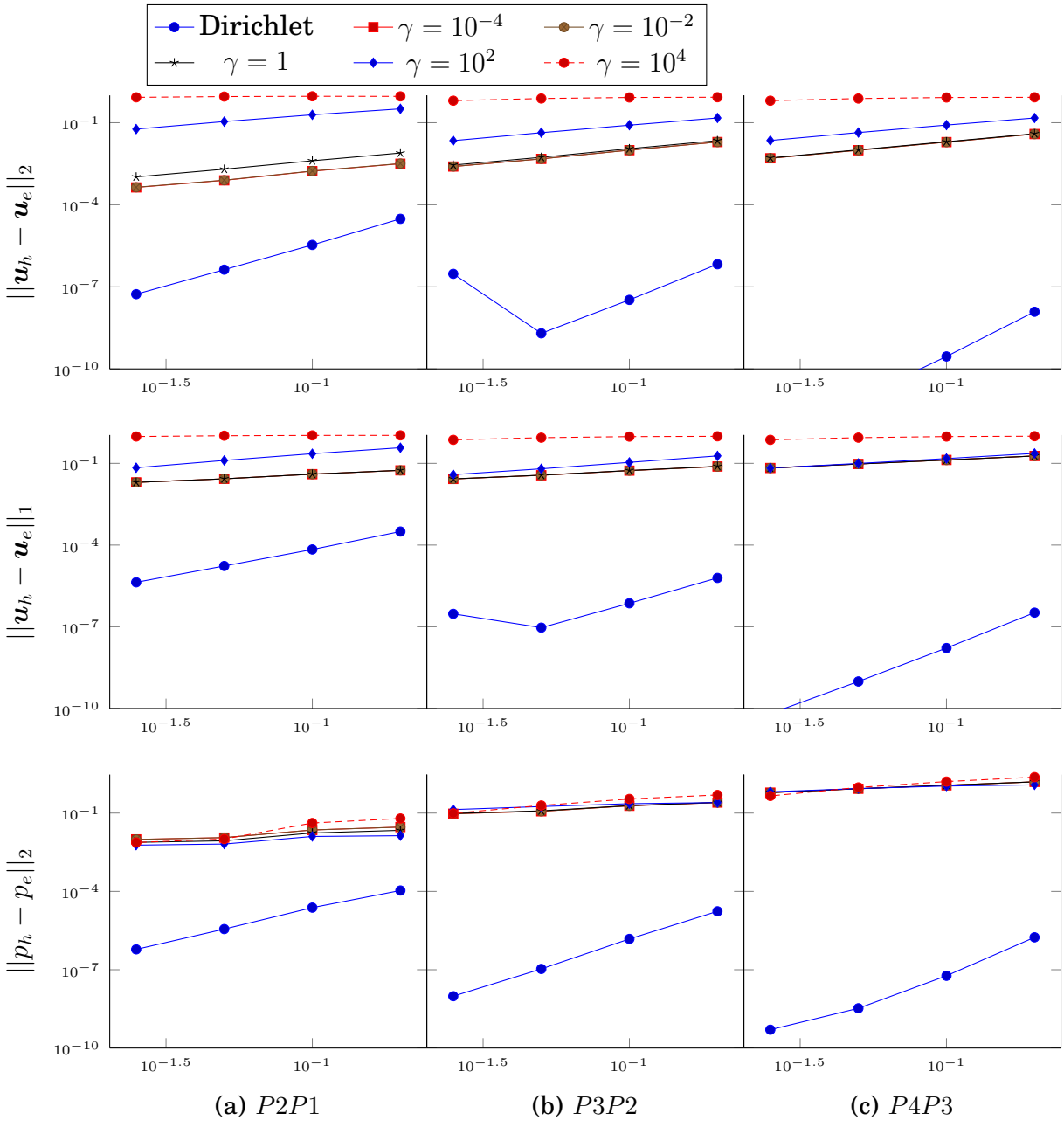
(a) $\alpha = 1, \delta = 1$ (b) $\alpha = 1, \delta = -1$ (c) $\alpha = -1, \delta = 1$ (d) $\alpha = -1, \delta = -1$

Figure 14 – Convergence study on square test case



(a) $\alpha = 1, \delta = 1$ (b) $\alpha = -1, \delta = -1$

Figure 15 – Convergence study on ring test case



(a) $P2P1$ (b) $P3P2$ (c) $P4P3$

Figure 16 – Convergence study on ring test case G1

3 Offline/Online Strategy for Residual Evaluation in the CRB

3.1 Coercive Problems

The residual defined in (5.22) can be rewritten using the affine decomposition of the problem

$$r_N(v;\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \theta_q^f(\boldsymbol{\mu}) f_q(v) - \sum_{q=1}^{Q_a} \theta_q^a a_q(u_N(\boldsymbol{\mu});\boldsymbol{\mu}) \tag{35}$$

and then the Riesz representation satisfies

$$(\hat{e}_N(\boldsymbol{\mu}), v)_{\bar{\boldsymbol{\mu}}} = \sum_{q=1}^{Q_f} \theta_q^f(\boldsymbol{\mu}) f_q(v) - \sum_{q=1}^{Q_a} \theta_q^a a_q(u_N(\boldsymbol{\mu});\boldsymbol{\mu}), \quad \forall v \in X_h. \tag{36}$$

Now using the projection of $u_N(\boldsymbol{\mu})$ on the RB space, as in (5.12), we obtain

$$(\hat{e}_N(\boldsymbol{\mu}), v)_{\bar{\boldsymbol{\mu}}} = \sum_{q=1}^{Q_f} \theta_q^f(\boldsymbol{\mu}) f_q(v) - \sum_{q=1}^{Q_a} \sum_{i=1}^{N} \theta_q^a u_{Ni}(\boldsymbol{\mu}) a_q(\xi_i;\boldsymbol{\mu}), \quad \forall v \in X_h. \tag{37}$$

And then, using the linear superposition principle, we can write $\hat{e}_N(\boldsymbol{\mu})$ as

$$\hat{e}_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \theta_q^f(\boldsymbol{\mu}) \hat{f}_q - \sum_{q=1}^{Q_a} \sum_{i=1}^{N} \theta_q^a u_{Ni}(\boldsymbol{\mu}) \hat{a}_{qi} \tag{38}$$

where the Riesz representations $\hat{f}_q$ and $\hat{a}_{qi}$ satisfy

$$(\hat{f}_q, v)_{\bar{\boldsymbol{\mu}}} = f_q(v), \qquad \forall v \in X_h, \quad 1 \le q \le Q_f, \tag{39}$$

$$(\hat{a}_{qi}, v)_{\bar{\boldsymbol{\mu}}} = a_q(\xi_i, v), \qquad \forall v \in X_h, \quad 1 \le i \le N, \quad 1 \le q \le Q_a. \tag{40}$$

Finally we can write

$$||\hat{e}_N(\boldsymbol{\mu})||^2_{\bar{\boldsymbol{\mu}}} = C^{ff}(\boldsymbol{\mu}) + \sum_{i=1}^{N}\sum_{j=1}^{N} (u_N(\boldsymbol{\mu}))_i (C_n^{aa}(\boldsymbol{\mu}))_{ij} (u_N(\boldsymbol{\mu}))_j - 2\sum_{i=1}^{N} (u_N(\boldsymbol{\mu}))_i (C_N^{fa}(\boldsymbol{\mu}))_i \tag{41}$$

$$= C^{ff}(\boldsymbol{\mu}) + u_N(\boldsymbol{\mu})^\intercal C_N^{aa}(\boldsymbol{\mu}) u_N(\boldsymbol{\mu}) - 2u_N(\boldsymbol{\mu})^\intercal C_N^{fa}(\boldsymbol{\mu}) \tag{42}$$

with,

$$C^{ff}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f}\sum_{q'=1}^{Q_f} \theta_q^f(\boldsymbol{\mu}) \theta_{q'}^f(\boldsymbol{\mu}) \underbrace{(\hat{f}_q, \hat{f}_{q'})_{\bar{\boldsymbol{\mu}}}}_{\text{precomputed}}, \tag{43}$$

$$C_N^{fa}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f}\sum_{q'=1}^{Q_a} \theta_q^f(\boldsymbol{\mu})\theta_{q'}^a(\boldsymbol{\mu}) \underbrace{C_{qq',N}^{fa}}_{\text{precomputed}}, \qquad (C_{qq',N}^{fa})_i = (\hat{f}_q, \hat{a}_{q'i})_{\bar{\boldsymbol{\mu}}}, \quad 1 \le i \le N, \tag{44}$$

$$C_N^{aa}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a}\sum_{q'=1}^{Q_a} \theta_q^a(\boldsymbol{\mu})\theta_{q'}^a(\boldsymbol{\mu}) \underbrace{C_{qq',N}^{aa}}_{\text{precomputed}}, \qquad (C_{qq',N}^{aa})_{ij} = (\hat{a}_{qi}, \hat{a}_{q'j})_{\bar{\boldsymbol{\mu}}}, \quad 1 \le i,j \le N. \tag{45}$$

The precomputed quantities are evaluated once during the online procedure and allows a efficient online evaluation, independent of the finite element dimension $\mathcal{N}$.

Using the same method, we can express Riesz representation norm of the residual for the dual problem as

$$||\hat{e}_N^{du}(\boldsymbol{\mu})||_{\bar{\boldsymbol{\mu}}}^2 = C^{ll}(\boldsymbol{\mu}) + \psi_N(\boldsymbol{\mu})^\intercal C_N^{aa}(\boldsymbol{\mu})\psi_N(\boldsymbol{\mu}) + 2\psi_N(\boldsymbol{\mu})^\intercal C_N^{la}(\boldsymbol{\mu}) \tag{46}$$

3.2 Saddle-Point Problems

Just like in the coercive case, we start by introducing the Riesz representation of each residual which satisfy

$$\begin{aligned} (\hat{r}_N^{\boldsymbol{u}}(\boldsymbol{\mu}), \boldsymbol{v})_V &= r_N^{\boldsymbol{u}}(\boldsymbol{v};\boldsymbol{\mu}), \quad \forall \boldsymbol{v} \in V_h \\ (\hat{r}_N^{p}(\boldsymbol{\mu}), q)_Q &= r_N^{p}(q;\boldsymbol{\mu}), \quad \forall q \in Q_h \end{aligned} \tag{47}$$

Using the affine decomposition (5.39) of the forms we can rewrite the relation (47) as,

$$\begin{aligned} (\hat{r}_N^{\boldsymbol{u}}(\boldsymbol{\mu}), \boldsymbol{v})_V &= \sum_{k=1}^{Q_f} \theta_k^f(\boldsymbol{\mu}) f_k(\boldsymbol{v}) - \sum_{k=1}^{Q_a} \theta_k^a(\boldsymbol{\mu}) a_k(\boldsymbol{u}_N(\boldsymbol{\mu}), \boldsymbol{v}) - \sum_{k=1}^{Q_b} \theta_k^b(\boldsymbol{\mu}) b_k(\boldsymbol{v}, p_N(\boldsymbol{\mu})), \\ (\hat{r}_N^{p}(\boldsymbol{\mu}), q)_Q &= \sum_{k=1}^{Q_g} \theta_k^g(\boldsymbol{\mu}) g_k(q) - \sum_{k=1}^{Q_b} \theta_k^b(\boldsymbol{\mu}) b_k(\boldsymbol{u}_N(\boldsymbol{\mu}), q). \end{aligned} \tag{48}$$

Now with the projection on the RB spaces we can write

$$\begin{aligned} (\hat{r}_N^{\boldsymbol{u}}(\boldsymbol{\mu}), \boldsymbol{v})_V &= \sum_{k=1}^{Q_f} \theta_k^f(\boldsymbol{\mu}) f_k(\boldsymbol{v}) - \sum_{k=1}^{Q_a}\sum_{i=1}^{2N} \theta_k^a(\boldsymbol{\mu}) u_{Ni}(\boldsymbol{\mu}) a_k(\boldsymbol{\zeta}_i, \boldsymbol{v}) - \sum_{k=1}^{Q_b}\sum_{i=1}^{N} \theta_k^b(\boldsymbol{\mu}) p_{Ni}(\boldsymbol{\mu}) b_k(\boldsymbol{v}, \eta_i), \\ (\hat{r}_N^{p}(\boldsymbol{\mu}), q)_Q &= \sum_{k=1}^{Q_g} \theta_k^g(\boldsymbol{\mu}) g_k(q) - \sum_{k=1}^{Q_b}\sum_{i=1}^{2N} \theta_k^b(\boldsymbol{\mu}) \boldsymbol{u}_{Ni}(\boldsymbol{\mu}) b_k(\boldsymbol{\zeta}_i, q), \end{aligned} \tag{49}$$

and then using the linear superposition principle

$$\begin{aligned} \hat{r}_N^{\boldsymbol{u}}(\boldsymbol{\mu}) &= \sum_{k=1}^{Q_f} \theta_k^f(\boldsymbol{\mu}) \hat{f}_k - \sum_{k=1}^{Q_a}\sum_{i=1}^{2N} \theta_k^a(\boldsymbol{\mu}) u_{Ni}(\boldsymbol{\mu}) \hat{a}_{ki} - \sum_{k=1}^{Q_b}\sum_{i=1}^{N} \theta_k^b(\boldsymbol{\mu}) p_{Ni}(\boldsymbol{\mu}) \hat{b}_{ki}^\intercal, \\ \hat{r}_N^{p}(\boldsymbol{\mu}) &= \sum_{k=1}^{Q_g} \theta_k^g(\boldsymbol{\mu}) \hat{g}_k - \sum_{k=1}^{Q_b}\sum_{i=1}^{2N} \theta_k^b(\boldsymbol{\mu}) \boldsymbol{u}_{Ni}(\boldsymbol{\mu}) \hat{b}_{ki}, \end{aligned} \tag{50}$$

where the Riesz representations $\hat{a}_{ki}$, $\hat{b}_{ki}$, $\hat{b}^{\intercal}_{ki}$, $\hat{f}_k$ and $\hat{g}_k$ satisfy

$$\begin{aligned}
(\hat{a}_{ki}, \boldsymbol{v})_V &= a_k(\boldsymbol{\zeta}_i, \boldsymbol{v}), \quad \forall \boldsymbol{v} \in V, 1 \leq k \leq Q_a,\ 1 \leq i \leq 2N \\
(\hat{b}^{\intercal}_{ki}, \boldsymbol{v})_V &= b_k(\boldsymbol{v}, \eta_i), \quad \forall \boldsymbol{v} \in V, 1 \leq k \leq Q_b,\ 1 \leq i \leq N \\
(\hat{b}_{ki}, q)_Q &= b_k(\boldsymbol{\zeta}_i, q), \quad \forall q \in Q, 1 \leq k \leq Q_b,\ 1 \leq i \leq 2N \\
(\hat{f}_k, \boldsymbol{v})_V &= f_k(\boldsymbol{v}), \qquad \forall \boldsymbol{v} \in V, 1 \leq k \leq Q_f \\
(\hat{g}_k, q)_Q &= g_k(q), \qquad \forall q \in Q, 1 \leq k \leq Q_g.
\end{aligned} \tag{51}$$

And finally we obtain

$$\begin{aligned}
||\hat{r}^{\boldsymbol{u}}_N(\boldsymbol{\mu})||^2_V =& C^{ff}(\boldsymbol{\mu}) + \boldsymbol{u}_N(\boldsymbol{\mu})^{\intercal} C^{aa}_N(\boldsymbol{\mu}) \boldsymbol{u}_n(\boldsymbol{\mu}) + p_N(\boldsymbol{\mu})^{\intercal} C^{b^{\intercal}b^{\intercal}}_N(\boldsymbol{\mu}) p_N(\boldsymbol{\mu}) \\
&+ 2\boldsymbol{u}_N(\boldsymbol{\mu})^{\intercal} C^{ab^{\intercal}}_N(\boldsymbol{\mu}) p_N(\boldsymbol{\mu}) - 2\boldsymbol{u}_N(\boldsymbol{\mu})^{\intercal} C^{af}_N(\boldsymbol{\mu}) - 2p_N(\boldsymbol{\mu})^{\intercal} C^{b^{\intercal}f}_N(\boldsymbol{\mu}) \\
||\hat{r}^{p}_N(\boldsymbol{\mu})||^2_Q =& C^{gg}(\boldsymbol{\mu}) + \boldsymbol{u}_N(\boldsymbol{\mu})^{\intercal} C^{bb}_N(\boldsymbol{\mu}) \boldsymbol{u}_N(\boldsymbol{\mu}) - 2\boldsymbol{u}_N(\boldsymbol{\mu})^{\intercal} C^{bg}_N(\boldsymbol{\mu})
\end{aligned} \tag{52}$$

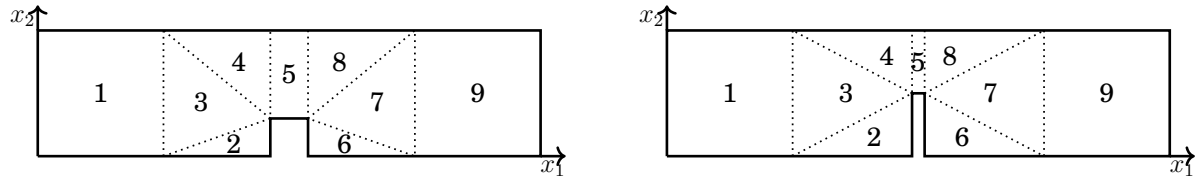
with

$$\begin{aligned}
C^{ff}(\boldsymbol{\mu}) &= \sum_{k=1}^{Q_f} \sum_{k'=1}^{Q_f} \theta^f_k(\boldsymbol{\mu}) \theta^f_{k'}(\boldsymbol{\mu}) \underbrace{(\hat{f}_k, \hat{f}_{k'})_V}_{\text{precomputed}}, \qquad C^{gg}(\boldsymbol{\mu}) = \sum_{k=1}^{Q_g} \sum_{k'=1}^{Q_g} \theta^g_k(\boldsymbol{\mu}) \theta^g_{k'}(\boldsymbol{\mu}) \underbrace{(\hat{g}_k, \hat{g}_{k'})_Q}_{\text{precomputed}}, \\
C^{af}_N(\boldsymbol{\mu}) &= \sum_{k=1}^{Q_f} \sum_{k'=1}^{Q_a} \theta^f_k(\boldsymbol{\mu}) \theta^a_{k'}(\boldsymbol{\mu}) \underbrace{C^{af}_{kk',N}}_{\text{precomputed}}, \qquad (C^{af}_{kk',N})_i = (\hat{f}_k, \hat{a}_{k'i})_V, \quad 1 \leq i \leq 2N, \\
C^{b^{\intercal}f}_N(\boldsymbol{\mu}) &= \sum_{k=1}^{Q_f} \sum_{k'=1}^{Q_b} \theta^f_k(\boldsymbol{\mu}) \theta^b_{k'}(\boldsymbol{\mu}) \underbrace{C^{b^{\intercal}f}_{kk',N}}_{\text{precomputed}}, \qquad (C^{b^{\intercal}f}_{kk',N})_i = (\hat{f}_k, \hat{b}^{\intercal}_{k'i})_V, \quad 1 \leq i \leq N, \\
C^{bg}_N(\boldsymbol{\mu}) &= \sum_{k=1}^{Q_g} \sum_{k'=1}^{Q_b} \theta^g_k(\boldsymbol{\mu}) \theta^b_{k'}(\boldsymbol{\mu}) \underbrace{C^{bg}_{kk',N}}_{\text{precomputed}}, \qquad (C^{bg}_{kk',N})_i = (\hat{g}_k, \hat{b}_{k'i})_Q, \quad 1 \leq i \leq N, \\
C^{aa}_N(\boldsymbol{\mu}) &= \sum_{k=1}^{Q_a} \sum_{k'=1}^{Q_a} \theta^a_k(\boldsymbol{\mu}) \theta^a_{k'}(\boldsymbol{\mu}) \underbrace{C^{aa}_{kk',N}}_{\text{precomputed}}, \qquad (C^{aa}_{kk',N})_{ij} = (\hat{a}_{ki}, \hat{a}_{k'j})_V, \quad 1 \leq i,j \leq 2N, \\
C^{bb}_N(\boldsymbol{\mu}) &= \sum_{k=1}^{Q_b} \sum_{k'=1}^{Q_b} \theta^b_k(\boldsymbol{\mu}) \theta^b_{k'}(\boldsymbol{\mu}) \underbrace{C^{bb}_{kk',N}}_{\text{precomputed}}, \qquad (C^{bb}_{kk',N})_{ij} = (\hat{b}_{ki}, \hat{b}_{k'j})_Q, \quad 1 \leq i,j \leq N, \\
C^{b^{\intercal}b^{\intercal}}_N(\boldsymbol{\mu}) &= \sum_{k=1}^{Q_b} \sum_{k'=1}^{Q_b} \theta^b_k(\boldsymbol{\mu}) \theta^b_{k'}(\boldsymbol{\mu}) \underbrace{C^{b^{\intercal}b^{\intercal}}_{kk',N}}_{\text{precomputed}}, \qquad (C^{b^{\intercal}b^{\intercal}}_{kk',N})_{ij} = (\hat{b}^{\intercal}_{ki}, \hat{b}^{\intercal}_{k'j})_Q, \quad 1 \leq i,j \leq N.
\end{aligned} \tag{53}$$

Here again, the precomputed quantities are evaluated only once during the offline phase and stored to be reused online. This way, during the online phase the computation of the residuals remain independent of the FE dimension $\mathcal{N}$. The same method is used to evaluate the dual norms of the dual residuals.

4 Detailed Affine Decomposition for RB with Stokes Flow

We present here the detailed formulation of the affine decomposition associated to the test case in section 3.



(a) $\mu = (\mu_r, \mu_r)$ (b) $\mu = (0.1, 0.5)$

Figure 17 – Domain Decomposition used for the mapping

The domain decomposition is presented on figure 17. The associated mappings on each sub-domain is

$$\mathcal{G}^1(\tilde{\boldsymbol{x}}, \mu) = \mathcal{G}^7(\tilde{\boldsymbol{x}}, \mu) = \tilde{\boldsymbol{x}}$$

$$\mathcal{G}^2(\tilde{\boldsymbol{x}}, \mu) = \begin{pmatrix} \frac{2-\mu_r}{2-\mu_1} & 0 \\ 0 & \frac{\mu_r}{\mu_2} \end{pmatrix} \tilde{\boldsymbol{x}} + \begin{pmatrix} \frac{\mu_r-\mu_1}{2-\mu_1} \\ 0 \end{pmatrix}, \qquad \mathcal{G}^6(\tilde{\boldsymbol{x}}, \mu) = \begin{pmatrix} \frac{2-\mu_r}{2-\mu_1} & 0 \\ 0 & \frac{\mu_r}{\mu_2} \end{pmatrix} \tilde{\boldsymbol{x}} + \begin{pmatrix} \frac{3(\mu_r-\mu_1)}{2-\mu_1} \\ 0 \end{pmatrix}$$

$$\mathcal{G}^3(\tilde{\boldsymbol{x}}, \mu) = \begin{pmatrix} \frac{2-\mu_r}{2-\mu_1} & 0 \\ \frac{2(\mu_r-\mu_2)}{2-\mu_1} & 1 \end{pmatrix} \tilde{\boldsymbol{x}} + \begin{pmatrix} \frac{\mu_r-\mu_1}{2-\mu_1} \\ \frac{-2(\mu_r-\mu_2)}{2-\mu_1} \end{pmatrix}, \quad \mathcal{G}^7(\tilde{\boldsymbol{x}}, \mu) = \begin{pmatrix} \frac{2-\mu_r}{2-\mu_1} & 0 \\ \frac{-2(\mu_r-\mu_2)}{2-\mu_1} & 1 \end{pmatrix} \tilde{\boldsymbol{x}} + \begin{pmatrix} \frac{3(\mu_r-\mu_1)}{2-\mu_1} \\ \frac{6(\mu_r-\mu_2)}{2-\mu_1} \end{pmatrix}$$

$$\mathcal{G}^4(\tilde{\boldsymbol{x}}, \mu) = \begin{pmatrix} \frac{2-\mu_r}{2-\mu_1} & 0 \\ 0 & \frac{1-\mu_r}{1-\mu_2} \end{pmatrix} \tilde{\boldsymbol{x}} + \begin{pmatrix} \frac{\mu_r-\mu_1}{2-\mu_1} \\ \frac{\mu_r-\mu_2}{1-\mu_2} \end{pmatrix}, \qquad \mathcal{G}^8(\tilde{\boldsymbol{x}}, \mu) = \begin{pmatrix} \frac{2-\mu_r}{2-\mu_1} & 0 \\ 0 & \frac{1-\mu_r}{1-\mu_2} \end{pmatrix} \tilde{\boldsymbol{x}} + \begin{pmatrix} \frac{3(\mu_r-\mu_1)}{2-\mu_1} \\ \frac{\mu_r-\mu_2}{1-\mu_2} \end{pmatrix}$$

$$\mathcal{G}^5(\tilde{\boldsymbol{x}}, \mu) = \begin{pmatrix} \frac{\mu_r}{\mu_1} & 0 \\ 0 & \frac{1-\mu_r}{1-\mu_2} \end{pmatrix} \tilde{\boldsymbol{x}} + \begin{pmatrix} \frac{-2(\mu_r-\mu_1)}{\mu_1} \\ \frac{\mu_r-\mu_2}{1-\mu_2}. \end{pmatrix}$$

It allows to write the affine decomposition of the problem, using the relations,

$$a(\boldsymbol{u}, \boldsymbol{v}; \mu) = \sum_{k=1}^{P} G^k_{mj}(\mu) G^k_{lj} J^k(\mu) \int_{\Omega^k} \frac{\partial u_i}{\partial x_m} \frac{\partial v_i}{\partial x_l} \, d\Omega, \qquad f(\boldsymbol{v}) = 0 \tag{54}$$

$$b(\boldsymbol{u}, q) = \sum_{k=1}^{P} -G^k_{ji} J^k(\mu) \int_{\Omega^k} \frac{\partial u_i}{\partial x_j} q \, d\Omega, \qquad g(q) = 0, \tag{55}$$

we have,

$$a(\boldsymbol{u}, \boldsymbol{v}; \mu) = \sum_{s=1}^{Q_a} \theta^s_a(\mu) a^s(\boldsymbol{u}, \boldsymbol{v}), \tag{56}$$

$$b(\boldsymbol{u}, q; \mu) = \sum_{s=1}^{Q_b} \theta^s_b(\mu) b^s(\boldsymbol{u}, q), \tag{57}$$

with

$$\theta_a^1(\mu) = 1, \qquad \theta_a^2(\mu) = \mu_1, \qquad \theta_a^3(\mu) = \frac{1-\mu_2}{\mu_1} \qquad \theta_a^4(\mu) = \frac{1}{2-\mu 1},$$

$$\theta_a^5(\mu) = \frac{\mu_2}{2-\mu_1}, \qquad \theta_a^6(\mu) = \frac{\mu_2^2}{2-\mu 1}, \qquad \theta_a^7(\mu) = \frac{2-\mu_1}{\mu_2}, \qquad \theta_a^8(\mu) = \frac{\mu_1}{1-\mu_2},$$

$$\theta_a^9(\mu) = \frac{2-\mu_1}{1-\mu_2},$$

and

$$a^1(\boldsymbol{u},\boldsymbol{v}) = \int_{\Omega^{1,9}} \nabla \boldsymbol{u} : \nabla \boldsymbol{v}\ d\Omega\ + \int_{\Omega^{3,7}} \frac{2}{2-\mu_r} \frac{\partial \boldsymbol{u}}{\partial x_2} \cdot \frac{\partial \boldsymbol{v}}{\partial x_2}\ d\Omega$$

$$a^2(\boldsymbol{u},\boldsymbol{v}) = \int_{\Omega^{3,7}} \frac{-1}{2-\mu_r} \frac{\partial \boldsymbol{u}}{\partial x_2} \cdot \frac{\partial \boldsymbol{v}}{\partial x_2}\ d\Omega\ ,$$

$$a^3(\boldsymbol{u},\boldsymbol{v}) = \int_{\Omega^5} \frac{\mu_r}{1-\mu_r} \frac{\partial \boldsymbol{u}}{\partial x_1} \cdot \frac{\partial \boldsymbol{v}}{\partial x_1}\ d\Omega$$

$$\begin{aligned} a^4(\boldsymbol{u},\boldsymbol{v}) = &\int_{\Omega^{3,7}} (2-\mu_r) \frac{\partial \boldsymbol{u}}{\partial x_2} \cdot \frac{\partial \boldsymbol{v}}{\partial x_2} + \frac{4\mu_r^2}{2-\mu_r} \frac{\partial \boldsymbol{u}}{\partial x_2} \cdot \frac{\partial \boldsymbol{v}}{\partial x_2}\ d\Omega\ + \int_{\Omega^{4,8}} \frac{2-\mu_r}{1-\mu_r} \frac{\partial \boldsymbol{u}}{\partial x_1} \cdot \frac{\partial \boldsymbol{v}}{\partial x_1}\ d\Omega \\ &+ \int_{\Omega^3} 2\mu_r \left(\frac{\partial \boldsymbol{u}}{\partial x_1} \cdot \frac{\partial \boldsymbol{v}}{\partial x_2} + \frac{\partial \boldsymbol{u}}{\partial x_2} \cdot \frac{\partial \boldsymbol{v}}{\partial x_1}\right) d\Omega\ + \int_{\Omega^7} -2\mu_r \left(\frac{\partial \boldsymbol{u}}{\partial x_1} \cdot \frac{\partial \boldsymbol{v}}{\partial x_2} + \frac{\partial \boldsymbol{u}}{\partial x_2} \cdot \frac{\partial \boldsymbol{v}}{\partial x_1}\right) d\Omega \\ &, \end{aligned}$$

$$\begin{aligned} a^5(\boldsymbol{u},\boldsymbol{v}) = &\int_{\Omega^{2,6}} \frac{2-\mu_r}{\mu_r} \frac{\partial \boldsymbol{u}}{\partial x_1} \cdot \frac{\partial \boldsymbol{v}}{\partial x_1}\ d\Omega\ + \int_{\Omega^{3,7}} \frac{-8\mu_r}{2-\mu_r} \frac{\partial \boldsymbol{u}}{\partial x_1} \cdot \frac{\partial \boldsymbol{v}}{\partial x_1}\ d\Omega \\ &+ \int_{\Omega^3} -2\left(\frac{\partial \boldsymbol{u}}{\partial x_1} \cdot \frac{\partial \boldsymbol{v}}{\partial x_2} + \frac{\partial \boldsymbol{u}}{\partial x_2} \cdot \frac{\partial \boldsymbol{v}}{\partial x_1}\right) d\Omega\ + \int_{\Omega^7} 2\left(\frac{\partial \boldsymbol{u}}{\partial x_1} \cdot \frac{\partial \boldsymbol{v}}{\partial x_2} + \frac{\partial \boldsymbol{u}}{\partial x_2} \cdot \frac{\partial \boldsymbol{v}}{\partial x_1}\right) d\Omega \\ &+ \int_{\Omega^{4,8}} -\frac{2-\mu_r}{1-\mu_r} \frac{\partial \boldsymbol{u}}{\partial x_1} \cdot \frac{\partial \boldsymbol{v}}{\partial x_1}\ d\Omega\ , \end{aligned}$$

$$a^6(\boldsymbol{u},\boldsymbol{v}) = \int_{\Omega^{3,7}} \frac{4}{2-\mu_r} \frac{\partial \boldsymbol{u}}{\partial x_2} \cdot \frac{\partial \boldsymbol{v}}{\partial x_2}\ d\Omega\ ,$$

$$a^7(\boldsymbol{u},\boldsymbol{v}) = \int_{\Omega^{2,6}} \frac{\mu_r}{2-\mu_r} \frac{\partial \boldsymbol{u}}{\partial x_2} \cdot \frac{\partial \boldsymbol{v}}{\partial x_2}\ d\Omega\ ,$$

$$a^8(\boldsymbol{u},\boldsymbol{v}) = \int_{\Omega^5} \frac{1-\mu_r}{\mu_r} \frac{\partial \boldsymbol{u}}{\partial x_2} \cdot \frac{\partial \boldsymbol{v}}{\partial x_2}\ d\Omega\ ,$$

$$a^9(\boldsymbol{u},\boldsymbol{v}) = \int_{\Omega^{4,8}} \frac{1-\mu_r}{2-\mu_r} \frac{\partial \boldsymbol{u}}{\partial x_2} \cdot \frac{\partial \boldsymbol{v}}{\partial x_2}\ d\Omega\ .$$

$$\theta_b^1(\mu) = 1, \qquad \theta_b^2(\mu) = \mu_1, \qquad \theta_b^3(\mu) = \mu_2,$$

and

$$
\begin{aligned}
b^1(\boldsymbol{u},q) = &\int_{\Omega^{1,9}} -q\nabla\cdot\boldsymbol{u}\; d\Omega \;+\int_{\Omega^5}\frac{-1}{1-\mu_r}\frac{\partial u_1}{\partial x_1}\; d\Omega\\
&+\int_{\Omega^{2,6}}\frac{-2}{2-\mu_r}q\frac{\partial u_2}{\partial x_2}\; d\Omega \;+\int_{\Omega^{4,8}}\frac{-1}{1-\mu_r}q\frac{\partial u_1}{\partial x_1}+\frac{-2}{2-\mu_r}q\frac{\partial u_2}{\partial x_2}\; d\Omega\\
&+\int_{\Omega^{3,7}}\frac{-2}{2-\mu_r}q\frac{\partial u_2}{\partial x_2}-q\frac{\partial u_1}{\partial x_1}\; d\Omega \;+\int_{\Omega^3}\frac{-2\mu_r}{2-\mu_r}q\frac{\partial u_1}{\partial x_2}\; d\Omega \;+\int_{\Omega^7}\frac{2\mu_r}{2-\mu_r}q\frac{\partial u_1}{\partial x_2}\; d\Omega\,,\\
b^2(\boldsymbol{u},q) = &\int_{\Omega^{2,6}}\frac{1}{2-\mu_r}q\frac{\partial u_2}{\partial x_2}\; d\Omega \;+\int_{\Omega^{3,7}}\frac{1}{2-\mu_r}q\frac{\partial u_2}{\partial x_2}\; d\Omega \;+\int_{\Omega^{4,8}}\frac{1}{2-\mu_r}q\frac{\partial u_2}{\partial x_2}\; d\Omega\\
&+\int_{\Omega^5}\frac{-1}{\mu_r}q\frac{\partial u_2}{\partial x_2}\; d\Omega\,,\\
b^3(\boldsymbol{u},q) = &\int_{\Omega^{2,6}}\frac{-1}{\mu_r}\frac{\partial u_1}{\partial x_1}\; d\Omega \;+\int_{\Omega^3}\frac{2}{2-\mu_r}q\frac{\partial u_1}{\partial x_2}\; d\Omega \;+\int_{\Omega^7}\frac{-2}{2-\mu_r}q\frac{\partial u_1}{\partial x_2}\; d\Omega\\
&+\int_{\Omega^{4,8}}\frac{1}{1-\mu_r}q\frac{\partial u_1}{\partial x_1}\; d\Omega \;+\int_{\Omega^5}\frac{1}{1-\mu_r}q\frac{\partial u_1}{\partial x_1}\; d\Omega\,,
\end{aligned}
$$

Bibliography

[1] Regina C Almeida and Renato S Silva. A stable petrov-galerkin method for convection-dominated problems. *Computer methods in applied mechanics and engineering*, 140(3-4):291–304, 1997.

[2] Etienne Balmès. Parametric families of reduced finite element models. theory and applications. *Mechanical Systems and Signal Processing*, 10(4):381–394, 1996.

[3] Maxime Barrault, Yvon Maday, Ngoc Cuong Nguyen, and Anthony T Patera. An empirical interpolation method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667–672, 2004.

[4] George Keith Batchelor. *An introduction to fluid dynamics*. Cambridge university press, 2000.

[5] Franco Brezzi. On the existence, uniqueness and approximation of saddle-point problems arising from lagrangian multipliers. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, 8(R2):129–151, 1974.

[6] Franco Brezzi and Michel Fortin. *Mixed and hybrid finite element methods*, volume 15. Springer Science & Business Media, 2012.

[7] A. Brooks and T. J R Hughes. Streamline-upwind/petrov-galerkin methods for advection dominated flows. *IN: PROC. 3RD INT. CONF. ON FINITE ELEMENTS IN FLOW PROBLEMS, , D.H. NORRIE (ED.)*, 2 , Calgary, Canada, Calgary Univ., no date, 1980.

[8] Alexander N. Brooks and Thomas J.R. Hughes. Streamline upwind/petrov-galerkin formulations for convection dominated flows with particular emphasis on the incompressible navier-stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 32(1):199 – 259, 1982.

[9] A. Buffa, Y. Maday, A. T. Patera, C. Prud'homme, and G. Turinici. A priori convergence of the greedy algorithm for the parametrized reduced basis. *M2AN Math. Model. Numer. Anal.*, 46:595–605, 2012.

[10] P. Burda, J. Novotny, and J. Sistek. On a modification of gls stabilized fem for solving incompressible viscous flows. *International Journal for Numerical Methods in Fluids*, 51:1001–1016, 2005.

[11] E. Burman, M. A. Fernandez, and P. Hansbo. Continuous interior penalty finite element method for oseen's equations. *SIAM Journal of Numerical Analysis*, 44:1248–1274, 2006.

[12] Erik Burman and Alexandre Ern. Nonlinear diffusion and discrete maximum principle for stabilized galerkin approximations of the convection–diffusion-reaction equation. *Computer Methods in Applied Mechanics and Engineering*, 191(35):3833–3855, 2002.

[13] Erik Burman and Alexandre Ern. Stabilized galerkin approximation of convection-diffusion-reaction equations: discrete maximum principle and convergence. *Mathematics of computation*, 74(252):1637–1652, 2005.

[14] N Cardoso and P Bicudo. Time dependent simulation of the driven lid cavity at high reynolds number. *arXiv preprint arXiv:0809.3098*, 2008.

[15] Vincent Chabannes. *Vers la simulation numérique des écoulements sanguins*. Theses, Université de Grenoble, July 2013.

[16] Saifon Chaturantabut and Danny C Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.

[17] Philippe G Ciarlet. *The finite element method for elliptic problems*. SIAM, 2002.

[18] Ramon Codina. A discontinuity-capturing crosswind-dissipation for the finite element solution of the convection-diffusion equation. *Computer Methods in Applied Mechanics and Engineering*, 110(3-4):325–342, 1993.

[19] Todd S Coffey, CT Kelley, and David E Keyes. Pseudotransient continuation and differential-algebraic equations. *SIAM Journal on Scientific Computing*, 25(2):553–569, 2003.

[20] Thierry Coupez and Elie Hachem. Solution of high-reynolds incompressible flow with stabilized finite element and adaptive anisotropic meshing. *Computer Methods in Applied Mechanics and Engineering*, 267:65–85, 2013.

[21] Nguyen Ngoc Cuong, Karen Veroy, and Anthony T Patera. Certified real-time solution of parametrized partial differential equations. In *Handbook of materials modeling*, pages 1529–1564. Springer, 2005.

[22] Cécile Daversin and Christophe Prud'Homme. Simultaneous empirical interpolation and reduced basis method for non-linear problems. *Comptes Rendus Mathématique*, 353(12):1105–1109, 2015.

[23] Cécile Daversin, Stéphane Veys, Christophe Trophime, and Christophe Prud'Homme. A reduced basis framework: Application to large scale non-linear multi-physics problems. In *ESAIM: Proceedings*, volume 43, pages 225–254. EDP Sciences, 2013.

[24] Cecile Daversin Catty. *Reduced basis method applied to large non-linear multi-physics problems : application to high field magnets design*. Theses, Université de Strasbourg, September 2016.

[25] D De Valhl Davis. Natural convection of air in a square cavity: a benchmark solution. *International Journal for Numerical Methods in Fluids*, 3:249–264, 1983.

[26] S. Deparis and G. Rozza. Reduced basis method for multi-parameter-dependent steady navier–stokes equations: applications to natural convection in a cavity. *Journal of Computational Physics*, 228(12):4359–4378, 2009.

[27] Ibrahima Dione. *Analyse théorique et numérique des conditions de glissement pour les fluides et les solides par la méthode de pénalisation*. PhD thesis, Université Laval, 2013.

[28] Eduardo Gomes Dutra Do Carmo and Augusto Cesar Galeão. Feedback petrov-galerkin methods for convection-dominated problems. *Computer methods in applied mechanics and engineering*, 88(1):1–16, 1991.

[29] Jim Douglas and Jun Ping Wang. An absolutely stabilized finite element method for the stokes problem. *Mathematics of computation*, 52(186):495–508, 1989.

[30] Vincent Doyeux. *Modélisation et simulation de systèmes multi-fluides. Applications aux écoulements sanguins*. PhD thesis, 2014. Thèse de doctorat dirigée par Peyla, Philippe et Ismail, Mourad Physique Grenoble 2014.

[31] Ercan Erturk and C Gökçöl. Fourth-order compact formulation of navier–stokes equations and driven cavity flow at high reynolds numbers. *International Journal for Numerical Methods in Fluids*, 50(4):421–436, 2006.

[32] JP Fink and WC Rheinboldt. 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, 1983.

[33] Leopoldo P Franca and F Valentin. On an improved unusual stabilized finite element method for the advective–reactive–diffusive equation. *Computer Methods in Applied Mechanics and Engineering*, 190(13):1785–1800, 2000.

[34] A.-L. Gerner and K. Veroy. Reduced basis a posteriori error bounds for the stokes equations in parametrized domains: A penalty approach. *Math. Models Methods Appl. Sci.*, 21:2103–2134, 2011.

[35] A-L. Gerner and K. Veroy. Certified reduced basis methods for parametrized saddle point problems. *SIAM, Journal of Scientific Computing*, 2012.

[36] Swetlana Giere, Traian Iliescu, Volker John, and David Wells. Supg reduced order models for convection-dominated convection–diffusion–reaction equations. *Computer Methods in Applied Mechanics and Engineering*, 289:454–474, 2015.

[37] Martin A Grepl, Yvon Maday, Ngoc C Nguyen, and Anthony T Patera. Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 41(3):575–605, 2007.

[38] PM Gresho, DF Griffiths, and DJ Silvester. Adaptive time-stepping for incompressible flow part i: scalar advection-diffusion. *MIMS EPrint*, 2008.

[39] H. Grotjans and F. Menter. Wall functions for general application CFD codes. In Papailou, editor, *ECCOMAS 98*, pages 1112–1117, 1998.

[40] Elie Hachem, Benjamin Rivaux, Thibaud Kloczko, Hugues Digonnet, and Thierry Coupez. Stabilized finite element method for incompressible flows with high reynolds number. *Journal of Computational Physics*, 229(23):8643–8665, 2010.

[41] Yannick Hoarau. *Analyse physique par simulation numérique et modélisation des écoulements décollés instationnaires autour de surfaces portantes*. PhD thesis, 2002. Thèse de doctorat dirigée par Braza, Marianna Dynamique des fluides Toulouse, INPT 2002.

[42] Thomas J.R. Hughes, Leopoldo P. Franca, and Marc Balestra. A new finite element formulation for computational fluid dynamics: V. circumventing the babuska-brezzi condition: a stable petrov-galerkin formulation of the stokes problem accommodating equal-order interpolations. *Computer Methods in Applied Mechanics and Engineering*, 59(1):85 – 99, 1986.

[43] Thomas J.R. Hughes, Leopoldo P. Franca, and Gregory M. Hulbert. A new finite element formulation for computational fluid dynamics: Viii. the galerkin/least-squares method for advective-diffusive equations. *Computer Methods in Applied Mechanics and Engineering*, 73(2):173 – 189, 1989.

[44] Thomas J.R. Hughes, Leopoldo P. Franca, and Michel Mallet. A new finite element formulation for computational fluid dynamics: Vi. convergence analysis of the generalized supg formulation for linear time-dependent multidimensional advective-diffusive systems. *Computer Methods in Applied Mechanics and Engineering*, 63(1):97 – 112, 1987.

[45] Thomas J.R. Hughes, Michel Mallet, and Mizukami Akira. A new finite element formulation for computational fluid dynamics: Ii. beyond supg. *Computer Methods in Applied Mechanics and Engineering*, 54(3):341 – 355, 1986.

[46] T.J.R. Hughes and A.N. Brooks. A multidimensional upwind scheme with no crosswind diffusion. *Finite Element Methods for Convection Dominated Flows*, 34:19–35, 1979.

[47] Dinh Bao Phuong Huynh, Gianluigi Rozza, Sugata Sen, and Anthony T Patera. A successive constraint linear optimization method for lower bounds of parametric coercivity and inf–sup stability constants. *Comptes Rendus Mathematique*, 345(8):473–478, 2007.

[48] T. J.R. Hughes I. Harari. What are c and h?: Inequalities for the analysis and design of finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 97:157–192, 1992.

[49] Kazufumi Ito and SS Ravindran. A reduced-order method for simulation and control of fluid flows. *Journal of computational physics*, 143(2):403–425, 1998.

[50] Teymour Javaherchi. Review of spalart-allmaras turbulence model and its modifications. *University of Washington*, 2010.

[51] Volker John and Petr Knobloch. On discontinuity—capturing methods for convection—diffusion equations. *Numerical mathematics and advanced applications*, pages 336–344, 2006.

[52] C Johnson, AH Schatz, and LB Wahlbin. Crosswind smear and pointwise errors in streamline diffusion finite element methods. *mathematics of computation*, 49(179):25–38, 1987.

[53] Claes Johnson, Uno Nävert, and Juhani Pitkäranta. Finite element methods for linear hyperbolic problems. *Computer methods in applied mechanics and engineering*, 45(1-3):285–312, 1984.

[54] DA Kay, PM Gresho, DF Griffiths, and DJ Silvester. Adaptive time-stepping for incompressible flow part ii: Navier-stokes equations. *SIAM Journal on Scientific Computing*, 2010.

[55] Carl Timothy Kelley and David E Keyes. Convergence analysis of pseudo-transient continuation. *SIAM Journal on Numerical Analysis*, 35(2):508–523, 1998.

[56] D. J. Knezevic, N-C. Nguyen, and A. T. Patera. Reduced basis approximation and a posteriori error estimation for the parametrized unsteady boussinesq equations. *Mathematical Models and Methods in Applied Sciences*, 2010.

[57] Andrey Nikolaevich Kolmogorov. The local structure of turbulence in incompressible viscous fluid for very large reynolds numbers. In *Dokl. Akad. Nauk SSSR*, volume 30, pages 299–303, 1941.

[58] L. Kovasznay. Laminar flow behind a two-dimensional grid. *Mathematical Proceedings of the Cambridge Philosophical Society*, 44:58–62, 1948.

[59] Dmitri Kuzmin, Otto Mierka, and Stefan Turek. On the implementation of the κ-ε turbulence model in incompressible flow solvers based on a finite element discretisation. *International Journal of Computing Science and Mathematics*, 1(2-4):193–206, 2007.

[60] P Lax and A Milgram. Ix. parabolic equations. *Contributions to the Theory of Partial Differential Equations.(AM-33)*, 33:167, 2016.

[61] Sérgio L. Frey Leopoldo P. Franca. Stabilized finite element methods:i. application to the advective-diffusive model. *Computer Methods in Applied Mechanics and Engineering*, 95:253–276, 1992.

[62] Sérgio L. Frey Leopoldo P. Franca. Stabilized finite element methods:ii. the incompressible navier-stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 99:209–233, 1992.

[63] Luc Machiels, Yvon Maday, Ivan B. Oliveira, Anthony T. Patera, and Dimitrios V. Rovas. 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, 2000.

[64] Yvon Maday, Andrea Manzoni, and Alfio Quarteroni. An online intrinsic stabilization strategy for the reduced basis approximation of parametrized advection-dominated problems. *Comptes Rendus Mathematique*, 354(12):1188–1194, 2016.

[65] Yvon Maday, Ngoc Cuong Nguyen, Anthony T. Patera, and George S.H. Pau. A general, multipurpose interpolation procedure: the magic points. working paper or preprint, September 2007.

[66] Yvon Maday, A.T. Patera, and Dimitrios Rovas. A blackbox reduced-basis output bound method for noncoercive linear problems. *Studies in Mathematics and Its Applications*, 31, 12 2002.

[67] MT Manzari. An explicit finite element algorithm for convection heat transfer problems. *International Journal of Numerical Methods for Heat & Fluid Flow*, 9(8):860–877, 1999.

[68] Andrea Manzoni. An efficient computational framework for reduced basis approximation and a posteriori error estimation of parametrized navier–stokes flows. *ESAIM: Mathematical Modelling and Numerical Analysis*, 48(4):1199–1226, 2014.

[69] David A Mayne, Asif S Usmani, and Martin Crapper. h-adaptive finite element solution of high rayleigh number thermally driven cavity problem. *International Journal of Numerical Methods for Heat & Fluid Flow*, 10(6):598–615, 2000.

[70] F Menter. Zonal two equation kw turbulence models for aerodynamic flows. In *23rd fluid dynamics, plasmadynamics, and lasers conference*, page 2906, 1993.

[71] Florian R Menter. Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA journal*, 32(8):1598–1605, 1994.

[72] FR Menter, M Kuntz, and R Langtry. Ten years of industrial experience with the sst turbulence model. *Turbulence, heat and mass transfer*, 4(1):625–632, 2003.

[73] S. Mittal. On the performance of high aspect ratio elements for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 188(1–3):269 – 287, 2000.

[74] Akira Mizukami and Thomas JR Hughes. A petrov-galerkin finite element method for convection-dominated flows: an accurate upwinding technique for satisfying the maximum principle. *Computer Methods in Applied Mechanics and Engineering*, 50(2):181–193, 1985.

[75] Federico Negri, Andrea Manzoni, and David Amsallem. Efficient model reduction of parametrized systems by matrix discrete empirical interpolation. *Journal of Computational Physics*, 303:431–454, 2015.

[76] Ahmed K Noor. Recent advances in reduction methods for nonlinear problems. *Computers & Structures*, 13(1-3):31–44, 1981.

[77] Y. Papadopoulos. A driven cavity exploration. https://www.acenumerics.com/the-cavity-sessions.html.

[78] Janet S Peterson. The reduced basis method for incompressible viscous flow calculations. *SIAM Journal on Scientific and Statistical Computing*, 10(4):777–786, 1989.

[79] Olivier Pironneau and Olivier Pironneau. *Finite element methods for fluids*. Wiley Chichester, 1989.

[80] C. Prud'homme, D. V. Rovas, K. Veroy, L. Machiels, Y. Maday, A. T. Patera, and G. Turinici. Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods. *Journal of Fluids Engineering*, 124(1):70–80, 2002.

[81] Christophe Prud'homme and AT Patera. Reduced-basis output bounds for approximately parametrized elliptic coercive partial differential equations. *Computing and Visualization in Science*, 6(2):147–162, 2004.

[82] Alfio Quarteroni and Gianluigi Rozza. Numerical solution of parametrized navier–stokes equations by reduced basis methods. *Numerical Methods for Partial Differential Equations: An International Journal*, 23(4):923–948, 2007.

[83] Alfio Quarteroni, Gianluigi Rozza, and Andrea Manzoni. Certified reduced basis approximation for parametrized partial differential equations and applications. *Journal of Mathematics in Industry*, 1(1):3, 2011.

[84] Werner C Rheinboldt. On the theory and error estimation of the reduced basis method for multi-parameter problems. *Nonlinear Analysis: Theory, Methods & Applications*, 21(11):849–858, 1993.

[85] D. V. Rovas. Reduced-basis output bound methods for parametrized partial differential equations. *Ph.D. thesis, Massachusetts Institute of Technology, Cambridge*, 2003.

[86] Dimitrios Vasileios Rovas. *Reduced-basis output bound methods for parametrized partial differential equations*. PhD thesis, Massachusetts Institute of Technology, 2003.

[87] Gianluigi Rozza, DB Phuong Huynh, and Andrea Manzoni. Reduced basis approximation and a posteriori error estimation for stokes flows in parametrized geometries: roles of the inf-sup stability constants. *Numerische Mathematik*, 125(1):115–152, 2013.

[88] Gianluigi Rozza, Dinh Bao Phuong Huynh, and Anthony T Patera. 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, 2007.

[89] Gianluigi Rozza, Dinh Bao Phuong Huynh, and Anthony T Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *Archives of Computational Methods in Engineering*, 15(3):229, 2008.

[90] Gianluigi Rozza and Karen Veroy. On the stability of the reduced basis method for stokes equations in parametrized domains. *Computer methods in applied mechanics and engineering*, 196(7):1244–1260, 2007.

[91] E. Schenone, S. Veys, and C. Prud'Homme. High Performance Computing for the Reduced Basis Method. Application to Natural Convection. *ESAIM: Proceedings*, pages 255 – 273, December 2013.

[92] PN Shankar and MD Deshpande. Fluid mechanics in the driven cavity. *Annual Review of Fluid Mechanics*, 32(1):93–136, 2000.

[93] Ratnesh K Shukla, Mahidhar Tatineni, and Xiaolin Zhong. Very high-order compact finite difference schemes on non-uniform grids for incompressible navier–stokes equations. *Journal of Computational Physics*, 224(2):1064–1094, 2007.

[94] Pavel E Smirnov and Florian R Menter. Sensitization of the sst turbulence model to rotation and curvature by applying the spalart–shur correction term. *Journal of Turbomachinery*, 131(4):041010, 2009.

[95] Philippe R Spalart and Christopher L Rumsey. Effective inflow conditions for turbulence models in aerodynamic calculations. *AIAA journal*, 45(10):2544–2553, 2007.

[96] PRaA Spalart and S1 Allmaras. A one-equation turbulence model for aerodynamic flows. In *30th aerospace sciences meeting and exhibit*, page 439, 1992.

[97] T Tonn. Reduced-basis method (rbm) for non-affine elliptic parametrized pdes.(phd). *Ulm University*, 2012.

[98] David J Tritton. *Physical fluid dynamics*. Springer Science & Business Media, 2012.

[99] C. T. Shin U. Ghia, K. N. Ghia. High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of Computational Physics*, 48:387–411, 1982.

[100] José M Urquiza, André Garon, and Marie-Isabelle Farinas. Weak imposition of the slip boundary condition on curved boundaries for stokes flow. *Journal of Computational Physics*, 256:748–767, 2014.

[101] Sylvain Vallaghé, Michel Fouquembergh, Annabelle Le Hyaric, and Christophe Prud'Homme. A successive constraint method with minimal offline constraints for lower bounds of parametric coercivity constant. working paper or preprint, June 2011.

[102] K. Veroy and A.T. Patera. Certified real-time solution of the parametrized steady incompressible navier-stokes equations: Rigorous reduced-basis a posteriori error bounds. *Int. J. Numer. Meth. Fluids*, 47:773–788, 2005.

[103] Karen Veroy, Christophe Prud'Homme, Dimitrios V Rovas, and Anthony T Patera. A posteriori error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations. In *Proceedings of the 16th AIAA computational fluid dynamics conference*, volume 3847, pages 23–26. Orlando, FL, 2003.

[104] Karen Veroy, Dimitrios V Rovas, and Anthony T Patera. A posteriori error estimation for reduced-basis approximation of parametrized elliptic coercive partial differential equations:"convex inverse" bound conditioners. *ESAIM: Control, Optimisation and Calculus of Variations*, 8:1007–1028, 2002.

[105] Stéphane Veys. *A computational reduced basis framework : applications to nonlinears multiphysics problems*. Theses, Université Joseph Fourier (Grenoble I), November 2014.

[106] D.C. Wan, B.S.V. Patnaik, and G.W. Wei. A new benchmark quality solution for the buoyancy-driven cavity by discrete singular convolution. *Numerical Heat Transfer: Part B: Fundamentals*, 40(3):199–228, 2001.

[107] Zhu Wang, Imran Akhtar, Jeff Borggaard, and Traian Iliescu. Proper orthogonal decomposition closure models for turbulent flows: a numerical comparison. *Computer Methods in Applied Mechanics and Engineering*, 237:10–26, 2012.

[108] K Yapici and Y Uludag. Finite volume simulation of 2-d steady square lid driven cavity flow at high reynolds numbers. *Brazilian Journal of Chemical Engineering*, 30(4):923–937, 2013.

[109] Olivier Zahm, Marie Billaud-Friess, and Anthony Nouy. Projection-based model order reduction methods for the estimation of vector-valued variables of interest. *SIAM Journal on Scientific Computing*, 39(4):A1647–A1674, 2017.

Summary: We present in this thesis our work on model order reduction for aerothermal simulations. We consider the coupling between the incompressible Navier-Stokes equations and an advection-diffusion equation for the temperature. Since the physical parameters induce high Reynolds and Peclet numbers, we have to introduce stabilization operators in the formulation to deal with the well known numerical stability issue. The chosen stabilization, applied to both fluid and heat equations, is the usual Streamline-Upwind/Petrov-Galerkin (SUPG) which add artificial diffusivity in the direction of the convection field. We also introduce our order reduction strategy for this model, based on the Reduced Basis Method (RBM). To recover an affine decomposition for this complex model, we implemented a discrete variation of the Empirical Interpolation Method (EIM) which is a discrete version of the original EIM. This variant allows building an approximated affine decomposition for complex operators such as in the case of SUPG. We also use this method for the non-linear operators induced by the shock capturing method. The construction of a EIM basis for non-linear operators involves a potentially huge number of non-linear FEM resolutions - depending on the size of the sampling. Even if this basis is built during an offline phase, we usually can not afford such expensive computational cost. We took advantage of the recent development of the Simultaneous EIM Reduced basis algorithm (SER) to tackle this issue.

Keywords: Model order reduction, Computational fluid dynamic, Reduced Basis Method, Empirical Interpolation Method, Finite element method, HPC computing

INSTITUT DE RECHERCHE MATHÉMATIQUE AVANCÉE
UMR 7501
Université de Strasbourg et CNRS
7 Rue René Descartes
67 084 STRASBOURG CEDEX

Tél. 03 68 85 01 29
Fax 03 68 85 03 28

www-irma.u-strasbg.fr
irma@math.unistra.fr

ISSN 0755-3390

IRMA 2018/003
http://tel.archives-ouvertes.fr/tel-0000000