Università degli Studi di Trieste

DIPARTIMENTO DI MATEMATICA E GEOSCIENZE
Corso di Laurea Magistrale in Matematica

Stabilized reduced basis method for transport PDEs with random inputs

Master Thesis in Mathematics

14 July 2016

Advisor:
Prof. Gianluigi Rozza
Co-Advisor:
Dr. Francesco Ballarin

Student:
Davide Torlo

Academic Year 2015/2016

Contents

Sommario

Questa tesi tratta di metodi stabilizzati a basi ridotte per l'approssimazione di equazioni di diffusione–trasporto alle derivate parziali (PDEs), quando il termine di trasporto domina su quello di diffusione, anche inserendole in un contesto di equazioni stocastiche con input aleatori.

Introdurremo rapidamente le PDE ellittiche parametrizzate, in particolare l'equazione di diffusione–trasporto. Poi, presenteremo il metodo a basi ridotte (RB) nei dettagli [19]. Ci concentreremo sulla strategia di campionamento, sullo stimatore dell'errore e sul metodo SCM per l'approssimazione della costante di coercività.

Mostreremo alcuni metodi classici di stabilizzazione per gli elementi finiti (FE) [5, 49] sulle equazioni di diffusione–trasporto con trasporto dominante, quindi studieremo due metodi di stabilizzazione per le basi ridotte [41]. Introdurremo la formulazione generale delle basi ridotte per i problemi parabolici e definiremo una tecnica di stabilizzazione adatta a quest'ultima, basata, anch'essa, sulle stabilizzazione usate negli elementi finiti.

Dopo un'introduzione sulle equazioni stocastiche alle derivate parziali, mostreremo il metodo pesato a basi ridotte [7] e lo combineremo con le tecniche di stabilizzazione studiate. Infine, forniremo un metodo che combina le due tecniche di stabilizzazione, al fine di risparmiare costi computazionali.

Abstract

The aim of this master thesis is to study a stabilized reduced basis method suitable for the approximation of parametrized advection–diffusion partial differential equations (PDEs), in advection dominated cases, even in stochastic equations context, considering random inputs.

We will briefly introduce elliptic coercive parametrized PDEs, in particular advection–diffusion equation. Then we will show the RB method and we will describe it in details [19]. In particular we will focus on sampling strategies, the *a posteriori* error estimator and the SCM for the approximation of the coercivity constant.

We will show some classical stabilization methods for FE approximation of advection dominated problems [5, 49], then we will study two reduced basis stabilization methods [41]. Furthermore, we will introduce the general RB method for parabolic problems and then we design a suitable stabilization technique, based on stabilization for the FE approximation of advection dominated parabolic problems.

After an introduction of stochastic partial differential equations, we will show the weighted RB method [7] and we will combine it with stabilization techniques. Finally, we will provide a method that uses both stabilization techniques to optimize computational costs.

Introduzione

L'obiettivo di questa tesi magistrale è di studiare un metodo stabilizzato a basi ridotte per l'approssimazione di equazioni di diffusione–trasporto alle derivate parziali (PDEs), quando il termine di trasporto domina su quello di diffusione, anche inserendole in un contesto di equazioni stocastiche con input aleatori.

Le equazioni di diffusione trasporto sono molto importanti in molte applicazioni ingegneristiche, infatti, sono usate per modellare, per esempio, fenomeni di diffusione del calore o degli agenti inquinanti nell'atmosfera. Noi siamo interessati a studiare le relative equazioni di diffusione–trasporto nel caso in cui sia alto il numero di Péclet, che, grossolanamente, rappresenta il rapporto tra il termine di trasporto e quello di diffusione.

Inoltre, in queste applicazioni, è spesso richiesta una valutazione rapida di soluzioni approssimate che dipendono da alcuni parametri in input. Ciò avviene, per esempio, quando si tratta di eseguire simulazioni in tempo reale o se abbiamo bisogno di risolvere le nostre equazioni per numerosi parametri. Queste situazioni sono molto comuni in alcuni problemi di ottimizzazione, dove il funzionale da ottimizzare può dipendere dalla soluzione delle equazioni al variare dei parametri.

Il metodo a basi ridotte (RB) può essere una soluzione a questi problemi. Infatti, ci fornisce velocemente delle approssimazioni delle soluzioni delle PDE e garantisce l'accuratezza della soluzione con uno stimatore dell'errore. In letteratura si trovano molti lavori riguardo l'applicazione delle basi ridotte a problemi di diffusione–trasporto con basso numero di Péclet [14, 47, 52] e alcuni su quelli con alti numeri di Péclet [41, 42, 43].

Per trattare i problemi con alti numeri di Péclet, bisogna preventivamente applicare delle tecniche di stabilizzazione [5, 49], in quanto le soluzioni numeriche più usate in questo ambito, ad esempio gli elementi finiti (FE), sulle quali le basi ridotte si basano, presentano forti instabilità.

Confronteremo due tecniche di stabilizzazione delle basi ridotte: una computazionalmente più costosa, che fornisce, però, stabilità a tutti i livelli dell'algoritmo, e un'altra più rapida, ma talvolta instabile. Testeremo questi metodi sia su problemi statici (equazioni ellittiche) che su problemi tempo–dipendenti (equazioni paraboliche).

Infine, studieremo i metodi a basi ridotte con equazioni stocastiche. In particolare, sarà interessante trattare input aleatori governati da note distribuzioni di probabilità. In questo contesto, presenteremo il metodo pesato a basi ridotte (wRB) [7] e lo combineremo con le tecniche di stabilizzazione precedentemente presentate. Inoltre, studieremo una strategia per ridurre i costi computazionali di stabilizzazione, sfruttando i due metodi di stabilizzazione. Tutti gli algoritmi saranno testati con diversi esempi.

Il contributo originale di questo lavoro è l'applicazione di metodi a base ridotta pesati

a equazioni con alto numero di Péclet, sia nelle equazioni ellittiche che paraboliche, con input aleatori e lo sviluppo di diversi metodi di stabilizzazione per ottenere soluzioni stabili ottimizzando i tempi computazionali.

Il lavoro si suddividerà nel seguente modo:

Cap. 1 Introdurremo rapidamente le PDE ellittiche parametrizzate, in particolare l'equazione di diffusione–trasporto. Poi presenteremo il metodo a basi ridotte nei dettagli. Ci concentreremo sulla strategia di campionamento, sullo stimatore dell'errore e sul metodo SCM per l'approssimazione della costante di coercività.

Cap. 2 Mostreremo alcuni metodi classici di stabilizzazione per gli elementi finiti sulle equazioni di diffusione–trasporto, quindi studieremo due metodi di stabilizzazione per le basi ridotte e li testeremo su alcuni esempi.

Cap. 3 Introdurremo la formulazione generale delle basi ridotte per i problemi parabolici e definiremo una tecnica di stabilizzazione adatta a quest'ultima, basata sulle stabilizzazione usate negli elementi finiti. Testeremo alcuni esempi.

Cap. 4 Dopo un'introduzione sulle equazioni alle derivate parziali stocastiche, mostreremo il metodo pesato a basi ridotte e lo combineremo con le tecniche di stabilizzazione studiate. Infine, forniremo un metodo che combina le due tecniche di stabilizzazione, al fine di risparmiare costi computazionali.

Per simulare numericamente tutti gli algoritmi proposti in questo lavoro, abbiamo usato la libreria RBniCS [3], sviluppata da SISSA mathLab, che è un'implementazione in FEniCS di svariate tecniche di modellazione ridotta. Durante il lavoro, la libreria è stata estesa e sviluppata implementando i metodi di stabilizzazione, i metodi a base ridotta per le equazioni paraboliche e il metodo pesato a basi ridotte.

Introduction

The aim of this master thesis is to study a stabilized reduced basis method suitable for the approximation of parametrized advection–diffusion partial differential equations (PDEs), in advection dominated cases, even in stochastic equations context, considering random inputs.

Advection–diffusion equations are very important in many engineering applications, because they are used to model, for example, heat transfer phenomena or the diffusion of pollutants in the atmosphere. We are interested in studying related advection–diffusion PDE when their Péclet numbers, representing, roughly, ratio between the advection and the diffusion field, are high.

Moreover, in such applications, we often need very fast evaluations of the approximated solution, depending on some input parameters. This happens, for example, in the case of *real-time* simulation or if we need to perform repeated approximations of solutions, for different input parameters. We find such *many-query* situations in optimization problems, in which the objective function to be optimized depends on the parameters through the solution of a PDE.

The reduced basis (RB) method can be a solution for these issues. It provides rapidly approximation of solution of PDEs and it is able to guarantee the *reliability* of the solution with a sharp *a posteriori* error bound. In literature we can find many works about application of the RB method to advection-diffusion problems, both with low Péclet number [14, 47, 52] and some with high Péclet number [41, 42, 43].

To deal with high Péclet number problems, we have to resort to some stabilization techniques [5, 49], since high fidelity numerical solutions, such as finite element method, that RB method aims to recover, exhibit strong instability problems.

Concerning stabilization techniques, we will compare two RB stabilization techniques that can be used: one requiring more computational load, but more efficient, and the other cheaper from a computational point of view, but still unstable. We will test these methods on both steady case (elliptic equation) and unsteady case (parabolic equation).

Finally, we will study RB methods, with stochastic equations. In particular, we are interested in dealing with random inputs defined by prescribed random variables. In this context, we will show the wRBM (weighted reduced basis method) [7] and we will apply it to stabilized reduced basis strategies. Moreover, we will study a way to reduce computational costs, combining the two reduced basis stabilization methods studied before. We will use several examples to tests our methods.

The original contribution of this work is to apply the wRBM on unstable high Péclet advection–diffusion equations, both in the steady and unsteady case, with random in-

put parameters and the development of different stabilization methods to obtain stable solutions with optimized computational times.

The work will be structured as following:

Ch. 1 We will briefly introduce elliptic coercive parametrized PDEs, in particular advection–diffusion equation. Then we will introduce the RB method and we will describe it in details. In particular we will focus on sampling strategies, the *a posteriori* error estimator and the SCM for the approximation of the coercivity constant.

Ch. 2 We will show some classical stabilization methods for FE approximation of advection dominated problems, then we will study two reduced basis stabilization methods and we will test them on some examples.

Ch. 3 We will introduce the general RB method for parabolic problems and then we design a suitable stabilization technique, based on stabilization for the FE approximation of advection dominated parabolic problems. We will test this method on few examples.

Ch. 4 After an introduction of stochastic partial differential equations, we will show the weighted RB method and we will combine it with stabilization techniques. Finally, we will provide a method that uses both stabilization techniques to optimize computational costs.

To perform computations that will be proposed in this works, we used RBniCS [3] library, developed at SISSA mathLab, which is an implementation in FEniCS of several reduced order modelling techniques. This library has been extended and developed while carrying out this work, implementing stabilization methods, parabolic RB methods and weighted RB method.

Chapter 1

Reduced basis methods for elliptic coercive PDEs

The reduced basis (RB) method is a reduced order modelling (ROM) technique which provides rapid and reliable solutions for parametrized partial differential equations (PPDEs), in which the parameters can be either physical or geometrical [19].

The need to solve this kind of problems arises in many engineering applications, in which the evaluation of some *output* quantities is required. These *outputs* are often functional of the solution of a PDE, which can in turn depend on some *input* parameters. The aim of the RB method is to provide a very fast computation of this *input-output* evaluation and so it turns out to be very useful especially in *real-time* or *many-query* contexts.

There are several options about the type of reduced basis to use. In this work we will focus on Lagrange basis, but it would be possible to choose Taylor basis [44], Hermite basis [26] and proper orthogonal decomposition (POD) basis [19, 51]. Moreover, we will use only hierarchical RB spaces [19, 51].

The (Lagrange) RB method starts form a high-fidelity approximation space, e.g. a finite element (FE) space, with a large degrees of freedom, then, for a chosen parameter it computes a Galerkin projection of the original solution onto the reduced basis (RB) subspace. This subspace is the one spanned by some pre-computed high-fidelity solutions (snapshots) of the continuous parametrized problem, corresponding to some suitably chosen values of the parameter.

Let us start considering elliptic coercive PPDEs. Denoting with $\boldsymbol{\mu}$ the p-vector parameter, belonging to the parameter space $\mathcal{D} \subset \mathbb{R}^p$, our problem is to find $u(\boldsymbol{\mu})$ in an Hilbert space X such that

$$a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = F(v; \boldsymbol{\mu}) \quad \forall v \in X \tag{1.1}$$

where $a(\cdot, \cdot; \boldsymbol{\mu})$ are coercive continuous bilinear forms and $F(\cdot; \boldsymbol{\mu})$ are continuous linear forms, for all $\boldsymbol{\mu}$ in $\mathcal{D}$. Moreover, we require that the map $\mathcal{D} \to X$ defined by $\boldsymbol{\mu} \to u(\boldsymbol{\mu})$ is smooth, and so the p-dimensional manifold

$$\mathcal{M} = \{u(\boldsymbol{\mu}) \in X | \boldsymbol{\mu} \in \mathcal{D}\} \tag{1.2}$$

turns out to be smooth too.

To proceed in the reduced basis method, we have to define an high-fidelity *truth* approximation space, called $X^{\mathcal{N}}$, which is a linear space of finite dimension $\mathcal{N}$, typically

very large, where we can find our *truth* solution $u^{\mathcal{N}}(\boldsymbol{\mu})$. In this work, we will choose as $X^{\mathcal{N}}$ the classical lagrangian FE space and we will use as *truth* solution the FE one. Other possible choices of *truth* solution can be found in literature, like spectral element [34] and finite volumes [18]. Acting in this way we can consider the *truth* manifold

$$\mathcal{M}^{\mathcal{N}} = \{u^{\mathcal{N}}(\boldsymbol{\mu}) \in X^{\mathcal{N}} | \boldsymbol{\mu} \in \mathcal{D}\} \tag{1.3}$$

where $u^{\mathcal{N}}(\boldsymbol{\mu})$ is the high-fidelity approximation of the solution of (1.1). The goal of the RB method is to provide a low-order approximation of the latter manifold.

The reduced basis method requires the following components [4, 19, 47]:

1. rapidly convergent global approximation by Galerkin projection onto an $N-$dimensional subspace of $X^{\mathcal{N}}$ spanned by solutions of the governing PPDE corresponding to N suitably selected values of the parameter $\boldsymbol{\mu}$. To get a significant reduction of the computational cost, it is crucial that $N \ll \mathcal{N}$.

2. rigorous and sharp a posteriori error estimators for the error between the RB solution and the *truth* one. This estimation is fundamental for both the certification of the method and the sampling procedure used to build the reduced basis. Moreover, we need to require that the computation of these error bound is inexpensive.

3. decoupling of the computation in two stages: an expensive *Offline* stage, to be performed only once, and a very inexpensive *Online* one, in which is actually performed the *input-output* evaluation.

Intuitively, we can represent the approximation of the truth manifold by mean of the Lagrangian RB method as sketched in Figure 1.1, when we are dealing with one dimensional parameter $\boldsymbol{\mu}$. The black line is the truth manifold in the $\mathcal{N}$-dimensional space $X^{\mathcal{N}}$. The black dots represent the snapshot solutions, which act like Lagrangian interpolation nodes. Finally, the red dashed "interpolant" is our RB approximation, that is built by linear combination of snapshot solutions.

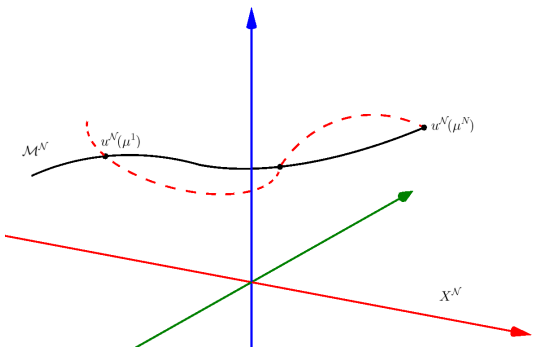


Figure 1.1: Intuitive representation of the truth manifold (1.2) (black line) and its RB approximation (red dashed line)

1.1 Elliptic coercive parametrized PDEs

Let $\boldsymbol{\mu}$ belong to the parameter domain $\mathcal{D}$, a subset of $\mathbb{R}^P$. Let Ω be a regular bounded open subset of $\mathbb{R}^d$ ($d = 1, 2, 3$) and X a suitable Hilbert space. Given a parameter value $\boldsymbol{\mu} \in \mathcal{D}$, let $a(\cdot, \cdot; \boldsymbol{\mu}) : X \times X \to \mathbb{R}$ be a bilinear form and let $F(\cdot; \boldsymbol{\mu}) : V \to \mathbb{R}$ be a linear functional. As we will consider only second order elliptic PDE, the space X will be such that $H_0^1(\Omega) \subset X \subset H^1(\Omega)$. Formally, our problem can be written as follows:

$$\begin{aligned} &\text{find } u(\boldsymbol{\mu}) \in X \text{ s.t.} \\ &a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = F(v; \boldsymbol{\mu}) \quad \forall v \in X. \end{aligned} \tag{1.4}$$

Let us define the norms and the inner products we will use. Let a^{sym} be the symmetric part of a. We define:

$$\begin{aligned} &((v, w))_{\boldsymbol{\mu}} := a^{sym}(v, w; \boldsymbol{\mu}) \quad \forall v, w \in X \\ &|||v|||_{\boldsymbol{\mu}} := a^{sym}(v, v; \boldsymbol{\mu})^{\frac{1}{2}} \quad \forall v \in X. \end{aligned} \tag{1.5}$$

The latter forms are of course $\boldsymbol{\mu}$-dependent, but, for our purpose, we will also need norm and inner product that do not depend on the parameter. Thus we choose a particular value of the parameter $\bar{\boldsymbol{\mu}} \in \mathcal{D}$ and we define:

$$\begin{aligned} &(v, w)_X := ((v, w))_{\bar{\boldsymbol{\mu}}} + \tau (v, w)_{L^2(\Omega)} \quad \forall v, w \in X \\ &||v||_X := ((v, v)_X)^{\frac{1}{2}} \quad \forall v \in X \end{aligned} \tag{1.6}$$

with $\tau > 0$. We will further discuss the choice of τ and $\boldsymbol{\mu}$.

The coercivity and continuity assumption on the form a can now be expressed by, respectively:

$$\exists\ \alpha_0 \text{ s.t. } \alpha_0 \leq \alpha(\boldsymbol{\mu}) = \inf_{v \in X} \frac{a(v, v; \boldsymbol{\mu})}{||v||_X^2} \quad \forall \boldsymbol{\mu} \in \mathcal{D} \tag{1.7}$$

and

$$+\infty > \gamma(\boldsymbol{\mu}) = \sup_{v \in X} \sup_{w \in X} \frac{|a(v, w; \boldsymbol{\mu})|}{||v||_X ||w||_X} \quad \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{1.8}$$

Now we shall make an important assumption: the *affine* dependence of a on the parameter $\boldsymbol{\mu}$. With *affine* we mean that the form can be written in the following way:

$$a(v, w; \boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \Theta_a^q(\boldsymbol{\mu}) a^q(v, w) \quad \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{1.9}$$

Here, $\Theta_a^q(\boldsymbol{\mu}) : \mathcal{D} \to \mathbb{R}$, $q = 1, \ldots, Q_a$, are smooth functions, while $a^q : X \times X \to \mathbb{R}$, $q = 1, \ldots, Q_a$, are $\boldsymbol{\mu}$-independent continuous bilinear forms. This assumption will turn out to be crucial for performing the Offline-Online decoupling of the computation. In a similar way we assume that also the functional F depends "affinely"on the parameter:

$$F(v; \boldsymbol{\mu}) = \sum_{q=1}^{Q_F} \Theta_F^q(\boldsymbol{\mu}) F^q(v) \quad \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{1.10}$$

where, also in this case, $\Theta_F^q(\boldsymbol{\mu}) : \mathcal{D} \to \mathbb{R}$, $q = 1, \dots, Q_F$, are smooth functions, while $F^q : X \to \mathbb{R}$, $q = 1, \dots, Q_F$, are $\boldsymbol{\mu}$-independent continuous linear functionals.

Recalling that $X^{\mathcal{N}}$ is a conforming finite element space [1] with $\mathcal{N}$ degrees of freedom, we can now set the *truth* approximation of the problem (1.4):

$$\begin{aligned} &\text{find } u^{\mathcal{N}}(\boldsymbol{\mu}) \in X^{\mathcal{N}} \text{ s.t.} \\ &a(u^{\mathcal{N}}(\boldsymbol{\mu}), v^{\mathcal{N}}; \boldsymbol{\mu}) = F(v^{\mathcal{N}}; \boldsymbol{\mu}) \quad \forall v^{\mathcal{N}} \in X. \end{aligned} \tag{1.11}$$

As we are considering the conforming FE case, conditions similar to (1.7) and (1.8) are fulfilled by restriction. More precisely, as regards the coercivity of the restriction of a to $X^{\mathcal{N}} \times X^{\mathcal{N}}$, we define:

$$\alpha^{\mathcal{N}}(\boldsymbol{\mu}) := \inf_{v^{\mathcal{N}} \in X^{\mathcal{N}}} \frac{a(v^{\mathcal{N}}, v^{\mathcal{N}}; \boldsymbol{\mu})}{||v^{\mathcal{N}}||_X^2} \quad \forall \boldsymbol{\mu} \in \mathcal{D} \tag{1.12}$$

and, as we are considering a restriction, it easily follows that:

$$\alpha(\boldsymbol{\mu}) \leq \alpha^{\mathcal{N}}(\boldsymbol{\mu}) \quad \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{1.13}$$

Similarly, for the continuity, we can define

$$+\infty > \gamma^{\mathcal{N}}(\boldsymbol{\mu}) = \sup_{v^{\mathcal{N}} \in X^{\mathcal{N}}} \sup_{w^{\mathcal{N}} \in X^{\mathcal{N}}} \frac{|a(v^{\mathcal{N}}, w^{\mathcal{N}}; \boldsymbol{\mu})|}{||v^{\mathcal{N}}||_X ||w^{\mathcal{N}}||_X} \quad \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{1.14}$$

and it holds that:

$$\gamma(\boldsymbol{\mu}) \geq \alpha^{\mathcal{N}}(\boldsymbol{\mu}) \quad \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{1.15}$$

In this work we will consider as *truth* approximation space space $X^{\mathcal{N}}$ a classical finite element space. [45]

1.1.1 Geometrical parametrization

An important feature of the the RB method is that it can be used even when the parameter is "geometrical", i.e. the domain of the equation depends on some parameters [1, 2, 19, 24, 33, 36, 37].

As we will see in the next section, to apply the reduced basis method we need a problem like (1.4), in which the forms involved have to be defined on a parameter independent space. To overcome this difficulty, the idea is to assume that the original parametrized domain is the image of a reference parameter-independent domain through a suitable transformation. By doing so, the parametric dependence actually moves from the domain to the coefficients of the equation.

Let us now call *parameter-dependent original problem* (subscript p) the one defined on the *parameter-dependent original domain* $\Omega_p(\boldsymbol{\mu})$. It reads as follows:

$$\begin{aligned} &\text{find } u_p(\boldsymbol{\mu}) \in X_p(\boldsymbol{\mu}) \text{ s.t.} \\ &a_p(u_p(\boldsymbol{\mu}), v_p; \boldsymbol{\mu}) = F_p(v_p; \boldsymbol{\mu}) \quad \forall v_p \in X_p(\boldsymbol{\mu}) \end{aligned} \tag{1.16}$$

[1] *Conforming finite element space* means that $X^{\mathcal{N}} \subset X$

where $X_p(\boldsymbol{\mu})$ is a functional space on $\Omega_p(\boldsymbol{\mu})$, satisfying the condition $H_0^1(\Omega_p(\boldsymbol{\mu})) \subset X_p(\boldsymbol{\mu}) \subset H^1(\Omega_p(\boldsymbol{\mu}))$. Moreover $a_p(\cdot,\cdot;\boldsymbol{\mu})$ and $F_p(\cdot;\boldsymbol{\mu})$ are a bilinear and a linear form, respectively, on $X_p(\boldsymbol{\mu})$. We assume that the bilinear form a_p satisfies conditions (1.7) and (1.8).

To set the reference domain we choose a particular value of the parameter, $\boldsymbol{\mu}_{ref} \in \mathcal{D}$, and define $\Omega = \Omega_p(\boldsymbol{\mu}_{ref})$ as the reference domain. The latter is related to the original domains through a parametric transformation $T(\cdot;\boldsymbol{\mu}) : \Omega \to \mathbb{R}^P$ such that $T(\Omega;\boldsymbol{\mu}) = \Omega_p(\boldsymbol{\mu})$.

We will now focus only on a particular classes of transformations and problems, as it is done in [19, 47, 51]. First of all, for all $\boldsymbol{\mu} \in \mathcal{D}$ we introduce a domain decomposition of $\Omega_p(\boldsymbol{\mu})$ such that:

$$\overline{\Omega}_p(\boldsymbol{\mu}) = \bigcup_{l=1}^{L_{dom}} \overline{\Omega}_p^l(\boldsymbol{\mu}) \tag{1.17}$$

where the subdomains $\Omega_p^l(\boldsymbol{\mu}),\ l = 1,\dots,L_{dom}$ are mutually non overlapping open subset of $\Omega_p(\boldsymbol{\mu})$, that is:

$$\Omega_p^l(\boldsymbol{\mu}) \cap \Omega_p^{l'}(\boldsymbol{\mu}) = \emptyset \quad 1 \le l < l' \le L_{dom}. \tag{1.18}$$

The need for domain decomposition can arise from modelling reasons, for instance it could happen that the PDE describes a particular application so that different regions of the domain correspond to different materials properties. This can lead to a PDE in which the coefficients show significant discontinuities or the PDE itself can have different form depending on the subdomain. However, a domain decomposition can be set to allow the construction of maps T which guarantees that the forms involved depend "affinely" on the parameter. We will now focus on this second aspect by introducing the piecewise affine transformations.

In order to define the global mapping from the reference domain to the original one, we start by defining the maps between subdomains. For each $\boldsymbol{\mu} \in \mathcal{D}$ we define $T^l(\cdot;\boldsymbol{\mu}) : \Omega^l \to \Omega_p^l(\boldsymbol{\mu})$, $l = 1,\dots,L_{dom}$, such that:

$$\begin{aligned} &T^l(\overline{\Omega^l};\boldsymbol{\mu}) = \overline{\Omega_p^l}(\boldsymbol{\mu}) \quad 1 \le l \le L_{dom}, \\ &T^l(\boldsymbol{x};\boldsymbol{\mu}) = T^{l'}(\boldsymbol{x};\boldsymbol{\mu}) \quad \forall \boldsymbol{x} \in \overline{\Omega^l} \cap \overline{\Omega^{l'}}, \quad 1 \le l < l' \le L_{dom}. \end{aligned} \tag{1.19}$$

We can now define the global mapping $T(\cdot;\boldsymbol{\mu}) : \Omega \to \Omega_p(\boldsymbol{\mu})$ by gluing together the local maps T^l, that is:

$$T(\boldsymbol{x};\boldsymbol{\mu}) := T^l(\boldsymbol{x};\boldsymbol{\mu}) \quad \forall \boldsymbol{x} \in \overline{\Omega^l} \cap \Omega. \tag{1.20}$$

We assume also that:

1. the maps $T^l,\ l = 1,\dots,L_{dom}$, are individually bijective and affine;
2. the map T is continuous.

Under this affinity assumption, each local map T^l can be described by

$$T_i^l(\boldsymbol{x};\boldsymbol{\mu}) = \boldsymbol{C}_i^l(\boldsymbol{\mu}) + \sum_{j=1}^{d} \boldsymbol{G}_{ij}^l(\boldsymbol{\mu}) x_j \quad \boldsymbol{x} \in \Omega^l, \quad 1 \le i \le d \tag{1.21}$$

where $\boldsymbol{C}^l : \mathcal{D} \to \mathbb{R}^d$ and $\boldsymbol{G}^l : \mathcal{D} \to \mathbb{R}^{d\times d}$ are smooth maps which associate to each value of the parameter a vector in $\mathbb{R}^d$ and an invertible $d \times d$ matrix, respectively. Roughly speaking, the matrix $\boldsymbol{G}^l(\boldsymbol{\mu})$ scales and rotates the reference domain, whereas $\boldsymbol{C}^l$ is a translation vector. For each $\boldsymbol{\mu} \in \mathcal{D}$, we denote with $J^l(\boldsymbol{\mu})$ the determinant of the matrix $\boldsymbol{G}^l(\boldsymbol{\mu})$. From now on we consider $d = 2$.

1.1.2 Advection-diffusion-reaction operators

After having introduced the geometry transformations, we have now to discuss the choice of the operators. An important class that can be effectively treated within an "affine" framework is the one of advection-diffusion-reaction operators:

$$Lv = \nabla \cdot (\boldsymbol{\eta}(\boldsymbol{\mu})\nabla v) + \boldsymbol{\beta}(\boldsymbol{\mu}) \cdot \nabla v + \gamma(\boldsymbol{\mu})v \tag{1.22}$$

being $\boldsymbol{\eta}(\boldsymbol{\mu})$ the 2×2 diffusivity tensor, $\boldsymbol{\beta}(\mu)$ the advection (transport) field in $\mathbb{R}^2$ and $\gamma(\boldsymbol{\mu})$ the reaction coefficient. The bilinear form defined on the original domain associated to (1.22) is, for each $v_p, w_p \in X_p(\boldsymbol{\mu})$

$$a_p(v_p, w_p; \boldsymbol{\mu}) = \sum_{l=1}^{L_{dom}} \int_{\Omega_p^l(\boldsymbol{\mu})} \begin{pmatrix} \frac{\partial v_p}{\partial x_{o1}} & \frac{\partial v_p}{\partial x_{o2}} & v_p \end{pmatrix} \boldsymbol{K}_{o,l}(\boldsymbol{\mu}) \begin{pmatrix} \frac{\partial w_p}{\partial x_{o1}} \\ \frac{\partial w_p}{\partial x_{o2}} \\ w_p \end{pmatrix} \tag{1.23}$$

where $\boldsymbol{K}_{o,l} : \mathcal{D} \to \mathbb{R}^{3\times 3}$, $l = 1, \ldots, L_{dom}$, is a smooth mapping such that, for every $\boldsymbol{\mu} \in \mathcal{D}$, the matrix $\boldsymbol{K}_{o,l}(\boldsymbol{\mu})$ has the form:

$$\boldsymbol{K}_{o,l}(\boldsymbol{\mu}) = \left(\begin{array}{c|c} \boldsymbol{\eta}(\boldsymbol{\mu}) & \boldsymbol{\beta}(\boldsymbol{\mu}) \\ \hline \boldsymbol{0} & \gamma(\boldsymbol{\mu}) \end{array}\right). \tag{1.24}$$

Our goal is now to obtain a formulation of the problem in which all the forms are defined on the reference domain.

Denoting with X the space $X_p(\boldsymbol{\mu}_{ref})$, given a value $\boldsymbol{\mu} \in \mathcal{D}$, for each $v_p \in X_p(\boldsymbol{\mu})$ we can define $v \in X$ as $v = v_p \circ T(\cdot, \boldsymbol{\mu})$ (note that we have actually defined a one-to-one correspondence between X and $X_p(\boldsymbol{\mu})$). We can now track back the integrals in (1.23) obtaining:

$$a(v, w; \boldsymbol{\mu}) = \sum_{l=1}^{L_{dom}} \int_{\Omega^l(\boldsymbol{\mu})} \begin{pmatrix} \frac{\partial v}{\partial x_1} & \frac{\partial v}{\partial x_2} & v \end{pmatrix} \boldsymbol{K}_l(\boldsymbol{\mu}) \begin{pmatrix} \frac{\partial w}{\partial x_1} \\ \frac{\partial w}{\partial x_2} \\ w \end{pmatrix} \tag{1.25}$$

with v and w belonging to X. In (1.25) $\boldsymbol{K}_l(\boldsymbol{\mu})$, $l = 1, \ldots, L_{dom}$, represents the transformed operator. The latter can be explicitly written in this way:

$$\boldsymbol{K}_l(\boldsymbol{\mu}) = J^l(\boldsymbol{\mu})\tilde{\boldsymbol{G}}^l(\boldsymbol{\mu})\boldsymbol{K}_{p,l}(\boldsymbol{\mu})(\tilde{\boldsymbol{G}}^l(\boldsymbol{\mu}))^T \tag{1.26}$$

where

$$\tilde{\boldsymbol{G}}^l(\boldsymbol{\mu}) = \left(\begin{array}{c|c} (\boldsymbol{G}^l(\boldsymbol{\mu}))^{-1} & \boldsymbol{0} \\ \hline \boldsymbol{0} & 1 \end{array}\right). \tag{1.27}$$

Similarly we can require that the linear form $f_p(\cdot;\boldsymbol{\mu}) : X_p(\boldsymbol{\mu}) \to \mathbb{R}$ in (1.16) is, for all $v \in X_p(\boldsymbol{\mu})$:

$$F_p(v_p;\boldsymbol{\mu}) = \sum_{l=1}^{L_{dom}} \int_{\Omega_p^l(\boldsymbol{\mu})} F_{p,l}(\boldsymbol{\mu}) v_p. \tag{1.28}$$

Here $F_{p,l}$, $l = 1, \ldots, L_{dom}$, is a function $\mathcal{D} \to \mathbb{R}$.

Acting exactly as before, we can obtain a linear form defined on the reference space X. This form turns out to be, for $v \in X$:

$$F(v;\boldsymbol{\mu}) = \sum_{l=1}^{L_{dom}} \int_{\Omega^l(\boldsymbol{\mu})} F_l(\boldsymbol{\mu}) v \tag{1.29}$$

where, for $l = 1, \ldots, L_{dom}$, the parametric coefficient $F_l(\boldsymbol{\mu})$ is

$$F_l(\boldsymbol{\mu}) = J^l(\boldsymbol{\mu}) F_{p,l}(\boldsymbol{\mu}). \tag{1.30}$$

It is important to note that after this discussion we have actually managed to rewrite the problem (1.16) in the form of (1.4), from which we can obtain the truth approximation formulation like in (1.11).

For further details about the construction of the domain decomposition performed, for example, by the RBniCS [3] software we refer to [19], while for rbMIT© software we refer to [51]. For more complex classes of geometries, which involve non-affine mappings, we need to resort to some interpolation technique (e.g. empirical interpolation method) in order to recover the "affinity" assumption [4, 12, 15, 33].

1.2 The Reduced Basis method

As already mentioned before, the RB method aims to approximate the *truth* solution $u^{\mathcal{N}}(\boldsymbol{\mu})$ of (1.11) by performing a Galerkin projection on a low-dimensional subspace of $X^{\mathcal{N}}$ spanned by solutions of (1.11), that we will call *snapshot* solutions, computed for a well-chosen set of parameter values. In this section we will at first explain the main features of the RB method and then we will illustrate the method used to choose the *snapshots*, highlighting in particular the *a posteriori* error estimates used.

1.2.1 Main features

Let us suppose that we are given a problem in the form (1.4) and its *truth* approximation (1.11). We recall that the dimension of the finite element space $X^{\mathcal{N}}$ is $\mathcal{N}$. We introduce now, given an integer $N_{max} \ll \mathcal{N}$, a sequence of at most N_{max} subspaces of $X^{\mathcal{N}}$. For $N = 1, ..., N_{max}$, let $X_N^{\mathcal{N}}$ be a N-dimensional hierarchical subspace of X such that:

$$X_1^{\mathcal{N}} \subset X_2^{\mathcal{N}} \subset \cdots \subset X_N^{\mathcal{N}} \subset \cdots \subset X_{N_{max}}^{\mathcal{N}}. \tag{1.31}$$

We will call these subspaces "RB spaces". Theoretically, the hierarchical choice of the subspaces would not be necessary. Nevertheless, it turns out to be very useful because it allows a better exploitation of the memory during the computation and, as a consequence, this improves the efficiency of the method.

As mentioned at the beginning of this chapter, we focus on Lagrange RB spaces. In order to define them, we need to introduce a set of N_{max} parameter values:

$$\Xi = \{\boldsymbol{\mu}^1, \dots, \boldsymbol{\mu}^{N_{max}}\} \tag{1.32}$$

and so we can define for $N = 1, \dots, N_{max}$:

$$X_N^{\mathcal{N}} = \text{span}\{u^{\mathcal{N}}(\boldsymbol{\mu}^n) | 1 \leq n \leq N\}. \tag{1.33}$$

The idea behind this definition is to interpolate the truth manifold (1.3) in correspondence of the parameter values belonging to Ξ.

We observe that, by definition, the spaces defined in (1.33) satisfy the hierarchical property (1.31).

Galerkin projection

Given a value $\boldsymbol{\mu} \in \mathcal{D}$ of the parameter and a dimension N, $1 \leq N \leq N_{max}$, of the RB space, we define the RB solution $u_N^{\mathcal{N}}(\boldsymbol{\mu})$ such that:

$$a(u_N^{\mathcal{N}}(\boldsymbol{\mu}), v_N; \boldsymbol{\mu}) = F(v_N; \boldsymbol{\mu}) \quad \forall v_N \in X_N^{\mathcal{N}}. \tag{1.34}$$

Recalling that $N \ll \mathcal{N}$, we emphasize the fact that to find the RB solution we need just to solve a $N \times N$ linear system, instead of the $\mathcal{N} \times \mathcal{N}$ one of the FE method.

If the bilinear form a is symmetric, is straightforward to prove (via Galerkin orthogonality) the following best "fit" approximation result:

$$|||u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N^{\mathcal{N}}(\boldsymbol{\mu})|||_{\boldsymbol{\mu}} \leq \inf_{w^{\mathcal{N}} \in x_N^{\mathcal{N}}} |||u^{\mathcal{N}}(\boldsymbol{\mu}) - w^{\mathcal{N}}|||_{\boldsymbol{\mu}}. \tag{1.35}$$

In order to discuss the Offline-Online computational decoupling, we write explicitly the linear system associated to (1.34). First of all we apply the Gram-Schmidt process [19, 48] with respect to the inner product $(\cdot, \cdot)_X$ defined in (1.6), to the *snapshots* $u(\boldsymbol{\mu}^n)$, $n = 1, \dots, N_{max}$, spanning the RB spaces. We denote with ζ_N^n, $n = 1, \dots, N_{max}$, the mutually orthonormal functions obtained. The RB solution can be now expressed by

$$u_N^{\mathcal{N}}(\boldsymbol{\mu}) = \sum_{m=1} u_{N\ m}^{\mathcal{N}}(\boldsymbol{\mu}) \zeta_m^{\mathcal{N}} \tag{1.36}$$

then, choosing $\zeta_n^{\mathcal{N}}$ as v in (1.35), we obtain

$$\sum_{m=1}^{N} a(\zeta_m^{\mathcal{N}}, \zeta_n^{\mathcal{N}}; \boldsymbol{\mu}) u_{Nm}^{\mathcal{N}}(\boldsymbol{\mu}) = F(\zeta_n^{\mathcal{N}}; \boldsymbol{\mu}) \tag{1.37}$$

and this can be done for $n = 1, \dots, N$, thus obtaining a $N \times N$ linear system [47].

Offline-Online computational decoupling

Given the system (1.37), we can now resort to the affine assumptions (1.9) and (1.10) to construct an efficient Offline-Online procedure. The system (1.37) can be rewritten

$$\sum_{m=1}^{N}\left(\sum_{q=1}^{Q_a}\Theta_a^q(\boldsymbol{\mu})a^q(\zeta_m^{\mathcal{N}},\zeta_n^{\mathcal{N}})\right)u_{Nm}^{\mathcal{N}}(\boldsymbol{\mu})=\sum_{q'=1}^{Q_F}\Theta_F^{q'}(\boldsymbol{\mu})f^{q'}(\zeta_n^{\mathcal{N}}) \tag{1.38}$$

for $n=1,\dots,N$. The system we have obtained can be expressed in matrix form

$$\left(\sum_{q=1}^{Q_a}\Theta_a^q(\boldsymbol{\mu})\mathbf{A}_N^q\right)\mathbf{u}_N(\boldsymbol{\mu})=\sum_{q'=1}^{Q_F}\Theta_F^{q'}(\boldsymbol{\mu})\mathbf{F}_N^{q'} \tag{1.39}$$

where

$$(\mathbf{u}_N(\boldsymbol{\mu}))_m=u_{Nm}^{\mathcal{N}}(\boldsymbol{\mu}),\quad (\mathbf{A}_N^q)_{nm}=a^q(\zeta_m^{\mathcal{N}},\zeta_n^{\mathcal{N}}),\quad (\mathbf{F}_N^q)_n=f^{q'}(\zeta_n^{\mathcal{N}}) \tag{1.40}$$

for $m,n=1,\dots,N$.

In order to compute the matrices $\mathbf{A}_N^q$ and $\mathbf{F}_N^{q'}$ we can recall that $\zeta_n^{\mathcal{N}}$ belongs to $X^{\mathcal{N}}$ for $n=1,\dots,N$ and so it holds that:

$$\zeta_n^{\mathcal{N}}=\sum_{i=1}^{\mathcal{N}}\zeta_{ni}^{\mathcal{N}}\quad 1\le i\le N, \tag{1.41}$$

being $\{\phi\}_{i=1}^{\mathcal{N}}$ the base of the FE space $X^{\mathcal{N}}$. Denoting with $\mathbf{Z}$ the $\mathcal{N}\times N$ matrix whose columns are the coordinates of $\zeta_1^{\mathcal{N}},\dots,\zeta_N^{\mathcal{N}}$ with respect to $\{\phi\}_{i=1}^{\mathcal{N}}$, we have

$$\begin{aligned}\mathbf{A}_N^q&=\mathbf{Z}^T\mathbf{A}_{\mathcal{N}}^q\mathbf{Z}\quad 1\le q\le Q_a\\ \mathbf{F}_N^{q'}&=\mathbf{Z}^T\mathbf{F}_{\mathcal{N}}^{q'}\quad 1\le q'\le Q_F\end{aligned} \tag{1.42}$$

where

$$(\mathbf{A}_{\mathcal{N}}^q)_{ij}=a^q(\phi_j,\phi_i),\quad (\mathbf{F}_{\mathcal{N}}^{q'})_i=F(\phi_i). \tag{1.43}$$

It is crucial to note that in (1.39) the matrices $\mathbf{A}_N^q$ and $\mathbf{F}_N^q$ do not depend on the parameter $\boldsymbol{\mu}$.
So, a good computational strategy is to compute and store them once for all. The computation and storage of the $\boldsymbol{\mu}$-independent structures is called "Offline" stage. More precisely in this stage we compute and store:

- FE stiffness matrices $\mathbf{A}_{\mathcal{N}}^q$, for $q=1,\dots,N$, and FE right-hand side terms $\mathbf{F}_{\mathcal{N}}^q$, for $q=1,\dots,N_{max}$;
- *snapshot* solutions and the corresponding orthonormal basis $\{\zeta_n^{\mathcal{N}}\}_{n=1}^{N_max}$;
- RB stiffness matrices $\mathbf{A}_N^q$, for $q=1,\dots,N$, and RB right-hand side terms $\mathbf{F}_N^{q'}$, for $q'=1,\dots,N$.

We recall that our aim is to obtain, given a new value $\boldsymbol{\mu} \in \mathcal{D}$, a fast and reliable approximation of $u^{\mathcal{N}}(\boldsymbol{\mu})$. To do this, we need to evaluate the coefficients $\Theta_a^q(\boldsymbol{\mu})$ and $\Theta_F^q(\boldsymbol{\mu})$ in order to assemble the $N \times N$ system in (1.39). Once this system has been solved, the RB solution is obtained trough the relation (1.37). The operations done to perform the evaluation $\boldsymbol{\mu} \mapsto u_N^{\mathcal{N}}(\boldsymbol{\mu})$ constitute the "Online" stage.

Let us now analyse the computational cost of the Online stage. First of all we have to consider a cost of $O(Q_a N^2) + O(Q_F N)$ to get the matrix and the right-hand side of the system (1.39), then we need $O(N^3)$ operations to solve it [19, 47, 48]. At last we have to do $O(N)$ operations to perform the product in (1.37) to obtain the solution. As regards the memory used, during the Online stage the storage cost is $O(Q_a N_{max}^2) + O(Q_F N_{max})$, thanks to the hierarchical space assumption (1.31). The latter assumption allow us to store the RB system matrices related to the RB space $X_{N_{max}}^{\mathcal{N}}$ so, if we want to use RB spaces of dimension $N \leq N_{max}$, we need just to take the principal submatrices (or subvectors) of the already stored ones.

The most important thing to note is that the Online stage cost is completely independent from $\mathcal{N}$.

1.2.2 Sampling strategies

We are now going to discuss about the greedy procedure [19, 47, 51] used to explore the parameter space and to construct the RB space. Let us define the train samples set Ξ_{train} as a finite subset of $\mathcal{D}$, with cardinality $|\Xi_{train}| = n_{train}$. We need that n_{train} is large enough to ensure that Ξ_{train} is a good "approximation" of the parameter space $\mathcal{D}$, i.e., even if we would refine the parameter sample, the greedy algorithm should return the same results [19, 51]. This choice can heavily affect the computation of the Offline phase while computation of the Online phase will remain the same. The choice of the train samples will be discussed later, in cases where we do not have sufficient information, we can proceed by using Monte Carlo methods with respect to a uniform or a log-uniform density.

In order to perform a greedy procedure, we need a sharp and computationally inexpensive *a posteriori* error estimator $\boldsymbol{\mu} \to \Delta_N(\boldsymbol{\mu})$, that is

$$|||u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N^{\mathcal{N}}(\boldsymbol{\mu})|||_{\boldsymbol{\mu}} \leq \Delta_N(\boldsymbol{\mu}) \quad \forall \boldsymbol{\mu} \in \mathcal{D}, \quad 1 \leq N \leq N_{max} \tag{1.44}$$

that we will define and discuss in section 1.2.3. The algorithm is recursive and each step is composed by two sub-steps: first, find the value $\bar{\boldsymbol{\mu}} \in \Xi_{train}$ for which the estimator $\Delta_N(\boldsymbol{\mu})$ is maximized, then compute the *truth* solution $u_N^{\mathcal{N}}(\bar{\boldsymbol{\mu}})$ and add it to the Lagrangian basis. By acting in this way, in the $(N+1)$-th iteration, we are adding to the already chosen N-basis the solution that is worst approximated by Galerkin projection onto $X_N^{\mathcal{N}}$. The algorithm stops when the maximum estimated error is less then a prescribed tolerance ϵ_{tol}^*. We introduce also a secondary stopping criterion by setting N_{max} as the maximum number of basis we are willing to accept. If the tolerance has been obtained with a number of basis N less than N_{max} we set $N_{max} = N$. The algorithm can be implemented as follows [19]:

Data: tol, $\boldsymbol{\mu}^1$, N_{max}
Result: A reduced space $X_N^{\mathcal{N}}$
$N = 1$;
$S_1 = \{\boldsymbol{\mu}^1\}$;
compute $u^{\mathcal{N}}(\boldsymbol{\mu}^1)$;
$X_1^{\mathcal{N}} = \text{span}\{u^{\mathcal{N}}(\boldsymbol{\mu}^1)\}$;
compute $\Delta_1(\boldsymbol{\mu}^1)$;
while $\Delta_N(\boldsymbol{\mu}^N) >$ *tol and* $N < N_{max}$ **do**
  $N = N + 1$; compute $\Delta_N(\boldsymbol{\mu}) \quad \forall \boldsymbol{\mu} \in \Xi_{train}$;
  $\boldsymbol{\mu}^N :=\text{argmax}_{\boldsymbol{\mu}\in\Xi_{train}} \Delta_{N-1}(\boldsymbol{\mu})$;
  $S_N = S_{N-1} \cup \{\boldsymbol{\mu}^N\}$;
  compute $u^{\mathcal{N}}(\boldsymbol{\mu}^N)$;
  $X_N^{\mathcal{N}} = X_{N-1}^{\mathcal{N}} \bigoplus \{u^{\mathcal{N}}(\boldsymbol{\mu}^N)\}$;
end

Algorithm 1: Greedy

1.2.3 *A posteriori* error estimates

One of the most important features of the reduced basis method is the *a posteriori* error estimation. As we have seen in section 1.2.2, the estimators Δ_N, $N = 1, \dots, N_{max}$, play a crucial role in the construction of the RB space. For our purposes, a good *a posteriori* error estimator have to fulfil the following characteristics:

- It has to be *rigorous*, in the sense that the inequality

$$|||u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N^{\mathcal{N}}(\boldsymbol{\mu})|||_{\boldsymbol{\mu}} \leq \Delta_N(\boldsymbol{\mu})$$

  must hold for all $\boldsymbol{\mu} \in \mathcal{D}$. This is a fundamental requirement to ensure reliability to the RB method.

- It has to be *sharp*. An overly conservative error bound can cause inefficient approximation spaces, that is with a dimension N unnecessarily high.

- It has to be computationally *efficient*. The computation of the error bound must be very inexpensive both to speed up the Offline stage (i.e. greedy algorithm) and to allow its use in the Online stage. The computational cost should be independent of $\mathcal{N}$.

Before defining the error estimator we will use, we need some preliminaries [19, 47]. First of all we observe that the error $e(\boldsymbol{\mu}) := u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N^{\mathcal{N}}(\boldsymbol{\mu})$ [2], that belongs to $X^{\mathcal{N}}$, satisfies for all $v^{\mathcal{N}} \in X^{\mathcal{N}}$

$$a(e(\boldsymbol{\mu}), v^{\mathcal{N}}; \boldsymbol{\mu}) = a(u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N^{\mathcal{N}}(\boldsymbol{\mu}); \boldsymbol{\mu}) = F(v^{\mathcal{N}}; \boldsymbol{\mu}) - a(u_N^{\mathcal{N}}(\boldsymbol{\mu}), v^{\mathcal{N}}; \boldsymbol{\mu}), \tag{1.45}$$

so we can define the RB residual $r(\cdot; \boldsymbol{\mu}) \in (x^{\mathcal{N}})'$

$$r(v^{\mathcal{N}}; \boldsymbol{\mu}) = F(v^{\mathcal{N}}; \boldsymbol{\mu}) - a(u_N^{\mathcal{N}}(\boldsymbol{\mu}), v^{\mathcal{N}}; \boldsymbol{\mu}) \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}}. \tag{1.46}$$

[2] Here $e(\boldsymbol{\mu})$ depends on $\mathcal{N}$ and N, but, for sake of simplicity, we avoid these heavy notations

As $r(\cdot;\boldsymbol{\mu})$ is a continuous linear functional over $X^{\mathcal{N}}$, we can apply the Riesz representation theorem and get $\hat{r}(\boldsymbol{\mu}) \in X^{\mathcal{N}}$ such that:

$$r(v^{\mathcal{N}};\boldsymbol{\mu}) = (\hat{r}(\boldsymbol{\mu}), v^{\mathcal{N}})_X, \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}}, \tag{1.47}$$

and

$$||\hat{r}(\boldsymbol{\mu})||_X = ||r(\cdot;\boldsymbol{\mu})||_{(X^{\mathcal{N}})'} = \sup_{v^{\mathcal{N}} \in X^{\mathcal{N}}} \frac{r(v^{\mathcal{N}};\boldsymbol{\mu})}{||v^{\mathcal{N}}||_X}. \tag{1.48}$$

Now, we can estimate the error between the FE solution and the RB one in energy norm. Indeed we know that

$$|||e(\boldsymbol{\mu})|||^2_{\boldsymbol{\mu}} = a(e(\boldsymbol{\mu}), e(\boldsymbol{\mu});\boldsymbol{\mu}) \geq \alpha^{\mathcal{N}} ||e(\boldsymbol{\mu})||^2_X \quad \text{i.e. } ||e(\boldsymbol{\mu})||_X \leq \frac{|||e(\boldsymbol{\mu})|||_{\boldsymbol{\mu}}}{\sqrt{\alpha^{\mathcal{N}}}} \tag{1.49}$$

$$|||e(\boldsymbol{\mu})|||^2_{\boldsymbol{\mu}} = a(e(\boldsymbol{\mu}), e(\boldsymbol{\mu});\boldsymbol{\mu}) = r(e(\boldsymbol{\mu})) \leq ||\hat{r}(\boldsymbol{\mu})||_X ||e(\boldsymbol{\mu})||_X \leq \frac{||\hat{r}(\boldsymbol{\mu})||_X}{\sqrt{\alpha^{\mathcal{N}}}} |||e(\boldsymbol{\mu})|||_{\boldsymbol{\mu}} \tag{1.50}$$

from which we can get to the first estimator

$$|||e(\boldsymbol{\mu})|||_{\boldsymbol{\mu}} \leq \frac{||\hat{r}(\boldsymbol{\mu})||_X}{\sqrt{\alpha^{\mathcal{N}}}}. \tag{1.51}$$

Now we introduce a lower bound $\alpha^{\mathcal{N}}_{LB} : \mathcal{D} \to \mathbb{R}$ for the coercivity constant $\alpha^{\mathcal{N}}$ such that:

$$0 < \alpha^{\mathcal{N}}_{LB}(\boldsymbol{\mu}) \leq \alpha^{\mathcal{N}}(\boldsymbol{\mu}) \quad \forall \boldsymbol{\mu} \in \mathcal{D} \tag{1.52}$$

and that the computational cost to evaluate $\boldsymbol{\mu} \to \alpha^{\mathcal{N}}_{LB}(\boldsymbol{\mu})$ is independent of $\mathcal{N}$.
To get this lower bound we can resort to the Successive Constraints Method (SCM) [19, 23, 51] that we will discuss in section 1.2.5.

Now we are ready to give the definition of our a posteriori error estimator. Let us then define the estimator for the energy norm of the error:

$$\Delta_N(\boldsymbol{\mu}) := \frac{||\hat{r}(\boldsymbol{\mu})||_X}{\sqrt{\alpha^{\mathcal{N}}_{LB}}} \tag{1.53}$$

In order to guarantee that this estimator is *sharp* we introduce the effectivity associated:

$$\eta_N(\boldsymbol{\mu}) := \frac{\Delta_N(\boldsymbol{\mu})}{|||e(\boldsymbol{\mu})|||_{\boldsymbol{\mu}}}. \tag{1.54}$$

We can prove the following result [19, 51]:

Proposition 1.2.1. *For any $N = 1, \ldots, N_{max}$ and for any $\boldsymbol{\mu} \in \mathcal{D}$ the effectivity satisfies*

$$1 \leq \eta_N(\boldsymbol{\mu}) \leq \sqrt{\frac{\gamma(\boldsymbol{\mu})}{\alpha^{\mathcal{N}}_{LB}(\boldsymbol{\mu})}} \tag{1.55}$$

Proof. The first inequality is given by (1.51), which says that our estimator is *rigorous*. Recalling (1.46) and the continuity assumption 1.8 we can prove that:

$$\begin{aligned} ||\hat{r}(\boldsymbol{\mu})||^2_{\boldsymbol{\mu}} = a(\hat{r}(\boldsymbol{\mu}), \hat{r}(\boldsymbol{\mu}); \boldsymbol{\mu}) \leq \gamma ||\hat{r}(\boldsymbol{\mu})||^2_X = \gamma(\boldsymbol{\mu}) a(e(\boldsymbol{\mu}), \hat{r}(\boldsymbol{\mu}); \boldsymbol{\mu}) \leq \\ \leq \gamma(\boldsymbol{\mu}) ||e(\boldsymbol{\mu})||_{\boldsymbol{\mu}} ||\hat{r}(\boldsymbol{\mu})||_{\boldsymbol{\mu}} \end{aligned} \tag{1.56}$$

and using this result (1.56), the definition of our error estimator (1.53) and Cauchy-Schwarz inequality

$$\Delta^2_N = \frac{||\hat{r}(\boldsymbol{\mu})||^2_X}{\alpha^{\mathcal{N}}_{LB}(\boldsymbol{\mu})} = \frac{a(e(\boldsymbol{\mu}), \hat{r}(\boldsymbol{\mu}); \boldsymbol{\mu})}{\alpha^{\mathcal{N}}_{LB}(\boldsymbol{\mu})} \leq \frac{||e(\boldsymbol{\mu})||_{\boldsymbol{\mu}} ||\hat{r}(\boldsymbol{\mu})||_{\boldsymbol{\mu}}}{\alpha^{\mathcal{N}}_{LB}(\boldsymbol{\mu})} \leq \frac{||e(\boldsymbol{\mu})||^2_{\boldsymbol{\mu}} \gamma(\boldsymbol{\mu})}{\alpha^{\mathcal{N}}_{LB}(\boldsymbol{\mu})}. \tag{1.57}$$

Finally, we get

$$\eta_N(\boldsymbol{\mu}) = \frac{\Delta_N(\boldsymbol{\mu})}{||e(\boldsymbol{\mu})||_{\boldsymbol{\mu}}} \leq \sqrt{\frac{\gamma(\boldsymbol{\mu})}{\alpha^{\mathcal{N}}_{LB}(\boldsymbol{\mu})}}. \tag{1.58}$$

□

Note that the upper bound on the effectivity in (1.55) is independent of N and hence stable with respect to RB "refinement". This can give us information about the sharpness of the estimation.
Moreover, the method we are using to construct the coercivity lower bound α_{LB} is designed in such a way that:

$$\frac{\alpha^{\mathcal{N}}(\boldsymbol{\mu})}{\alpha^{\mathcal{N}}_{LB}(\boldsymbol{\mu})} \leq C \quad \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{1.59}$$

where C is a positive constant. Recalling (1.13), we can observe that:

$$\eta_N(\boldsymbol{\mu}) \leq \sqrt{\frac{\gamma(\boldsymbol{\mu})}{\alpha^{\mathcal{N}}_{LB}}} \leq \sqrt{\frac{\gamma(\boldsymbol{\mu})}{\alpha^{\mathcal{N}}_{LB}} \frac{\alpha^{\mathcal{N}}(\boldsymbol{\mu})}{\alpha(\boldsymbol{\mu})}} \leq \sqrt{C \frac{\gamma(\boldsymbol{\mu})}{\alpha(\boldsymbol{\mu})}} \tag{1.60}$$

which means that the upper bound for the sensitivity does not depend on the finite element approximation.

1.2.4 Computation of the dual norm of the residual

In order to compute the error bound, we want to show how the dual norm of the residual, that is $||\hat{r}(\boldsymbol{\mu})||_X$, and the lower bound for the coercivity constant $\alpha^{\mathcal{N}}_{LB}(\boldsymbol{\mu})$ can be computed. The goal will be to build a procedure with a computational cost independent of $\mathcal{N}$, by exploiting the affine assumptions (1.9) and (1.10).

Let us start from the residual. First of all, we expand it into its affine terms. So, for all $v^{\mathcal{N}} \in X^{\mathcal{N}}$, we have:

$$\begin{aligned} r(v^{\mathcal{N}}; \boldsymbol{\mu}) &= F(v^{\mathcal{N}}; \boldsymbol{\mu}) - a(u^{\mathcal{N}}_N(\boldsymbol{\mu}), v^{\mathcal{N}}; \boldsymbol{\mu}) \\ &= \sum_{q=1}^{Q_F} \Theta^q_F(\boldsymbol{\mu}) F^q(v^{\mathcal{N}}) - \sum_{q=1}^{Q_a} \Theta^q_a(\boldsymbol{\mu}) a^q \left(\sum_{m=1}^{N} u^{\mathcal{N}}_{N\,m}(\boldsymbol{\mu}) \zeta^{\mathcal{N}}_m, v^{\mathcal{N}} \right) \\ &= \sum_{q=1}^{Q_F} \Theta^q_F(\boldsymbol{\mu}) F^q(v^{\mathcal{N}}) - \sum_{m=1}^{N} u^{\mathcal{N}}_{N\,m}(\boldsymbol{\mu}) \sum_{q=1}^{Q_a} \Theta^q_a(\boldsymbol{\mu}) a^q \left(\zeta^{\mathcal{N}}_m, v^{\mathcal{N}} \right). \end{aligned} \tag{1.61}$$

Recalling (1.46), we have:

$$\hat{r}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_F} \Theta_F^q(\boldsymbol{\mu}) \mathcal{F}^q + \sum_{m=1}^{N} u_{N\,m}^{\mathcal{N}}(\boldsymbol{\mu}) \sum_{q=1}^{Q_a} \Theta_a^q(\boldsymbol{\mu}) \mathcal{A}_m^q \tag{1.62}$$

where $\mathcal{F}^q$ and $\mathcal{A}_m^q$ are solutions of the following systems:

$$\begin{aligned} (\mathcal{F}^q, v^{\mathcal{N}}) &= F^q(v^{\mathcal{N}}) \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}}, \quad 1 \le q \le Q_F, \\ (\mathcal{A}^q, v^{\mathcal{N}}) &= -a^q(\zeta_m^{\mathcal{N}}, v^{\mathcal{N}}) \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}}, \quad 1 \le q \le Q_a,\ 1 \le m \le N. \end{aligned} \tag{1.63}$$

From this expression of the residual, we easily obtain that:

$$\begin{aligned} ||\hat{r}(\boldsymbol{\mu})||_X^2 = &\sum_{q=1}^{Q_F} \sum_{q'=1}^{Q_F} \Theta_F^q(\boldsymbol{\mu}) \Theta_F^{q'}(\boldsymbol{\mu}) (\mathcal{F}^q, \mathcal{F}^{q'})_X + \sum_{q=1}^{Q_a} \sum_{m=1}^{N} \Theta_a^q(\boldsymbol{\mu}) u_{N\,m}^{\mathcal{N}}(\boldsymbol{\mu}) \cdot \\ &\cdot \left[2 \sum_{q'=1}^{Q_F} \Theta_F^{q'}(\boldsymbol{\mu}) (\mathcal{F}^{q'}, \mathcal{A}_m^q)_X + \sum_{q'=1}^{Q_a} \sum_{m=1}^{N} \Theta_a^{q'}(\boldsymbol{\mu}) (\mathcal{A}_m^{q'}, \mathcal{A}_{m'}^{q'})_X \right]. \end{aligned} \tag{1.64}$$

It is now possible to see that we can compute and store the parameter independent quantities once for all. These quantities are:

- the FE "pseudo-solutions" $\mathcal{F}^q$ and $\mathcal{A}_m^{q'}$;
- the scalar products $(\mathcal{F}^q, \mathcal{F}^{q'})_X$, $(\mathcal{F}^q, \mathcal{A}_m^{q'})_X$ and $(\mathcal{A}_m^q, \mathcal{A}_{m'}^{q'})$.

The cost of their computation depends on Q_a, Q_F, $\mathcal{N}$ and N_{max}.

Once we have the latter quantities, we can evaluate $\boldsymbol{\mu} \to ||\hat{r}(\boldsymbol{\mu})||_X^2$ at a very low computational cost. Given a new value $\boldsymbol{\mu} \in \mathcal{D}$ we need just to evaluate the Θ functions and then perform the weighted sum of the already stored quantities. The operation count is $O(N^2 Q_a^2 + N Q_a Q_F + Q_F^2)$ and it does not depend on $\mathcal{N}$.

1.2.5 Lower bound for the coercivity costant

As we mentioned in section 1.2.3 when we defined the *a posteriori* error estimator, we have now to introduce the Successive Constraint Method (SCM) [19, 23] for the evaluation of the coercivity lower bound $\alpha_{LB}^{\mathcal{N}}$.

First of all, we note that the computation of the discrete coercivity constant (1.12) is actually a generalized minimum eigenvalue problem. Denoting with $\{\phi_i\}_{i=1}^{\mathcal{N}}$ a lagrangian FE base for $X^{\mathcal{N}}$, we can define the mass matrix $\overline{\boldsymbol{M}}$ corresponding to the scalar product (1.6) such that:

$$\overline{\boldsymbol{M}}_{ij} = (\phi_i, \phi_j)_X \quad 1 \le i, j \le \mathcal{N}. \tag{1.65}$$

We can also define, for $q = 1, \ldots, N$, the $\boldsymbol{\mu}$-dependent symmetric matrix $\boldsymbol{B}(\boldsymbol{\mu})$ associated to the symmetric part of the bilinear form a:

$$\boldsymbol{B}_{ij} = a^{sym}(\phi_i, \phi_j) \quad 1 \le i, j \le \mathcal{N}. \tag{1.66}$$

Denoting with $\mathtt{v}$ the coordinate vector of $v^{\mathcal{N}} \in X^{\mathcal{N}}$ with respect to the given base, we have that:

$$\frac{a(v^{\mathcal{N}}, v^{\mathcal{N}}; \boldsymbol{\mu})}{||v^{\mathcal{N}}||_X^2} = \frac{a^{sym}(v^{\mathcal{N}}, v^{\mathcal{N}}; \boldsymbol{\mu})}{||v^{\mathcal{N}}||_X^2} = \frac{\mathtt{v}^T \boldsymbol{B}(\boldsymbol{\mu})\mathtt{v}}{\mathtt{v}^T \overline{\boldsymbol{M}}\mathtt{v}} \tag{1.67}$$

and the problem

$$\begin{aligned} &\text{given } \boldsymbol{\mu} \in X^{\mathcal{N}}, \text{ compute} \\ &\alpha^{\mathcal{N}} = \inf_{v^{\mathcal{N}} \in X^{\mathcal{N}}} \frac{a(v^{\mathcal{N}}, v^{\mathcal{N}}; \boldsymbol{\mu})}{||v^{\mathcal{N}}||_X^2} \end{aligned} \tag{1.68}$$

is equivalent to find the minimum generalized eigenvalue of the following eigenvalue problem:

$$\boldsymbol{B}(\boldsymbol{\mu})\mathtt{v} = \lambda \overline{\boldsymbol{M}}\mathtt{v} \tag{1.69}$$

that can be treated, for example, by Lanczos method [10]. To improve the efficiency, we define the norm (1.6) by setting

$$\tau = \inf_{v^{\mathcal{N}} \in X^{\mathcal{N}}} \frac{a(v^{\mathcal{N}}, v^{\mathcal{N}}; \bar{\boldsymbol{\mu}})}{||v^{\mathcal{N}}||_X^2} \tag{1.70}$$

as done in [19, 51].

Before introducing the SCM, let us call $a^{sym,q}$ the symmetric part of the form a^q defined in (1.9).

We define now an objective functional $\mathcal{J}^{obj} : \mathcal{D} \times \mathbb{R}^{Q_a} \to \mathbb{R}$ as

$$\mathcal{J}^{obj}(\boldsymbol{\mu}; y) = \sum_{i=1}^{Q_a} \Theta_a^q(\boldsymbol{\mu}) y^q, \tag{1.71}$$

where $y = (y_1, \ldots, y_{Q_a})$. An equivalent formulation of the problem (1.68) can be:

$$\begin{aligned} &\text{given } \boldsymbol{\mu} \in X^{\mathcal{N}}, \text{ compute} \\ &\alpha^{\mathcal{N}}(\boldsymbol{\mu}) = \inf_{y \in \mathcal{Y}} \mathcal{J}^{obj}(\boldsymbol{\mu}; y) \end{aligned} \tag{1.72}$$

where

$$\mathcal{Y} = \left\{ y \in \mathbb{R}^{Q_a} | \exists w_y^{\mathcal{N}} \in X^{\mathcal{N}} \text{ s.t. } y_q = \frac{a^{sym,q}(w_y^{\mathcal{N}}, w_y^{\mathcal{N}})}{||w_y^{\mathcal{N}}||_X^2}, 1 \le q \le Q_a \right\}. \tag{1.73}$$

We want to provide an upper and a lower bound for α, so we will build a method that give us two sets such that $\mathcal{Y}_{UB} \subset \mathcal{Y} \subset \mathcal{Y}_{LB}$ and will define:

$$\alpha_{LB}(\boldsymbol{\mu}) = \min_{y \in \mathcal{Y}_{LB}} J^{obj}(\boldsymbol{\mu}, y) \qquad \text{and} \qquad \alpha_{UB}(\boldsymbol{\mu}) = \min_{y \in \mathcal{Y}_{UB}} J^{obj}(\boldsymbol{\mu}, y). \tag{1.74}$$

We are interested in both lower and upper bounds to ensure the accuracy for the error quantity

$$\eta(\boldsymbol{\mu}) := \frac{\alpha_{UB}(\boldsymbol{\mu}) - \alpha_{LB}(\boldsymbol{\mu})}{\alpha_{UB}(\boldsymbol{\mu})}. \tag{1.75}$$

The offline part of the SCM is based on a greedy approach where the n-th iteration of the procedure is initiated by assuming that

1. We know coercivity constants $\alpha(\boldsymbol{\mu}_j)$, $1 \leq j \leq n$, for some parameter values $C_n = \{\boldsymbol{\mu}_1^{SCM}, \ldots, \boldsymbol{\mu}_n^{SCM}\} \in \mathcal{D}$.
2. Let Ξ^{SCM} be a representative finite set of parameters of $\mathcal{D}$. For each $\boldsymbol{\mu} \in \Xi^{SCM}$, we have some lower bound $\alpha_{LB}^{n-1}(\boldsymbol{\mu}) \geq 0$ of $\alpha(\boldsymbol{\mu})$ from the previous iteration.

The eigensolutions $(\alpha(\boldsymbol{\mu}_j), v_j^{\mathcal{N}}) \in \mathbb{R}^+ \times X^{\mathcal{N}}$ are solutions to the generalized eigenvalue problem (1.68). From the eigenfunctions $\{v_j^{\mathcal{N}}\}_{j=1}^n$ we get the corresponding eigenvectors

$$(y_j)^q = \frac{a^{sym,q}(v_j^{\mathcal{N}}, v_j^{\mathcal{N}})}{||v_j^{\mathcal{N}}||_X^2}, \quad 1 \leq q \leq Q_a,\, 1 \leq j \leq n \tag{1.76}$$

and then we set

$$\mathcal{Y}_{UB}^n = \{y_j \in \mathbb{R}^{Q_a} | 1 \leq l \leq n\}, \tag{1.77}$$

which is clearly a subset of $\mathcal{Y}$. We can use $\mathcal{Y}_{UB}^n$ to compute $\alpha_{UB}^n(\boldsymbol{\mu}) = \min_{y \in \mathcal{Y}_{UB}^n} J^{obj}(\boldsymbol{\mu}, y)$. This is clearly independent of $\mathcal{N}$ once the vector y_j have been built.

To get the lower bound, first, we define a "bounding box" containing $\mathcal{Y}$

$$\mathcal{B} = \prod_{q=1}^{Q_a} \left[\inf_{v^{\mathcal{N}} \in X^{\mathcal{N}}} \frac{a^{sym,q}(v^{\mathcal{N}}, v^{\mathcal{N}})}{||v^{\mathcal{N}}||_X^2}, \sup_{v^{\mathcal{N}} \in X^{\mathcal{N}}} \frac{a^{sym,q}(v^{\mathcal{N}}, v^{\mathcal{N}})}{||v^{\mathcal{N}}||_X^2} \right] \subset \mathbb{R}^{Q_a}; \tag{1.78}$$

from the continuity hypothesis, $\mathcal{B}$ is bounded. We denote with $C(M, \boldsymbol{\mu}, E)$ the set of $M \geq 1$ points in E (that can be both Ξ^{SCM} or C_n) closest to $\boldsymbol{\mu} \in \mathcal{D}$, with respect to the usual Euclidean norm in $\mathbb{R}^P$. If $M > \texttt{card}(E)$, we set $C(M, \boldsymbol{\mu}, E) = E$.

Now, given $\boldsymbol{\mu} \in \mathcal{D}$ we define the "lower bound" set $\mathcal{Y}_{LB}^n(\boldsymbol{\mu}) \subset \mathbb{R}^{Q_a}$ for some M_e, M_p as

$$\begin{aligned} \mathcal{Y}_{LB}^n(\boldsymbol{\mu}) = \Big\{ & y \in \mathcal{B} | J^{obj}(\boldsymbol{\mu}', y) \geq \alpha^{\mathcal{N}}(\boldsymbol{\mu}'), \quad \forall \boldsymbol{\mu}' \in C(M_e, \boldsymbol{\mu}, C_n), \\ & J^{obj}(\boldsymbol{\mu}', y) \geq \alpha_{LB}^{n-1}(\boldsymbol{\mu}'), \quad \forall \boldsymbol{\mu}' \in C(M_p, \boldsymbol{\mu}, \Xi^{SCM} \setminus C_n) \Big\}. \end{aligned} \tag{1.79}$$

It can be shown that $\mathcal{Y}_{UB}^n \subset \mathcal{Y} \subset \mathcal{Y}_{LB}^n(\boldsymbol{\mu})$ [23]. Consequently, these sets are nested as

$$\mathcal{Y}_{UB}^1 \subset \mathcal{Y}_{UB}^2 \subset \cdots \subset \mathcal{Y}_{UB}^n \subset \cdots \subset \mathcal{Y} \subset \cdots \subset \mathcal{Y}_{LB}^n(\boldsymbol{\mu}) \subset \cdots \subset \mathcal{Y}_{LB}^2(\boldsymbol{\mu}) \subset \mathcal{Y}_{LB}^1(\boldsymbol{\mu}). \tag{1.80}$$

Note that finding $\alpha_{LB}^n(\boldsymbol{\mu}) = \min_{y \in \mathcal{Y}_{LB}^n} J^{obj}(\boldsymbol{\mu}, y)$ corresponds to a linear programming problem of Q_a design variables and $2Q_a + M_e + M_p$ conditions. The complexity of the linear programming problem is thus independent of $\mathcal{N}$.

Having defined the two sets $\mathcal{Y}_{UB}^n, \mathcal{Y}_{LB}^n(\boldsymbol{\mu})$, we can define a greedy selection to enrich

the space C_n and build C_{n+1} at all stages n, minimizing the error of the accuracy (1.75).

Data: tol, $\boldsymbol{\mu}^1$, N_{max}
Result: The sample points C_n, corresponding coercivity constants $\alpha^{\mathcal{N}}(\boldsymbol{\mu}_j)$, vectors y_j, lower bounds $\alpha_{LB}(\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in \Xi^{SCM}$
$C_1 = \{\boldsymbol{\mu}^1\}$;
for $\boldsymbol{\mu} \in \Xi^{SCM}$ **do**
  compute $\alpha^1_{UB}(\boldsymbol{\mu})$ and $\alpha^1_{LB}(\boldsymbol{\mu})$;
  compute $\eta(\boldsymbol{\mu}, C_1) = 1 - \frac{\alpha^1_{LB}(\boldsymbol{\mu})}{\alpha^n_{UB}(\boldsymbol{\mu})}$;
end
while $\max_{\boldsymbol{\mu}\in\Xi^{SCM}} \eta(\boldsymbol{\mu}) >$tol **do**
  select $\boldsymbol{\mu}^n = \text{argmax}_{\boldsymbol{\mu}\in\Xi^{SCM}} \eta(\boldsymbol{\mu}, C_{n-1})$;
  for $\boldsymbol{\mu} \in \Xi^{SCM}$ **do**
    compute $\alpha^n_{UB}(\boldsymbol{\mu})$ and $\alpha^n_{LB}(\boldsymbol{\mu})$;
    compute $\eta(\boldsymbol{\mu}, C_n) = 1 - \frac{\alpha^n_{LB}(\boldsymbol{\mu})}{\alpha^n_{UB}(\boldsymbol{\mu})}$;
  end
  solve the generalized eigenvalue problem (1.68) associated with $\boldsymbol{\mu}^{n+1}$;
  store $\alpha^{\mathcal{N}}(\boldsymbol{\mu}^{n+1})$ and y_{n+1};
end

Algorithm 2: Offline procedure of the SCM

Let us observe that our construction of the lower bound guarantees the condition (1.59). We can indeed see that:

$$\frac{\alpha^{\mathcal{N}}(\boldsymbol{\mu})}{\alpha^N_{LB}(\boldsymbol{\mu})} \leq \frac{\alpha^{\mathcal{N}}(\boldsymbol{\mu})}{\alpha^N_{UB}(\boldsymbol{\mu})} \frac{1}{1-\texttt{tol}} \leq \frac{1}{1-\texttt{tol}} \quad \forall \boldsymbol{\mu} \in \Xi^{SCM}. \tag{1.81}$$

Computational Cost

During the Offline stage the following computations are performed:

1. $2Q_a$ eigenproblems over $X^{\mathcal{N}}$ to build the "continuity constraint" box $\mathcal{B}$". Cost: $O(2Q_a\mathcal{N})$.
2. N_{max} eigenproblems over $X^{\mathcal{N}}$ to form the set $\{\alpha^{\mathcal{N}}(\boldsymbol{\mu}|\boldsymbol{\mu} \in C_{N_{max}}\}$. Cost: $O(N_{max}\mathcal{N})$.
3. $N_{max}Q_a$ inner products over $X^{\mathcal{N}}$ to compute $\mathcal{Y}^{N_{max}}_{UB}$. Cost: $O(N_{max}Q_a\mathcal{N})$.
4. $n^{SCM}_{train}N_{max}$ lower bound linear programming problems of size $2Q_a+M_e+M_p$ and the associated enumerations to compute the upper bounds. Cost: $O(n^{SCM}_{train}N_{max}Q_aM)$.

Note that the global Offline computational cost does not depend on the product $n^{SCM}_{train}\mathcal{N}$, so we can choose large train sets and *truth* approximation space with high dimension $\mathcal{N}$ without worsening too much the computational efficiency.

In the Online stage, for each evaluation $\boldsymbol{\mu} \mapsto \alpha_{LB}(\boldsymbol{\mu})$ we have to:

1. sort of N_{max} points of C_n to build $C(M, \boldsymbol{\mu}, E)$;
2. evaluate the Θ functions;

3. solve the resulting linear programming to obtain the lower bound.

This procedure is independent of $\mathcal{N}$. Finally we want to say that the SCM can be efficiently applied also to compute the inf-sup lower bound for non-coercive problems [47, 53]. Moreover, several improvements for the SCM have been recently proposed [8, 22, 32].

Chapter 2

Stabilized reduced basis method for advection dominated PDEs

In this chapter we will study a class of advection diffusion equations whose FE approximations are numerically unstable. To deal with such problems, we will use some stabilization techniques [49]. In particular we will focus on the RB method in the approximation of advection diffusion equations when the advection effects are much stronger than the diffusive ones (advection dominated problems).

As the advection-diffusion equations are often used to model heat transfer phenomena, we can find in literature many results about the RB approximation of heat transfer problem such as Graetz problem or "thermal fin" problem [14, 47, 50, 52]. However, the latter works consider only the case in which the Péclet number – that is, roughly speaking, the ratio between the advection coefficient and the diffusion one – is low, in a sense that we will specify in section 2.1. The stabilization methods have been used in the RB framework in some works about the approximation of steady advection reaction equations and steady control problems [9, 46]. In these works we can also find some applications to environmental science problems concerning, in particular, air pollution.

In this chapter we will follow main ideas of [40, 41, 42, 43], about further results on stabilized RB method for advection diffusion problems. In particular, after a brief introduction on some stabilization method (section 2.1), we will discuss results obtained stabilizing only the Offline stage or both Online and Offline stages with several values of the Péclet number. We will see that, with high Péclet number, the latter strategy is the only one which provides stable results and a good approximation with respect to the FE stabilized solution. This will be an introduction to chapter 4, where we will use this method in case of random input parameter distributed as some random variables.

2.1 Stabilization methods

In this chapter we will illustrate some stabilization methods for advection-diffusion equations. We will focus in particular on a simplified advection-diffusion equation of this form

$$-\varepsilon\Delta u + \boldsymbol{\beta}\cdot\nabla u = f \quad \text{in } \Omega \subset \mathbb{R}^2. \tag{2.1}$$

This is a particular case of the general problem $Lu = f$, with L the diffusion-advection-reaction operator defined in (1.22), when we do not have a reaction term and the diffusivity tensor is a multiple of the identity. If the advective $\boldsymbol{\beta} \cdot \nabla u$ term dominates the diffusive term $-\varepsilon \Delta u$, that is when $|\boldsymbol{\beta}| \gg \varepsilon$, even the classical discretizations, like FE approach, can be very unsatisfactory, because the approximated solution can show strong instability phenomena, especially along the direction of the advection field. In the next sections we will illustrate and analyse some stabilization methods, able to fix this lack of stability.

2.1.1 Advection dominated problems

Let us make precise assumption on our setting. As mentioned before, we will focus on equations like (2.1) where:

- the diffusion coefficient $\varepsilon : \Omega \to \mathbb{R}$ belongs to $L^\infty(\Omega)$ and exist $\varepsilon_0 > 0$ such that

$$\varepsilon(x) \geq \varepsilon_0 \quad \forall x \in \Omega; \tag{2.2}$$

- the advection field $\boldsymbol{\beta} : \Omega \to \mathbb{R}^2$ belongs to $(L^\infty(\Omega))^2$:
- $f : \Omega \to \mathbb{R}^2$ is an $L^2(\Omega)$ function.

In order to guarantee the well-posedness of our problem we suppose also that the following inequality holds:

$$0 \geq \mathrm{div}\boldsymbol{\beta}(x) \geq -d_1 \quad \forall x \in \Omega \tag{2.3}$$

where d_1 is a positive real constant.

We suppose now that we are given a regular triangulation $\mathcal{T}_h$ (for the definition see [45]), where h is the maximum element diameter (mesh size). For any element $K \in \mathcal{T}_h$, we can then define the local Péclet number [45, 49]:

$$\mathbb{P}\mathrm{e}_K(x) := \frac{|\boldsymbol{\beta}(x)| h_k}{2\varepsilon(x)} \quad \forall x \in K, \tag{2.4}$$

where h_K is the diameter of K.

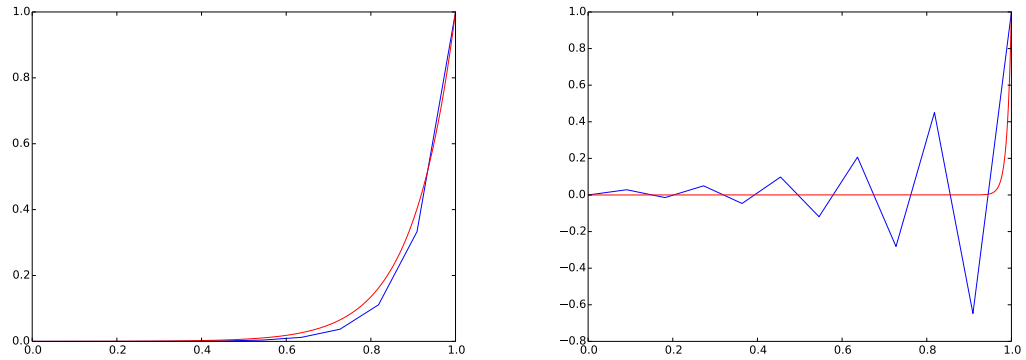
We say that we are dealing with an *advection dominated* problem if

$$\mathbb{P}\mathrm{e}_K(x) > 1 \quad \forall x \in K, \quad \forall K \in \mathcal{T}_h. \tag{2.5}$$

To give an example of what can happen in *advection dominated* situation, we consider the following 1-dimensional problem:

$$\begin{cases} -\varepsilon u'' + u' = 0 \quad \text{in } \Omega := (0,1) \\ u(0) = 0 \\ u(1) = 1 \end{cases} \tag{2.6}$$

If the coefficient ε is "large", e.g. $\varepsilon = \frac{1}{10}$, the FE method yields a good approximation of that solution, as it can be seen in figure 2.1(a). On contrary if we choose a "smaller" value for the same coefficient, e.g. $\varepsilon = \frac{1}{100}$, the FE solution is highly affected by spurious

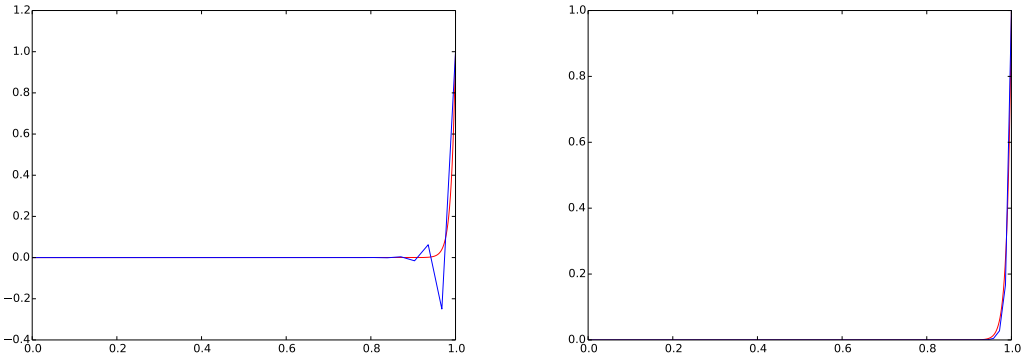


(a) $\varepsilon = \frac{1}{10}$, $h = \frac{1}{10}$ (b) $\varepsilon = \frac{1}{100}$, $h = \frac{1}{10}$

Figure 2.1: One dimensional case, different diffusivities

oscillation as we can see in figure 2.1(b). The mesh used in these computations is an equispaced of size $h = \frac{1}{10}$. (In red we have the exact solution, in blue the FE approximated one).

Theoretically, to avoid instability problems, it would be sufficient to reduce the mesh size h in order to lower the local Péclet number. In this one-dimensional case, it is not a big effort, and we can see in figure 2.2(a) and 2.2(b), that refining the mesh, we will have more accurate approximated solutions.



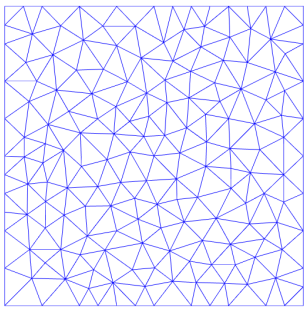
(a) $\varepsilon = \frac{1}{100}$, $h = \frac{1}{30}$ (b) $\varepsilon = \frac{1}{100}$, $h = \frac{1}{70}$

Figure 2.2: One dimensional case, different mesh sizes

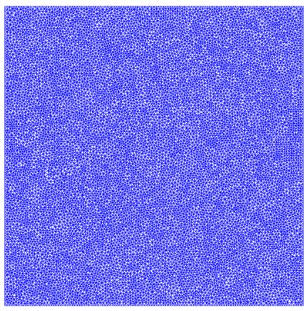
Conversely, if we are dealing with higher dimensional meshes, a "small" mesh size yields a significant increase of the computational cost. Just to give an example, let us take

$$\begin{cases} -\varepsilon \Delta u + (1,1) \cdot \nabla u = 1 & \text{in } \Omega := (0,1) \times (0,1) \\ u = 0 & \text{on } \partial\Omega \end{cases} \tag{2.7}$$

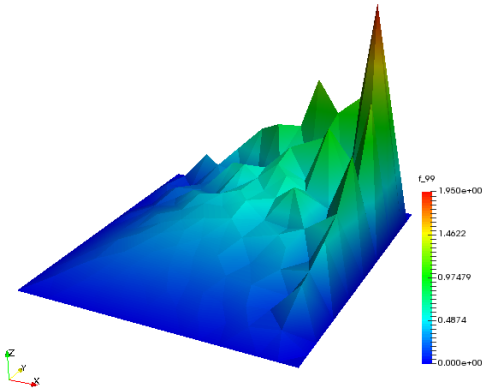
Even in this simple example, if we want to have $\mathbb{P}\mathtt{e}_K(x) < 1$ with $\varepsilon = \frac{1}{100}$, we need a mesh such that $h < 0.015$. Our computational cost will grow significantly, for example, in figure 2.3(c) and 2.3(d) we can see solutions for problem (2.7) with $\varepsilon = \frac{1}{100}$ and different mesh sizes. The first mesh 2.3(a), with a $\mathbb{P}^1$ polynomial space, generates a 174 degrees of freedom space $X^{\mathcal{N}}$, while the second 2.3(b) forms a 12859 one and the maximum mesh size is respectively $h_1 = 0.14$ and $h_2 = 0.013$.
The former has large instabilities over the whole square, while the latter is a good approximation of the exact solution.



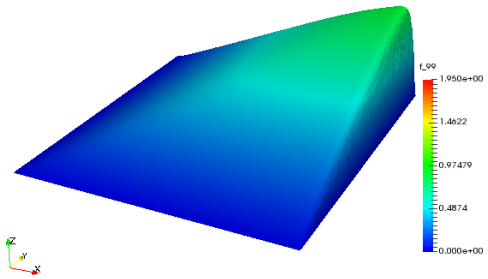
(a) mesh, $h \approx 0.14$, $\mathcal{N} = 174$



(b) $h \approx 0.013$, $\mathcal{N} = 12859$



(c) FE solution, $h \approx 0.14$, $\mathcal{N} = 174$



(d) FE solution, $h \approx 0.013$, $\mathcal{N} = 12859$

Figure 2.3: Two dimensional case, different mesh sizes

Several stabilization methods have been developed to fix the approximated solution without resorting to mesh refinement [49]. One type of stabilizations consists on adding some sort of artificial diffusion, in order to smooth the "jumps"(boundary or internal layers) that the exact solution can show. Let us note also that ε_0 is the coercivity constant of the bilinear form associated to the equation (2.1), while, when we are in an advection dominated situation, $||\boldsymbol{\beta}||_{L^\infty(\Omega)}$ is actually the continuity constant. So, enforcing somehow the coercivity constant we can obtain a better Céa error estimate [49].

Other stabilization techniques are inspired by the *spectral vanishing viscosity* method [35] and they work particularly with RB method, irrespectively on the stabilization op-

erated on *truth* solutions. We will not deal with this method, but we will be focused on adding–terms methods.

A first proposal can be to add to the left-hand side of the equation one of the following additional diffusion terms:

$$L^{ad}u = -h||\boldsymbol{\beta}||_{L^\infty(\Omega)}\Delta u, \tag{2.8}$$

$$L^{sd}u = -\frac{h}{||\boldsymbol{\beta}||_{L^\infty(\Omega)}}\mathrm{div}\left[(\boldsymbol{\beta}\cdot\nabla u)\boldsymbol{\beta}\right]. \tag{2.9}$$

The term (2.8) corresponds to the so called *artificial diffusion method* [49]. By adding this term, we are increasing the diffusion in all directions, producing then an unnecessary "crosswind" smoothing of the solution. To reduce the amount of diffusion introduced, we could use the term (2.9), which add diffusion only along the wind direction, thus avoiding the crosswind smoothing. The latter method is called *streamline upwind diffusion method* [49]. Both these methods are only weakly consistent, with a consistency error of $O(h)$. As a consequence, these methods can be useful only with $\mathbb{P}^1$ FE approximation. For further details about these methods we refer to [49].

2.1.2 Strongly consistent stabilization methods

As mentioned in the previous section, the main problem of the *artificial diffusion* and the *streamline upwind diffusion* methods is the fact that they are not consistent, which deteriorates the accuracy of the polynomial space used in the FE approximation. A way to avoid this kind of problem is to use a strongly consistent stabilization method. Several methods have been proposed and many of them can be considered as particular cases of the general class that we are going to introduce.

Let us consider now the advection-diffusion operator defined on $H_0^1(\Omega)$:

$$Lv = -\varepsilon\Delta v + \boldsymbol{\beta}\cdot\nabla u \quad \forall v \in H_0^1(\Omega). \tag{2.10}$$

We can split the operator L into its symmetric and skew-symmetric parts:

$$\text{symmetric part} \qquad L_S v = -\varepsilon\Delta u - \frac{1}{2}(\mathrm{div}\boldsymbol{\beta})v \tag{2.11}$$

$$\text{skew-symmetric part} \qquad L_{SS} v = \boldsymbol{\beta}\cdot\nabla v + \frac{1}{2}(\mathrm{div}\boldsymbol{\beta})v \tag{2.12}$$

and the following relation holds:

$$L = L_S + L_{SS}. \tag{2.13}$$

Symmetric and skew-symmetric parts can be recovered using the formulae:

$$\begin{aligned} L_S &= \frac{L + L^*}{2} \\ L_{SS} &= \frac{L - L^*}{2} \end{aligned} \tag{2.14}$$

where L^* is the adjoint operator.

We consider now the weak form of the problem (2.1), that is:

$$\begin{aligned} &\text{find } u \in H_0^1(\Omega) \text{ s.t.} \\ &a(u,v) = F(v) \quad \forall v \in H_0^1(\Omega) \end{aligned} \tag{2.15}$$

where a is the bilinear form associated to the advection diffusion operator

$$a(u,v) = \int_\Omega \varepsilon \nabla u \cdot \nabla v + \boldsymbol{\beta} \cdot \nabla u v, \quad v, w \in H_0^1(\Omega), \tag{2.16}$$

while F is the linear form defined by

$$F(v) = \int_\Omega f v, \quad v \in H_0^1(\Omega). \tag{2.17}$$

As in the previous section, let us suppose that we are given a regular triangulation $\mathcal{T}_h$. We consider the following piecewise polynomial approximation space:

$$\mathbb{P}^r(\mathcal{T}_h) = \{v \in H^1(\Omega) \text{ s.t. } v|_K \in \mathbb{P}^r(K), K \in \mathcal{T}_h\} \tag{2.18}$$

where $\mathbb{P}^r(K)$ is the space of polynomials of degree r on the element K. We will denote with $X^{\mathcal{N}}$ the space $\mathbb{P}^r(\mathcal{T}_h) \cap H_0^1(\Omega)$, where $\mathcal{N}$ is its dimension, i.e. the number of degrees of freedom.
We define the stabilization terms

$$s^{(\rho)}(u^{\mathcal{N}}, v^{\mathcal{N}}) = \sum_{K \in \mathcal{T}_h} \delta_K \left(L u^{\mathcal{N}}, \frac{h_K}{|\boldsymbol{\beta}|}(L_{SS} + \rho L_S) v^{\mathcal{N}} \right)_K \tag{2.19}$$

$$\phi^{(\rho)}(v^{\mathcal{N}}) = \sum_{K \in \mathcal{T}_h} \delta_K \left(f, \frac{h_K}{|\boldsymbol{\beta}|}(L_{SS} + \rho L_S) v^{\mathcal{N}} \right)_K \tag{2.20}$$

where $(\cdot,\cdot)_K$ is the inner scalar product in $L^2(K)$. The weights $\delta_K > 0$ for all $K \in \mathcal{T}_h$, have to be chosen as well as the parameter $\rho \in \mathbb{R}$, which identifies the method.

We can consider now the stabilized problem:

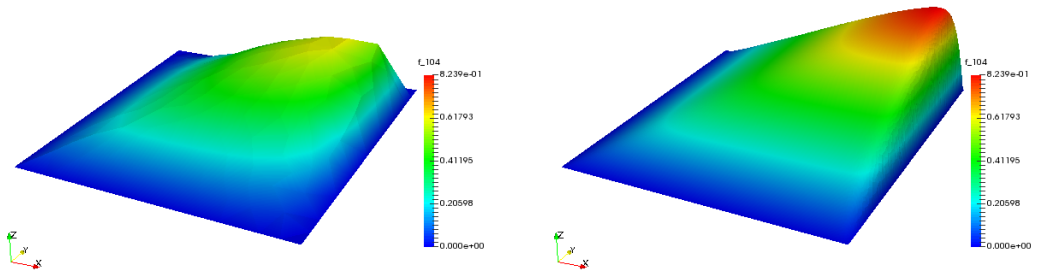
$$\begin{aligned} &\text{find } u \in X^{\mathcal{N}} \text{ s.t.} \\ &a(u^{\mathcal{N}}, v^{\mathcal{N}}; \boldsymbol{\mu}) + s^{(\rho)}(u^{\mathcal{N}}, v^{\mathcal{N}}) = F(v^{\mathcal{N}}) + \phi^{(\rho)}(v^{\mathcal{N}}) \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}}. \end{aligned} \tag{2.21}$$

Note that this formulation is strongly consistent, i.e. the continuous solution of (2.15) satisfies the variational equality (2.21).

We have actually defined a family of strongly consistent methods that can be identified through the parameter ρ, as we have said before. Several possible choice of ρ have been studied in literature. A first choice can be to set the parameter ρ equal to zero, thus defining the so called Streamline Upwind/Petrov Galerkin (SUPG) method [5, 20, 28, 29]. Another possibility is to choose $\rho = 1$ and the corresponding method is called Galerkin/Least-Squares (GALS) method [21]. The choice $\rho = -1$ leads to the Douglas-Wang/Galerkin (DWG) method [13].

Remark 1. If the polynomial approximation space chosen is $\mathbb{P}^1(\mathcal{T}_h)$ and the advection field is divergence free, i.e. $\operatorname{div}(\boldsymbol{\beta}) = 0$, any choice of the parameter ρ yields the same method.

Before going on with the analysis of the SUPG method, in figure 2.4 we show how this method can approximate the solution of (2.7) with $\varepsilon = \frac{1}{100}$. The mesh used are the same used in figure 2.3. We recall that the approximated solution given by the standard FE method was very unsatisfactory for the first mesh ($h \approx 0.14$ in figure 2.3(c)). Comparing figure 2.4(a) with 2.3(c) it is evident the smoothing of the boundary layer due to the increased diffusivity of the stabilized method. While the satisfactory solution ($h \approx 0.013$ in figure 2.3(d)) performed by FE is really similar to the stabilized one in figure 2.4(b).



(a) SUPG stabilized solution, $h \approx 0.14$, $\mathcal{N} = 174$ (b) SUPG stabilized solution, $h \approx 0.013$, $\mathcal{N} = 12859$

Figure 2.4: Two dimensional case, stabilized solutions

Analysis of the SUPG method

As in this work we will focus mainly on the SUPG method, we will now analyse its properties.

Denoting with $|||\cdot|||$ the energy norm associated to the bilinear form a, which turns out to be

$$|||v|||^2 = \varepsilon||\nabla v||^2_{L^2(\Omega)} + \frac{1}{2}||(\mathrm{div}\boldsymbol{\beta})^{\frac{1}{2}} v||^2_{L^2(\Omega)} \quad \forall v \in H^1_0(\Omega) \tag{2.22}$$

we define the SUPG norm on $H^1_0(\Omega)$ as

$$||v||^2_{SUPG} = |||v|||^2 + \sum_{K\in\mathcal{T}_h} \delta_K \left(L_{SS}v, \frac{h_k}{|\boldsymbol{\beta}|} L_{SS}v \right)_K \quad \forall v \in H^1_0(\Omega). \tag{2.23}$$

It holds that the SUPG bilinear form is coercive with respect to the SUPG norm. This is straightforward, even without any requirement on the parameters δ_K, if the polynomial approximation space is $\mathbb{P}^1(\mathcal{T}_h)$ and the advection field is divergence free, as the energy norm of the SUPG bilinear form is actually the SUPG norm. In general, we have the following theorem, as it is shown in [49]:

Theorem 2.1.1 (Stability)**.** *We assume that we are dealing with the advection dominated case (2.5) and that for any element $K \in \mathcal{T}_h$ parameter δ_K satisfies the conditions*

$$0 < \delta_K < C_r^{-1} \tag{2.24}$$

where C_r is the constant of the inverse inequality

$$\sum_{K\in\mathcal{T}_h} h_K^2 \int_K |\Delta v^{\mathcal{N}}|^2 \leq C_r ||\nabla v^{\mathcal{N}}||^2_{L^2(\Omega)} \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}} \tag{2.25}$$

in which r stands for the degree of the piecewise polynomial approximation space. Moreover, if the advection field is not divergence free, we suppose that we have a positive constant $d_0 > 0$ such that

$$0 < d_0 < -\nabla \cdot \boldsymbol{\beta} \tag{2.26}$$

and we require that

$$\delta_K h_K \leq \frac{|\boldsymbol{\beta}(x)|}{|\nabla \cdot \boldsymbol{\beta}(x)|} \quad \forall x \in K. \tag{2.27}$$

Then the bilinear form associated to the SUPG method is coercive with respect to the advection diffusion energy norm, that is:

$$a(v^{\mathcal{N}}, v^{\mathcal{N}}) + s^{(0)}(v^{\mathcal{N}}, v^{\mathcal{N}}) \geq \frac{1}{2} ||v^{\mathcal{N}}||_{SUPG}. \tag{2.28}$$

The theorem 2.1.1 yields the following result [49]:

Proposition 2.1.2. *There exists $C > 0$, independent of h, such that*

$$||u^{\mathcal{N}}||_{SUPG} \leq C||f||_{L^2(\Omega)} \tag{2.29}$$

where $u^{\mathcal{N}}$ is the solution of (2.21) with $\rho = 0$.

Acting like in [13, 49], it is also possible to prove the following convergence theorem:

Theorem 2.1.3 (Convergence)**.** *Assume that the advection dominated condition (2.5) holds and assume also that the space $X^{\mathcal{N}}$ satisfies the following approximation property: for each $v \in H_0^1(\Omega) \cap H^{k+1}(\Omega)$ there exists $\hat{v}^{\mathcal{N}} \in X^{\mathcal{N}}$ such that*

$$\begin{aligned} ||v - \hat{v}^{\mathcal{N}}||_{L^2(K)} + h_K ||\nabla(v - \hat{v}^{\mathcal{N}})||_{L^2(K)} + h_K^2 ||D^2(v - \hat{v}^{\mathcal{N}})||_{L^2(K)} \\ \leq C h_K^{r+1} |v|_{H^{r+1}(K)} \end{aligned} \tag{2.30}$$

for each $K \in \mathcal{T}_h$. Then the SUPG method has the following order of convergence:

$$||u - u^{\mathcal{N}}||_{SUPG} \leq C h^{r+\frac{1}{2}} |u|_{H^{r+1}(K)} \tag{2.31}$$

provided that $u \in H^{r+1}(K)$.

Proof. First of all we point out that if we choose the approximation space (2.18), for any $v \in H_0^1(\Omega) \cap H^{k+1}(\Omega)$ there exists an element $\hat{v}^{\mathcal{N}}$ fulfilling the condition (2.30). Indeed, it sufficient to choose

$$\hat{v}^{\mathcal{N}} = \Pi_h^r v \tag{2.32}$$

where Π_h^r is the piecewise polynomial interpolation operator (see [45] for definition).

We can now define

$$\sigma^{\mathcal{N}} := u^{\mathcal{N}} - \hat{u}^{\mathcal{N}}, \quad \eta := u - \hat{u}^{\mathcal{N}} \tag{2.33}$$

and we observe that

$$u - u^{\mathcal{N}} = \eta - \sigma^{\mathcal{N}} \tag{2.34}$$

We want estimate the quantity $||\sigma^{\mathcal{N}}||_{SUPG}$. We observe that, from theorem 2.1.3 and Galerkin orthogonality, we have

$$\frac{1}{2}||\sigma^{\mathcal{N}}||^2_{SUPG} \leq a(\sigma^{\mathcal{N}}, \sigma^{\mathcal{N}}) + s^{(0)}(\sigma^{\mathcal{N}}, \sigma^{\mathcal{N}}) = a(\eta, \sigma^{\mathcal{N}}) + s^{(0)}(\eta, \sigma^{\mathcal{N}}). \tag{2.35}$$

In order to make effective estimates, we write explicitly the right-hand side of (2.35):

$$\begin{aligned} a(\eta, \sigma^{\mathcal{N}}) + s^{(0)}(\eta, \sigma^{\mathcal{N}}) = &\varepsilon \int_\Omega \nabla\eta \cdot \nabla\sigma^{\mathcal{N}} + \int_\Omega \boldsymbol{\beta}\nabla\eta\sigma^{\mathcal{N}} + \\ &+ \sum_{K\in\mathcal{T}_h} \delta_K \left(-\varepsilon\Delta\eta + \boldsymbol{\beta}\cdot\nabla\eta, \frac{h_k}{|\boldsymbol{\beta}|}(\boldsymbol{\beta}\cdot\sigma^{\mathcal{N}} + \frac{1}{2}\mathrm{div}\boldsymbol{\beta}\sigma^{\mathcal{N}}) \right). \end{aligned} \tag{2.36}$$

Let us start by estimating the first term of the sum (2.36), using Young's inequality:

$$\varepsilon \int_\Omega \nabla\eta \cdot \nabla\sigma^{\mathcal{N}} \leq \frac{\varepsilon}{8}||\nabla\sigma^{\mathcal{N}}||^2_{L^2(\Omega)} + 2\varepsilon||\nabla\eta||^2_{L^2(\Omega)}. \tag{2.37}$$

As regards the second term, we know from Stokes theorem that

$$\int_\Omega \mathrm{div}(\eta\sigma^{\mathcal{N}}\boldsymbol{\beta}) = \int_{\partial\Omega} \eta\sigma^{\mathcal{N}}\boldsymbol{\beta}\cdot\boldsymbol{n} = 0. \tag{2.38}$$

Moreover, we can rewrite ad then estimate the second term, using Young inequality, as

$$\begin{aligned} &\int_\Omega \boldsymbol{\beta}\cdot\nabla\eta\sigma^{\mathcal{N}} = \int_\Omega -\eta\sigma^{\mathcal{N}}\mathrm{div}\boldsymbol{\beta} - \eta\boldsymbol{\beta}\cdot\nabla\sigma^{\mathcal{N}} = \\ &= -\int_\Omega \frac{1}{2}\eta\sigma^{\mathcal{N}}\mathrm{div}(\boldsymbol{\beta}) - \sum_{K\in\mathcal{T}_h}\int_K \eta\sqrt{\frac{h_K\delta_K}{|\boldsymbol{\beta}|}}\left(\boldsymbol{\beta}\cdot\nabla\sigma^{\mathcal{N}} + \frac{1}{2}\sigma^{\mathcal{N}}\mathrm{div}(\boldsymbol{\beta})\right)\sqrt{\frac{|\boldsymbol{\beta}|}{h_K\delta_K}} \leq \\ &\leq \frac{1}{4}\varepsilon\int_\Omega \nabla\sigma^{\mathcal{N}}\cdot\nabla\sigma^{\mathcal{N}} + \int_\Omega 2\eta^2\mathrm{div}\boldsymbol{\beta} + \frac{1}{8}\int_\Omega(\sigma^{\mathcal{N}})^2\mathrm{div}\boldsymbol{\beta} + \\ &\qquad + \frac{1}{4}\sum_{K\in\mathcal{T}_h}\frac{\delta_K h_K}{|\boldsymbol{\beta}|}\left(L_{SS}\sigma^{\mathcal{N}}, L_{SS}\sigma^{\mathcal{N}}\right)_K + 2\sum_{K\in\mathcal{T}_h}\left\|\sqrt{\frac{|\boldsymbol{\beta}|}{h_K\delta_K}}\eta\right\|^2_{L^2(K)} \leq \\ &\leq \frac{1}{4}||\sigma^{\mathcal{N}}||^2_{SUPG} + 2||\mathrm{div}\boldsymbol{\beta}||_{L^\infty(\Omega)}||\eta||^2_{L^2(\Omega)} + 2\sum_{K\in\mathcal{T}_h}\left\|\sqrt{\frac{|\boldsymbol{\beta}|}{h_K\delta_K}}\eta\right\|^2_{L^2(K)}. \end{aligned} \tag{2.39}$$

Finally, we have to care about the stabilization term in (2.36). Also in this case we can

resort to Young's inequality and obtain:

$$\begin{aligned}
&\sum_{K\in\mathcal{T}_h}\delta_K\left(-\varepsilon\Delta\eta+\boldsymbol{\beta}\cdot\nabla\eta,\frac{h_K}{|\boldsymbol{\beta}|}(\boldsymbol{\beta}\cdot\sigma^{\mathcal{N}}+\frac{1}{2}\mathrm{div}\boldsymbol{\beta}\sigma^{\mathcal{N}})\right)\leq\\
&\leq\frac{1}{8}||\sigma^{\mathcal{N}}||^2_{SUPG}+2\sum_{K\in\mathcal{T}_h}\delta_K\int_K\frac{h_K}{|\boldsymbol{\beta}|}(-\varepsilon\Delta\eta+\boldsymbol{\beta}\cdot\nabla\eta)^2\leq\\
&\leq\frac{1}{8}||\sigma^{\mathcal{N}}||^2_{SUPG}+4\sum_{K\in\mathcal{T}_h}\delta_K h_K\left\|\frac{\varepsilon}{\sqrt{|\boldsymbol{\beta}|}}\Delta\eta\right\|^2_{L^2(K)}+4\sum_{K\in\mathcal{T}_h}\delta_K h_K||\mathrm{div}\boldsymbol{\beta}||_{L^\infty(\Omega)}\left\|\nabla\eta\right\|^2_{L^2(K)}\leq\\
&\leq\frac{1}{8}||\sigma^{\mathcal{N}}||^2_{SUPG}+2\sum_{K\in\mathcal{T}_h}\delta_K||\boldsymbol{\beta}||_{L^\infty(\Omega)}h_K^3||\Delta\eta||^2_{L^2(K)}+4\sum_{K\in\mathcal{T}_h}\delta_K h_K||\mathrm{div}\boldsymbol{\beta}||_{L^\infty(\Omega)}\left\|\nabla\eta\right\|^2_{L^2(K)}
\end{aligned}\tag{2.40}$$

and in the last inequality we used the advection dominated condition (2.5). From inequalities (2.35), (2.37), (2.39), (2.40) we obtain that

$$\begin{aligned}
&\frac{1}{8}||\sigma^{\mathcal{N}}||^2_{SUPG}\leq 2||\nabla\eta||^2_{L^2(\Omega)}+2||\mathrm{div}\boldsymbol{\beta}||_{L^\infty(\Omega)}||\eta||^2_{L^2(\Omega)}+\sum_{K\in\mathcal{T}_h}2\left\|\sqrt{\frac{|\boldsymbol{\beta}|}{h_K\delta_K}}\eta\right\|^2_{L^2(K)}+\\
&+2\sum_{K\in\mathcal{T}_h}\delta_K||\boldsymbol{\beta}||_{L^\infty(\Omega)}h_K^3||\Delta\eta||^2_{L^2(K)}+4\sum_{K\in\mathcal{T}_h}\delta_K h_K||\mathrm{div}\boldsymbol{\beta}||_{L^\infty(\Omega)}||\nabla\eta||^2_{L^2(K)}.
\end{aligned}\tag{2.41}$$

Now we can get rid of the dependence of h_K in the third term of the right-hand side thanks to (2.5), and rearrange the inequality:

$$\begin{aligned}
&\frac{1}{8}||\sigma^{\mathcal{N}}||^2_{SUPG}\leq\sum_{K\in\mathcal{T}_h}2\left(||\mathrm{div}\boldsymbol{\beta}||_{L^\infty(\Omega)}+\left\|\frac{\boldsymbol{\beta}}{\sqrt{2\varepsilon}}\right\|_{L^\infty(\Omega)}\right)||\eta||^2_{L^2(K)}+\\
&+(2+||\mathrm{div}\boldsymbol{\beta}||_{L^\infty(\Omega)}\delta_K)h_K||\nabla\eta||^2_{L^2(K)}+2\delta_K||\boldsymbol{\beta}||_{L^\infty(\Omega)}h_K^3||\Delta\eta||^2_{L^2(K)}
\end{aligned}\tag{2.42}$$

From this, we now that there exist C_1, C_2, C_3 and C_4 such that:

$$\begin{aligned}
||\sigma^{\mathcal{N}}||^2_{SUPG}&\leq C_1||\eta||^2_{L^2(K)}+C_2h||\nabla\eta||^2_{L^2(K)}+C_3h^3||\Delta\eta||^2_{L^2(K)}\leq\\
&\leq C_4\frac{1}{h}\left(\sqrt{h}||\eta||_{L^2(K)}+h||\nabla\eta||_{L^2(K)}+h^2||\Delta\eta||_{L^2(K)}\right)^2\leq\\
&\leq C_4\frac{1}{h}\left(||\eta||_{L^2(K)}+h||\nabla\eta||_{L^2(K)}+h^2||\Delta\eta||_{L^2(K)}\right)^2.
\end{aligned}\tag{2.43}$$

Exploiting the condition (2.30), we obtain that:

$$||\sigma^{\mathcal{N}}||_{SUPG}\leq Ch^{r+\frac{1}{2}}|u|_{H^{r+1}(\Omega)}\tag{2.44}$$

where the constant C does not depend on h_K and on $\mathbb{P}\mathtt{e}_K(x)$. It depends on other quantities such as $\boldsymbol{\beta}$ and ε.
Using the same procedure, we can have:

$$||\eta||_{SUPG}\leq Ch^{r+\frac{1}{2}}|u|_{H^{r+1}(\Omega)}.\tag{2.45}$$

Thus, by triangular inequality, we can conclude the theorem. □

Remark 2. As pointed out in [13], if the advection dominated condition (2.5) is not fulfilled for all $K \in \mathcal{T}_h$ we locally lose even the h^r convergence rate of the standard FE method. To recover at least the standard convergence rate, we need that the coefficient C_K of the term $\sum_{K\in\mathcal{T}_h} C_K||\Delta\eta||^2_{L^2(K)}$ in (2.40) must scale, at least, as h^2. A possible way to overcome to this trouble is to act on weights δ_K, distinguishing between the elements for which $\mathbb{P}\mathtt{e}_K(x) > 1$ and $\mathbb{P}\mathtt{e}_K(x) \leq 1$. Unfortunately, in RB context, this strategy does not allow an immediate affine representation (1.9) of the bilinear form.

2.2 Stabilized reduced basis: introduction and numerical examples

In this section we will try to design an efficient stabilization procedure for the RB method. In particular, we will focus on a couple of problems that were taken in consideration in [41, 43]. Finally, we will discuss the differences between two different approaches: *Offline-Online* stabilization method and an *Offline-only* stabilized one, when used to approximate the solution of and advection-diffusion problem:

$$-\varepsilon(\boldsymbol{\mu})\Delta u(\boldsymbol{\mu}) + \boldsymbol{\beta}(\boldsymbol{\mu})\cdot\nabla u(\boldsymbol{\mu}) = 0 \quad \text{onto } \Omega_p(\boldsymbol{\mu}) \subset \mathbb{R}^2. \tag{2.46}$$

Offline-Online approach means that the Galerkin projections are performed, in both *Offline* and *Online* stages, with respect to the SUPG stabilized bilinear form, that are:

$$\begin{aligned} a_{stab}(u^{\mathcal{N}}, v^{\mathcal{N}}) = &\int_\Omega \varepsilon\nabla u^{\mathcal{N}}\cdot\nabla v^{\mathcal{N}} + \left(\boldsymbol{\beta}\cdot\nabla u^{\mathcal{N}}\right) v^{\mathcal{N}} + \\ &+ \sum_{K\in\mathcal{T}_h} \delta_K \int_K (-\varepsilon\Delta u^{\mathcal{N}} + \boldsymbol{\beta}\cdot\nabla u^{\mathcal{N}}) \left(\frac{h_K}{|\boldsymbol{\beta}|}\boldsymbol{\beta}\cdot\nabla v^{\mathcal{N}}\right) \end{aligned} \tag{2.47}$$

with $u^{\mathcal{N}}, v^{\mathcal{N}} \in X^{\mathcal{N}} \subset \mathbb{P}^r(\mathcal{T}_h)$, where $\mathcal{T}_h$ is a triangulation of Ω. This is a bilinear coercive form, so we can apply the already developed theory in order to use the reduced basis method. The alternative method we want to study – the *Offline-only* stabilized method – consists in using the stabilized form (2.47) only during the *Offline* stage and then projecting, during the *Online* stage, with respect to the standard advection-diffusion bilinear form. The underlying heuristic idea is to be able to build stabilized basis, to avoid the *Online* stabilization.

In both these approaches we have to provide an affine expansion like (1.9) and (1.10) of the involved bilinear forms and right-hand side operators. If this is not possible in an exact way, we could resort to some interpolation techniques (e.g. empirical interpolation [4, 12, 15, 19, 33]. As we always need an affine expansion, the advantage of using the *Offline-only* method could be a certain reduction of the computational cost, that could be significant if the number of affine terms is very high.

We will start from the study of a couple of simple test problems, for which is straightforward to obtain the affine expansion. We will consider first the Poiseulle-Graetz problem [25, 41, 43, 47], which shows strong instability effects that can be fixed with SUPG stabilization method. Secondly, we will study another example where the parameter dependence will affect the angle of the advection field.

Let us make a remark about the notation we will use from now on. First of all, we will write explicitly the FE space dimension $\mathcal{N}$ only when it will be necessary. Moreover, as we will use only the SUPG stabilization method, we will omit the value of ρ in the stabilization terms.

2.2.1 Graetz-Poiseuille problem

As a first example, we will focus on a Graetz problem [14, 25, 47, 52] where we have two parameters: one physical (the diffusivity coefficient μ_1, which is proportional to the Péclet number) and one geometrical (the length of the domain $\mu_2 + 1$). The Graetz problem deals with steady forced heat convection (advective phenomenon) combined with heat conduction (diffusive phenomenon) in a duct with walls at different temperature. Let us define $\boldsymbol{\mu} = (\mu_1, \mu_2)$ with both μ_1 and μ_2 positive, real numbers. Let $\Omega_p(\boldsymbol{\mu})$ be the rectangle $(0, 1 + \mu_2) \times (0, 1)$ in $\mathbb{R}^2$. The domain is shown in figure 2.5. The problem is to find a

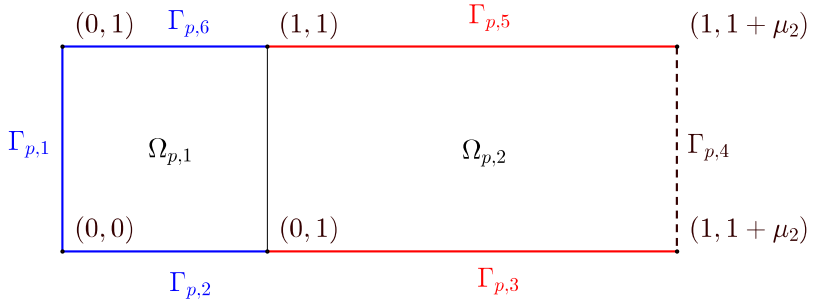


Figure 2.5: Geometry of Graetz problem. Parametrized domain. Boundary conditions: homogeneous Dirichlet on blue sides, $u = 1$ on red sides, homogeneous Neumann on the dashed side

solution $u(\boldsymbol{\mu})$, representing the temperature distribution, such that:

$$\begin{cases} -\frac{1}{\mu_1}\Delta u(\boldsymbol{\mu}) + 4y(1-y)\partial_x u(\boldsymbol{\mu}) = 0 & \text{in } \Omega_p(\boldsymbol{\mu}) \\ u(\boldsymbol{\mu}) = 0 & \text{on } \Gamma_{p,1}(\boldsymbol{\mu}) \cup \Gamma_{p,2}(\boldsymbol{\mu}) \cup \Gamma_{p,6}(\boldsymbol{\mu}) \\ u(\boldsymbol{\mu}) = 1 & \text{on } \Gamma_{p,3}(\boldsymbol{\mu}) \cup \Gamma_{p,5}(\boldsymbol{\mu}) \\ \frac{\partial u}{\partial \nu} = 0 & \text{on } \Gamma_{p,4}(\boldsymbol{\mu}). \end{cases} \tag{2.48}$$

In order to use a RB approach, we need to set a reference domain Ω that we choose $\Omega = (0, 1) \times (0, 2)$, that is the original domain for parameter $\mu_2 = 1$, i.e. $\Omega = \Omega_p(\mu_1, 1)$. It is useful to subdivide the reference domain into two subdomains, so we define $\Omega^1 = (0, 1) \times (0, 1)$ and $\Omega^2 = (1, 2) \times (0, 1)$. Now, as we have seen in section 1.1.1, we need the affine transformation that maps the reference domain into the original one [19, 41, 47, 51], so we define:

$$\begin{aligned} &T^1(\boldsymbol{\mu}) : \Omega^1 \to \Omega_{p,1}(\boldsymbol{\mu}) \subset \mathbb{R}^2 \\ &T^1\left(\begin{pmatrix} x \\ y \end{pmatrix}; \boldsymbol{\mu}\right) = \begin{pmatrix} x \\ y \end{pmatrix} \end{aligned} \tag{2.49}$$

that is the identity on the first subdomain Ω^1 and

$$\begin{gathered} T^2(\boldsymbol{\mu}) : \Omega^2 \to \Omega_{p,2}(\boldsymbol{\mu}) \subset \mathbb{R}^2 \\ T^2\left(\begin{pmatrix} x \\ y \end{pmatrix}; \boldsymbol{\mu}\right) = \begin{pmatrix} \mu_2 x \\ y \end{pmatrix} + \begin{pmatrix} 1-\mu_2 \\ 0 \end{pmatrix} = \boldsymbol{G}^2 \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 1-\mu_2 \\ 0 \end{pmatrix} \end{gathered} \tag{2.50}$$

where

$$\boldsymbol{G}^2 = \begin{pmatrix} \mu_2 & 0 \\ 0 & 1 \end{pmatrix}. \tag{2.51}$$

If we glue together these two transformations, for each $\boldsymbol{\mu} \in \mathcal{D}$ we actually define a transformation $T(\boldsymbol{\mu})$ over the whole domain Ω. Note that $T(\boldsymbol{\mu})$ is a continuous one-to-one transformation.

The weak formulation of the Poiseuille-Graetz problem is the following one:

$$\begin{gathered} \text{find } u_p(\boldsymbol{\mu}) \in V_p := \{v_p \in H^1(\Omega_p) \text{ s.t. } v \text{ satisfies BC in (2.48)}\} \text{ s.t.} \\ a(u_p(\boldsymbol{\mu}), v_p; \boldsymbol{\mu}) = 0 \quad v_p \in H_0^1(\Omega) \end{gathered} \tag{2.52}$$

where

$$a(u_p, v_p; \boldsymbol{\mu}) := \int_{\Omega_p} \frac{1}{\mu_1} \nabla u \cdot \nabla v + 4y(1-y)\partial_x uv. \tag{2.53}$$

We know from the general theory of PDEs that the problem (2.52) admits a unique solution. Now we want to set the FE approximation of this problem. Let $\mathcal{T}_h$ be a proper triangulation of Ω. The FE problem will be:

$$\begin{gathered} \text{find } u_h(\boldsymbol{\mu}) \in V_h := \{v_h \in \mathbb{P}^r(\mathcal{T}_h) \text{ s.t. } v_h \text{ satisfies BC in (2.48)}\} \text{ s.t.} \\ a(u_h(\boldsymbol{\mu}), v^{\mathcal{N}}; \boldsymbol{\mu}) = 0 \quad v^{\mathcal{N}} \in X^{\mathcal{N}} \end{gathered} \tag{2.54}$$

with $X^{\mathcal{N}}$ defined as the subspace of $\mathbb{P}^r(\mathcal{T}_h)$ made up by the functions that vanish on the boundary of Ω. Finally, let us define the function l as a lifting in $\mathbb{P}^r(\mathcal{T}_h)$ of the Dirichlet boundary conditions. We can now define $u^{\mathcal{N}}(\boldsymbol{\mu}) = u_h(\boldsymbol{\mu}) - l$ that belongs to $X^{\mathcal{N}}$. Thus we obtain the final FE formulation of our problem:

$$\begin{gathered} \text{find } u^{\mathcal{N}}(\boldsymbol{\mu}) \in X^{\mathcal{N}} \text{ s.t.} \\ a(u^{\mathcal{N}}(\boldsymbol{\mu}), v^{\mathcal{N}}; \boldsymbol{\mu}) = F(v^{\mathcal{N}}) \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}} \end{gathered} \tag{2.55}$$

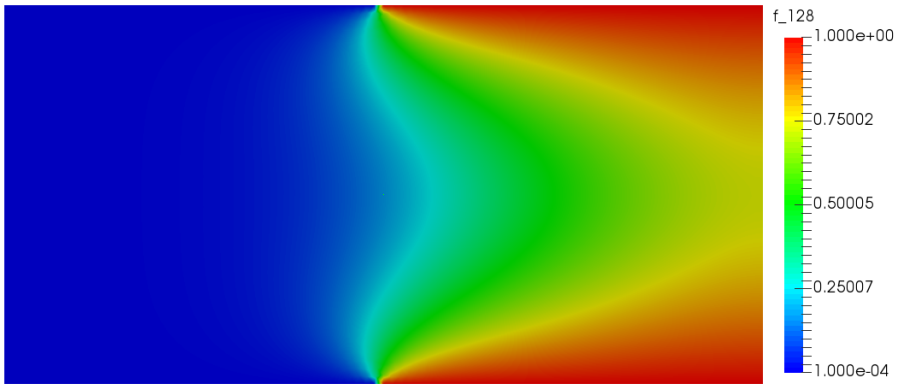
where

$$F(v^{\mathcal{N}}; \boldsymbol{\mu}) := -a(l, v^{\mathcal{N}}; \boldsymbol{\mu}). \tag{2.56}$$

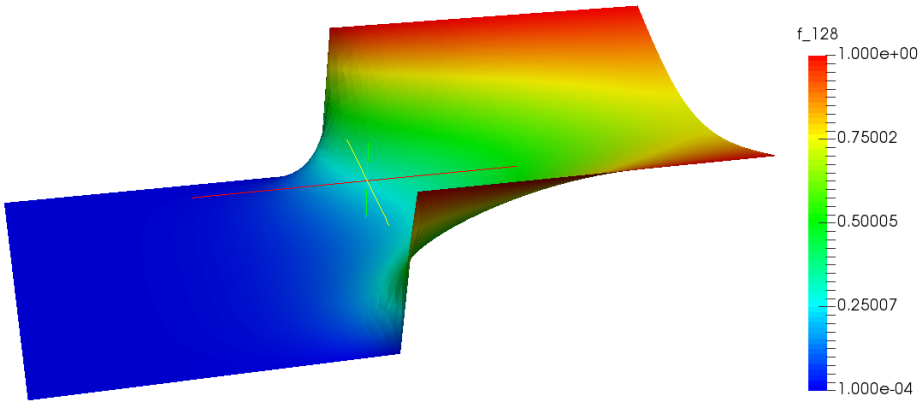
When the parameter μ_1 takes "small" values we do not have instability problems. More precisely, we can obtain stable solution if

$$\mathbb{P}e_K := \frac{\mu_1 h_K}{|\boldsymbol{\beta}|} = \frac{\mu_1 h_K}{|4y(1-y)|} < 1 \quad \forall K \in \mathcal{T}_h \tag{2.57}$$

that is when the advection dominated condition (2.5) is not fulfilled. In figure 2.6 the approximated $\mathbb{P}^1$-FE solution obtained for $\mu_1 = 6$ is shown. We can use the standard RB method to approximate the solution of the problem (2.48) for a parameter μ_1 range from 1 to 10.

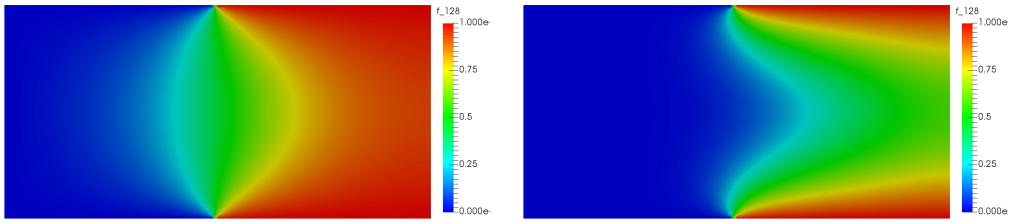


(a)



(b)

Figure 2.6: FE solution for $\mu_1 = 6$



(a) $\mu_1 = 1$ (b) $\mu_1 = 10$

Figure 2.7: FE solution for low Péclet numbers, $\mu_1 = 1$ (left), $\mu_1 = 10$ (right)

In figure 2.8 we report the energy norm of the difference between the RB solution and the FE solution (RB approximation error) as a function of the parameter μ_1, while we have fixed the parameter $\mu_2 = 1$. More precisely, in figure 2.8 we show the linear interpolation of the RB approximation error computed for 200 equispaced parameter values between 1 and 10. The local minima are located in correspondence of the parameter values selected by the

greedy algorithm [19, 47], where, clearly, the error tends to vanish. This phenomenon is clearly expected because, since we are using Lagrange basis, our RB solution "interpolates" exactly the truth manifold (1.1) in the "interpolation nodes" represented by the snapshot solutions.
In figure 2.7 we show some representative RB solution computed in correspondence of some value of the parameter μ_1. The dimension of the RB space is $N = 6$.

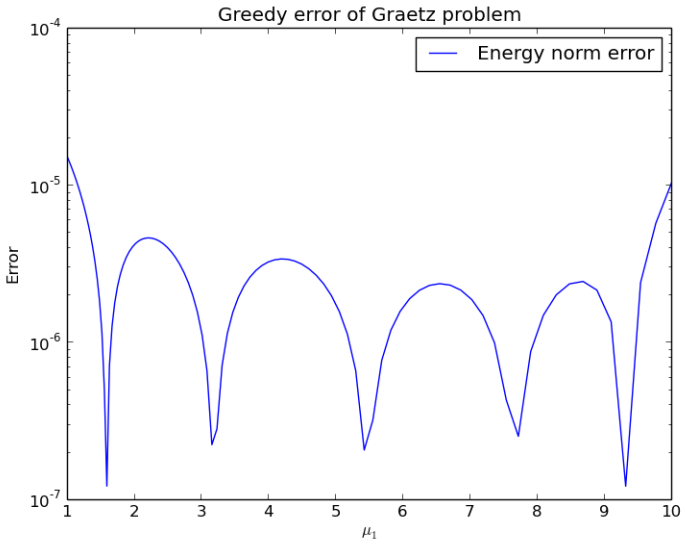
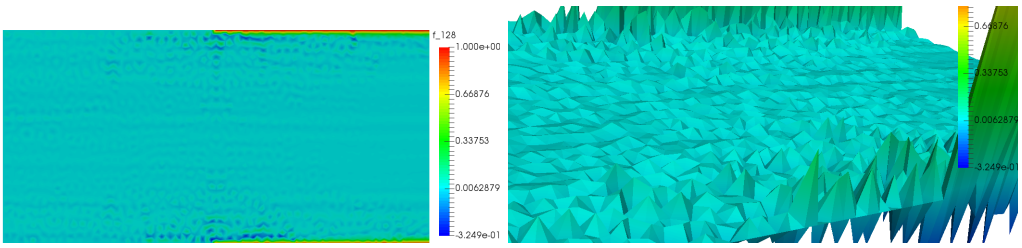


Figure 2.8: RB error for varying Péclet number, $\mu_1 \in [1, 10]$, $\mu_2 = 1$

More interesting is when the Péclet number assumes higher values, for which the condition (2.57) is not fulfilled. In figure 2.9 the solution obtained by using a FE approximation without stabilization at $\mu_1 = 10^5$ is represented. Clearly, the FE solution is distant from the exact one.



(a) $\mu_1 = 10^5$ (b) $\mu_1 = 10^5$, zoom on the central part

Figure 2.9: FE solution for high Péclet numbers, $\mu_1 = 10^5$, $\mu_2 = 1$

Anyway, even in this case we can perform a RB approximation of the solution, but the RB solutions reflect all the instability problems of the FE solution, as we can see in figure 2.10. For this simple case, if we let the parameter range from 10^4 to 10^5 the greedy algorithm converges and the energy norm of the difference between the RB solution and

the FE solution behaves as for lower values of the Péclet number, as we can see in figure 2.10.

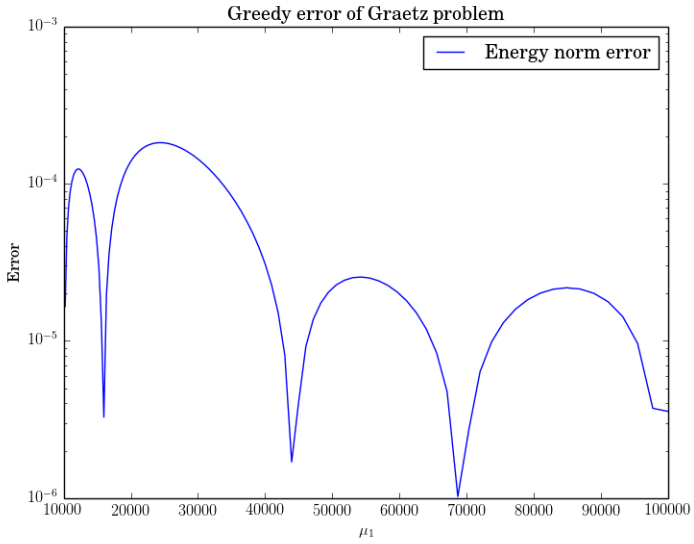


Figure 2.10: RB error for varying Péclet number, $\mu_1 \in \left[10^4, 10^5\right]$, $\mu_2 = 1$

This happens because the "target" of the RB approach is to approximate the exact continuous solution of the problem by trying to recover the FE solution using a significantly lower number of degrees of freedom. The point is now that the FE solution is not a good approximation of the exact one.

A possible way to fix this instability problems could be to use some stabilization methods. We chose the SUPG stabilization method. First of all we have to impose the stabilization correction to the weak formulation (2.54). We thus introduce the stabilization terms. To do so, let us define a mesh $\mathcal{T}_h$ defined on the reference domain Ω and let us call $\mathcal{T}_h^1$ and $\mathcal{T}_h^2$ the restrictions of $\mathcal{T}_h$ onto Ω_1 and Ω_2 respectively. We can also define a mesh on $\Omega_p(\boldsymbol{\mu})$ just by taking the image of $\mathcal{T}_h$ through the transformation $T(\cdot, \boldsymbol{\mu})$, that is:

$$\mathcal{T}_{h,p}(\boldsymbol{\mu}) = \{K_p(\boldsymbol{\mu}) = T(k; \boldsymbol{\mu}) | K \in \mathcal{T}_h\}. \tag{2.58}$$

We can now write the stabilization term, for the $\mathbb{P}^1$-FE case, to be added to the left-hand side:

$$s(u_h, v_h; \boldsymbol{\mu}) := \sum_{K_p(\boldsymbol{\mu}) \in \mathcal{T}_{h,p}(\boldsymbol{\mu})} \delta_{K_p(\boldsymbol{\mu})} \int_{K_p(\boldsymbol{\mu})} (4y(1-y)\partial_x u_h)(h_{K_p(\boldsymbol{\mu})} \partial_x v_h). \tag{2.59}$$

Now we have to set the problem onto the reference domain, thus our problem turns out to be:

$$\begin{aligned} &\text{find } u_h(\boldsymbol{\mu}) \in V_h := \{v_h \in \mathbb{P}^r(\mathcal{T}_h) \text{ s.t. } v_h \text{ satisfies BC in (2.48)}\} \text{ s.t.} \\ &\qquad a(u_h(\boldsymbol{\mu}), v^{\mathcal{N}}; \boldsymbol{\mu}) + s(u_h(\boldsymbol{\mu}; v^{\mathcal{N}}; \boldsymbol{\mu}) = 0 \quad v^{\mathcal{N}} \in X^{\mathcal{N}} \end{aligned} \tag{2.60}$$

where $X^{\mathcal{N}}$ is defined as in the previous section, a is:

$$\begin{aligned} a(u_h, v_h; \boldsymbol{\mu}) := &\int_{\Omega^1} \frac{1}{\mu_1} \nabla u_h \nabla v_h + 4y(1-y)\partial_x u_h v_h + \\ &+ \int_{\Omega^2} \frac{1}{\mu_1 \mu_2} \partial_x u_h \partial_x v_h + \frac{\mu_2}{\mu_1} \partial_x u_h \partial_y v_h + 4\mu_2 y(1-y)\partial_x u_h v_h \end{aligned} \tag{2.61}$$

and s is:

$$\begin{aligned} s(u_h, v_h; \boldsymbol{\mu}) := &\sum_{K \in \mathcal{T}_h^1} h_K \int_K (4y(1-y)\partial_x u_h)\partial_x v_h + \\ &+ \sum_{K \in \mathcal{T}_h^1} \frac{h_K}{\sqrt{\mu_2}} \int_K (4y(1-y)\partial_x u_h)\partial_x v_h. \end{aligned} \tag{2.62}$$

By introducing a lifting of the Dirichlet boundary condition and setting $u^{\mathcal{N}}(\boldsymbol{\mu}) = u_h(\boldsymbol{\mu}) - l$, we can obtain the final stabilized FE formulation:

$$\begin{aligned} &\text{find } u^{\mathcal{N}}(\boldsymbol{\mu}) \in X^{\mathcal{N}} \text{ s.t.} \\ &a(u^{\mathcal{N}}(\boldsymbol{\mu}), v^{\mathcal{N}}; \boldsymbol{\mu}) + s(u^{\mathcal{N}}(\boldsymbol{\mu}); v^{\mathcal{N}}; \boldsymbol{\mu}) = F(v^{\mathcal{N}}; \boldsymbol{\mu}) + F^s(v^{\mathcal{N}}; \boldsymbol{\mu}) \quad v^{\mathcal{N}} \in X^{\mathcal{N}} \end{aligned} \tag{2.63}$$

where F is defined in (2.56) and

$$F^s(v^{\mathcal{N}}) := -s(l, v^{\mathcal{N}}). \tag{2.64}$$

Let us call a_{stab} the bilinear form and F_{stab} the right-hand side, that is

$$\begin{aligned} a_{stab} &= a + s \\ F_{stab} &= F + F^s. \end{aligned} \tag{2.65}$$

We point out that for $K \in \mathcal{T}_h^2$ we should choose $\delta_{K_p}(\boldsymbol{\mu})$ such that $\delta_{K_p(\boldsymbol{\mu})} h_{K_p(\boldsymbol{\mu})} = h_K \sqrt{\mu_2}$. The underlying idea is that we would like to choose $\delta_{K_p(\boldsymbol{\mu})} = 1$ but we have to consider how the element diameter transforms, that is $h_{K_p(\boldsymbol{\mu})} \approx h_K \sqrt{J(\boldsymbol{\mu})} = h_K \sqrt{\mu_2}$. This rescaling is done mainly for preserving the convergence rate of the SUPG method. We need to make an assumption like this also because it would not make any sense, in an RB point of view, to compute Online every exact value of $h_{K_p(\boldsymbol{\mu})}$. Indeed, the Online stage of the RB method actually forgets about the triangulation.

Recalling remark 2, we want to observe that by using a weighting that depends on both parameter and element size we lose the affinity assumption (1.9) on the bilinear form, or better, we lose that assumption with a number of affine terms Q_a independent of $\mathcal{N}$. So, if we are facing problems in which the advection dominated condition (2.5) is not fulfilled for all $K \in \mathcal{T}_h$ and we want to rigorously recover the convergence order of the FE method, in order to resort to a weighting $\delta = \delta(x, \boldsymbol{\mu})$ (as proposed in [13]) we need to exploit some interpolation techniques involving the empirical interpolation [4, 19]. In this case it would be also worth to be checked if it were possible to define a weighting that does not depend on each h_K, but on the mesh size h, under suitable regularity assumptions [29].

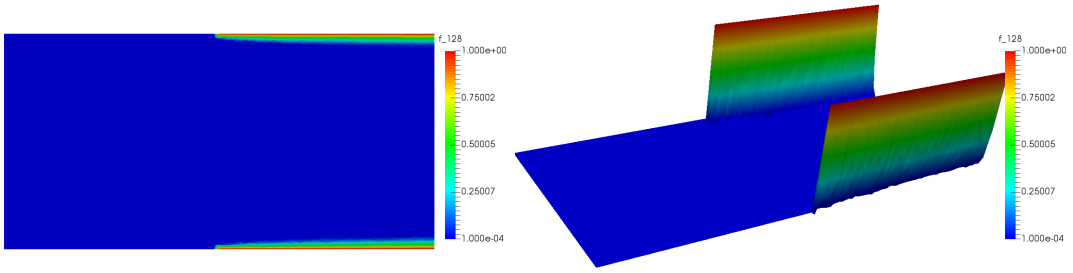
We would like also to recall that the convergence performances of the stabilization method depend on the regularity properties of the mesh. So, as the meshes $T_{h,p}(\boldsymbol{\mu})$ we are

actually using to stabilize on the original domains are the image through T of the triangulation defined on the reference domain, we should guarantee that the transformation T does not worsen the properties of the reference triangulation. In our numerical tests the reference domain will be the one corresponding to $\mu_2 = 1$ and we will let the parameter range from 0.5 to 4, so we will not have an excessive deformation. We will also use a quite coarse mesh (mesh size $0.019 < h < 0.037$) and high values for μ_1 (we will first try from 10^4 to 10^5, then from 10^5 to 10^6 and then from 1 to 10^6) in order to have significant instability.

The point is that the boundary layer arise in an area in which the norm of the advection field (and thus the value of the local Péclet number) is relatively small (near upper and lower walls, where $|\boldsymbol{\beta}| \approx 0.01$ on quadrature points. We will have that the Péclet number will assume different value depending on the choice of the diffusivity range.

$$\begin{aligned}
\mathbb{P}\mathrm{e}_K(x) = \frac{|\boldsymbol{\beta}(x)|h_k}{2\varepsilon(x)} &\in \left[\frac{0.01 \cdot 0.02}{2 \cdot 10^{-4}} = 1, \frac{1 \cdot 0.03}{2 \cdot 10^{-5}} \approx 10^3\right] \quad \text{for } \mu_1 \in \left[10^4, 10^5\right] \\
\mathbb{P}\mathrm{e}_K(x) = \frac{|\boldsymbol{\beta}(x)|h_k}{2\varepsilon(x)} &\in \left[\frac{0.01 \cdot 0.02}{2 \cdot 10^{-5}} = 10, \frac{1 \cdot 0.03}{2 \cdot 10^{-6}} \approx 10^4\right] \quad \text{for } \mu_1 \in \left[10^5, 10^6\right] \qquad (2.66) \\
\mathbb{P}\mathrm{e}_K(x) = \frac{|\boldsymbol{\beta}(x)|h_k}{2\varepsilon(x)} &\in \left[\frac{0.01 \cdot 0.02}{2 \cdot 1} = 10^{-2}, \frac{1 \cdot 0.03}{2 \cdot 10^{-6}} \approx 10^4\right] \quad \text{for } \mu_1 \in \left[1, 10^6\right]
\end{aligned}$$

In figure 2.11 we show a solution computed using the SUPG stabilization. The differences with the non-stabilized figure 2.9 are clear, in particular along the boundary layer, where the stabilization deletes every peak, which were not part of the exact solution.

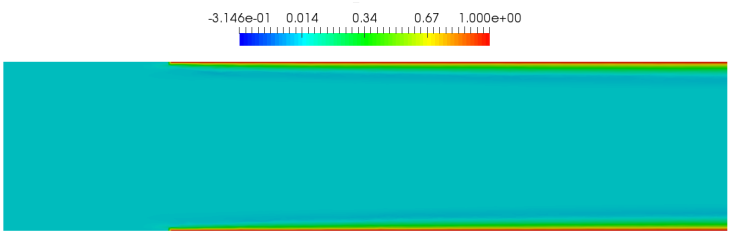


(a) Stabilized, $\mu_1 = 10^5$ (b) Stabilized, $\mu_1 = 10^5$

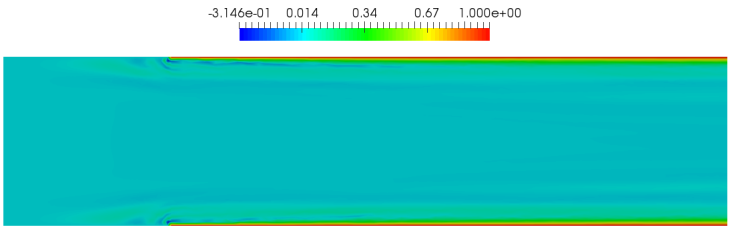
Figure 2.11: Stabilized SUPG, FE solution, $\mu_1 = 10^5$

Performing the Greedy algorithm with and without stabilization we obtain solution like in figure 2.12. As we can see, the Offline-Online stabilized RB solution is showing a behaviour similar to the exact one, while the Offline-only stabilized RB solution still has some noise near the boundary layer and some peaks near discontinuities of solution at top and bottom walls.

Now we will compare timing we need to perform all the computation in different cases. First, to compute a FE solution in this case (we are using a space where $\mathcal{N} = 4369$) we used in the stabilized case $4.11 \cdot 10^{-2}$ seconds, while in a non stabilized case $3.6 \cdot 10^{-2}$ seconds. To compute one Online stabilized RB solution we need $5.12 \cdot 10^{-3}$ seconds, while



(a) Offline-Online stabilized, $\boldsymbol{\mu} = (10^{4.8}, 3.3)$



(b) Offline Stabilized, $\boldsymbol{\mu} = (10^{4.8}, 3.3)$

Figure 2.12: RB solution, stabilized Offline-Online and Offline, $\boldsymbol{\mu} = (10^{4.8}, 3.3)$

a non stabilized Online RB solution needs $1.51 \cdot 10^{-3}$ seconds. Finally, the time used to compute the whole offline phase (included the SCM algorithm) is 241 seconds.

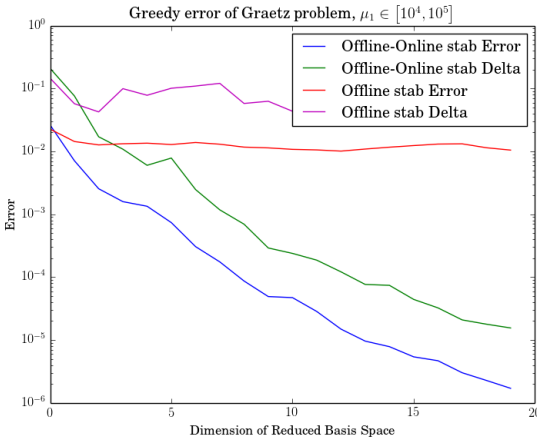
Now, we want to compare errors between stabilized FE solutions and RB solutions in both situations (Online–Offline stabilized and Offline–only stabilized) for different ranges of μ_1. In figure 2.13 we can see this comparison through the energy norm error computed over a sample of $\mathcal{D}$ and the $\Delta_N(\boldsymbol{\mu})$ error bound performed as in (1.53). We can compare different choices of $\mathcal{D} = [\mu_{1,min}, \mu_{1,max}] \times [0.5, 4]$, but in every case we can notice several behaviours. The Online–Offline stabilization has an error that converge to a really small error as the RB space enriches its dimension. The error of the Offline–only stabilized solutions stays over 10^{-2}, this is clearly higher than the other, because in the Online phase we are solving an equation which is different from the Offline one.

Nevertheless, there exists an error bound for this error which is sharper than RB one [40, 41]. Of course it will be of order of h_K as the stabilization that we have introduced has the same order, and it depends on the tolerance ε^* of the Greedy algorithm to compute the reduced space N. One can prove that the error between stabilized FE solution and Offline-only stabilized RB solution is such that:

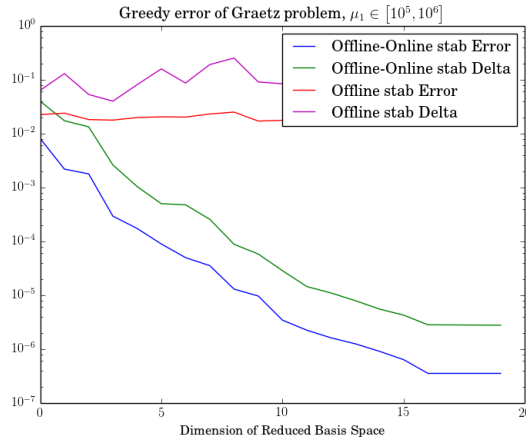
$$\begin{aligned} |||u_N(\boldsymbol{\mu}) - u^{stab,\mathcal{N}}(\boldsymbol{\mu})|||_{\boldsymbol{\mu}} \leq & h_{max}(\boldsymbol{\mu}) C(\boldsymbol{\mu}) ||\boldsymbol{\beta} \cdot \nabla u^{stab,\mathcal{N}}(\boldsymbol{\mu})||_{L^2(\Omega_p(\boldsymbol{\mu}))} + \\ & + (1 + h_{max}(\boldsymbol{\mu}) C(\boldsymbol{\mu})^2 ||\boldsymbol{\beta}||_{L^\infty(\Omega_p(\boldsymbol{\mu}))}) \varepsilon^*. \end{aligned} \tag{2.67}$$

we point out that this bound depends on the L^2 norm of the streamline derivative. This means that the Offline–only method has better performances when applied to problems in which the strongest variations occur along a direction orthogonal to the advection field. This could happen in the cases in which boundary layers are parallel to the advection field, e.g. the Graetz-Poiseuille problem.

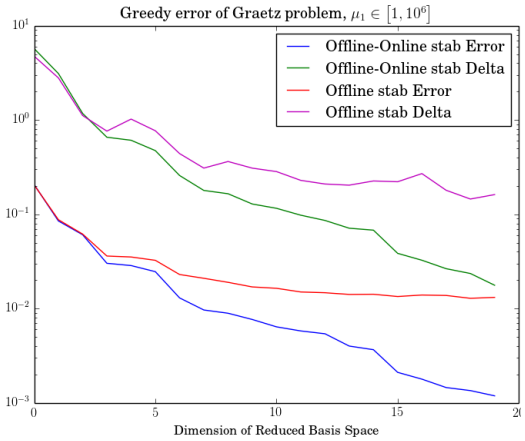
We will discuss in chapter 4 how we can partially use the Offline–only stabilization



(a) $\mu_1 \in (10^4, 10^5)$



(b) $\mu_1 \in (10^5, 10^6)$



(c) $\mu_1 \in (1, 10^6)$

Figure 2.13: Error comparison between Offline and Online-Offline stabilization

(that can be useful in case of very heavy computation for stabilization terms).

2.2.2 Propagating front in a square problem

In this section we will test the stabilization method for another test case, again in the RB context.

The problem we want to present is set over a squared domain $\Omega \subset \mathbb{R}^2$, as sketched in figure 2.14, and it has two parameter $\mu_1, \mu_2 \in \mathbb{R}$. It is the following:

$$\begin{cases} -\frac{1}{\mu_1}\Delta u(\boldsymbol{\mu}) + (\cos\mu_2, \sin\mu_2)\cdot\nabla u(\boldsymbol{\mu}) = 0 & \text{in } \Omega \\ u(\boldsymbol{\mu}) = 1 & \text{on } \Gamma_1 \cup \Gamma_2 \\ u(\boldsymbol{\mu}) = 0 & \text{on } \Gamma_3 \cup \Gamma_4 \cup \Gamma_5. \end{cases} \tag{2.68}$$

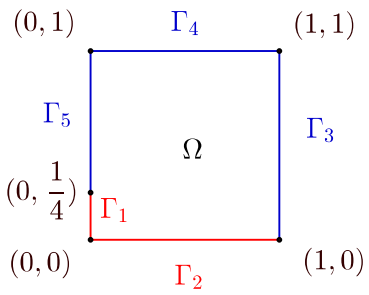


Figure 2.14: Geometry Square problem

Let us note that μ_1 represents the Péclet number of the advection-diffusion problem, while μ_2 is the angle between the x axis and the direction of the constant advection field. The bilinear form associated to the problem is:

$$a(u, v; \boldsymbol{\mu}) = \int_\Omega \frac{1}{\mu_1}\nabla u \cdot \nabla v + (\cos\mu_2\, \partial_x u + \sin\mu_2\, \partial_y u)v. \tag{2.69}$$

We introduce again a triangulation $\mathcal{T}_h$ on the domain Ω and we consider $\mathbb{P}^1(\mathcal{T}_h)$, that is the piecewise polynomial interpolation space of order 1. Now we can define our stabilization term:

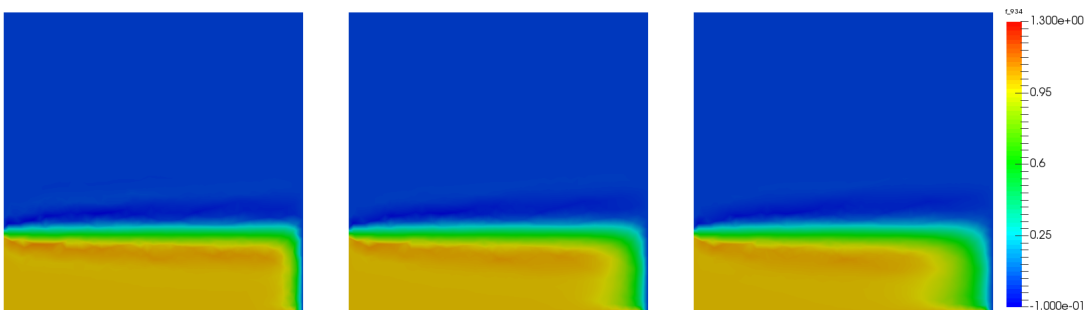
$$s(u_h, v_h; \boldsymbol{\mu}) = \sum_{K\in\mathcal{T}_h} \delta_K \int_K (\cos\mu_2, \sin\mu_2)\cdot\nabla u_h \ (\cos\mu_2, \sin\mu_2)\cdot\nabla v_h \tag{2.70}$$

in which the value of the weights δ_K is to be assigned.

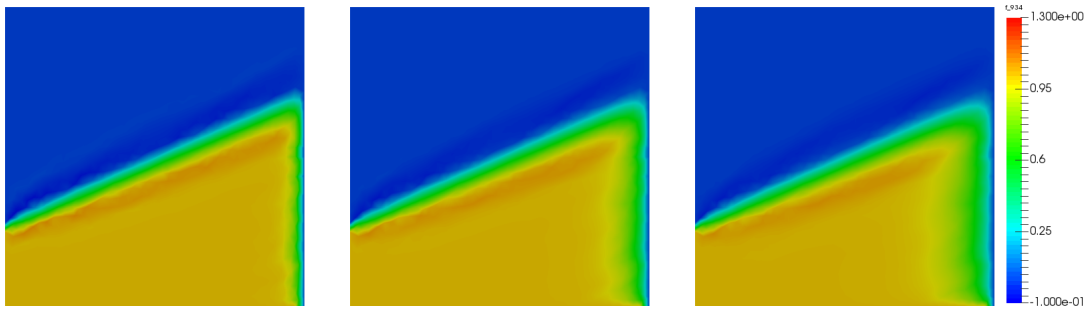
As we did in section 2.2.1 we define $l \in \mathbb{P}^1(\mathcal{T}_h)$ a lifting of the boundary conditions and then we can obtain our final FE approximation problem:

$$\begin{aligned} &\text{find } u^{s,\mathcal{N}}(\boldsymbol{\mu}) \in X^{\mathcal{N}} \text{ s.t.} \\ &a_{stab}(u^{s,\mathcal{N}}(\boldsymbol{\mu}), v^{\mathcal{N}}; \boldsymbol{\mu}) = F_{stab}(v^{\mathcal{N}}; \boldsymbol{\mu}) \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}}, \end{aligned} \tag{2.71}$$

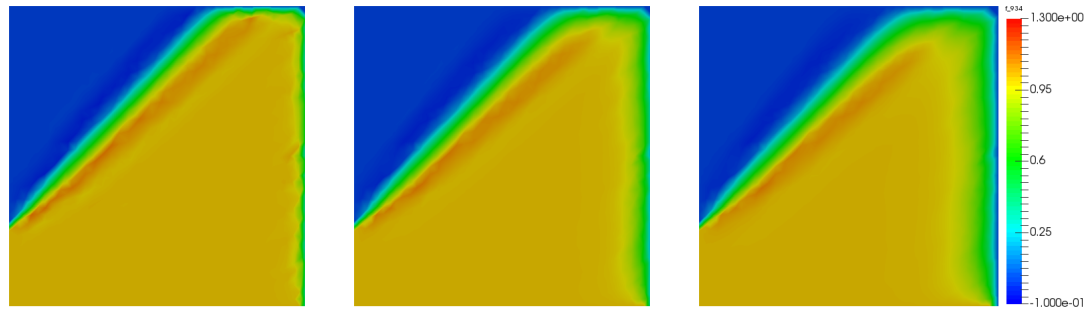
where $X^{\mathcal{N}}$, a_{stab} and F_{stab} are defined as in (2.65). Here, we tried to change δ_K for different choice of the parameter μ_2, with $\mu_1 = 2\cdot 10^4$. As we can see in figure 2.15, there are some simulations that are too stabilized and others that are not enough stabilized, and even in



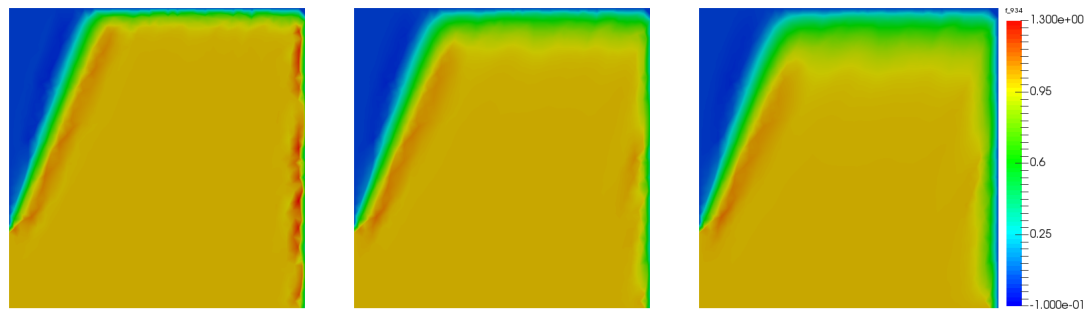
(a) $\mu_2 = 0, \delta_K = 0.7$ (b) $\mu_2 = 0, \delta_K = 1.4$ (c) $\mu_2 = 0, \delta_K = 2.1$



(d) $\mu_2 = 0.4, \delta_K = 0.7$ (e) $\mu_2 = 0.4, \delta_K = 1.4$ (f) $\mu_2 = 0.4, \delta_K = 2.1$



(g) $\mu_2 = 0.8, \delta_K = 0.7$ (h) $\mu_2 = 0.8, \delta_K = 1.4$ (i) $\mu_2 = 0.8, \delta_K = 2.1$



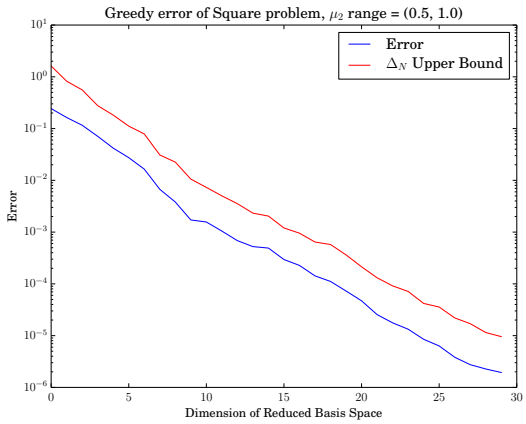
(j) $\mu_2 = 1.2, \delta_K = 0.7$ (k) $\mu_2 = 1.2, \delta_K = 1.4$ (l) $\mu_2 = 1.2, \delta_K = 2.1$

Figure 2.15: FE solution comparison varying δ_K and μ_2

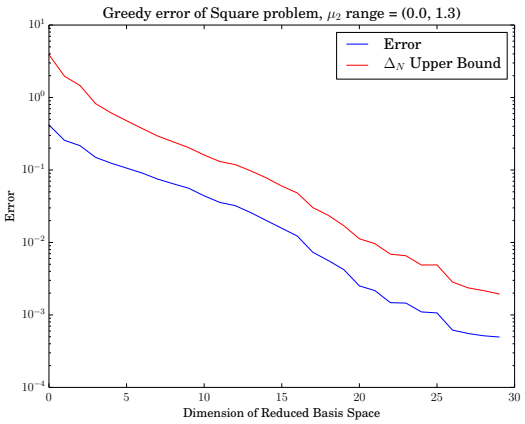
the same simulation we can find zones that are too much stabilized and zones that are not. For example, in case $\mu_2 = 1.2$ we can see in figure 2.15(j) that boundary layers near the right side and across the square are not enough stabilized, indeed physically, the solution should not get over 1, but we reached 1.3. If we add more stabilization ($\delta_K = 2.1$) as in figure 2.15(l) we can fix the problem on the right boundary layer, but not on the left boundary layer (still too perturbed).
Moreover, we have to consider the fact that we are working in a RB context and we can not choose δ_K varying depending on the parameter μ_2 or on the triangular $K \in \mathcal{T}_h$. If we look at picture 2.15(c) we are losing the exact solution, adding too much diffusivity, in particular in the right side of the solution. Overall, we have to find a compromise that will be $\delta_K = 1$ from now on.

Observing the greedy algorithm, we think that it is worth to be noted that if we reduce the mesh size, increasing then the number of the degrees of freedom $\mathcal{N}$, the number N of basis function, that we need to achieve the same value for the error bound, increases too. For example, we have tried with 2 different mesh with $\mathcal{N} = 4688$ and $\mathcal{N} = 1121$. For $\mathcal{D} = \{10^4\} \times [0.5, 1]$ we have that the former needs 15 basis functions to reach a tolerance of 10^{-7} while the latter needs only 12 basis function. Moreover, after 18 basis function the first can guarantee an error bound of $1.319 \cdot 10^{-7}$, while the second guarantees an error bound of $9.409 \cdot 10^{-9}$.
Our (heuristic) explanation of this phenomenon is that, by reducing the mesh size h, we are able to capture more information about the sharp layers, but this means that the *number* of possible configurations of the system, depending on the parameter, rises. As a consequence, we will need more basis functions to obtain the same accuracy. This behaviour of the stabilized method have been highlighted also in [9].

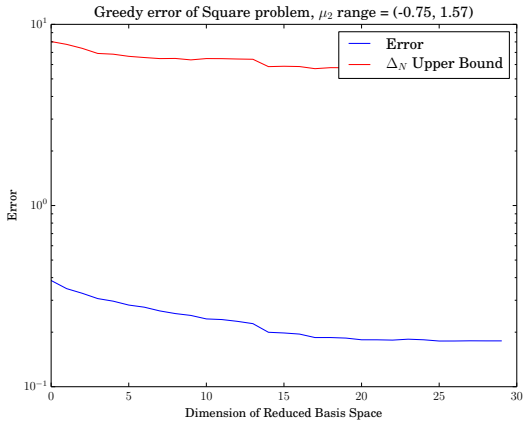
Another important aspect that we can observe is that, for different ranges of parameter $\boldsymbol{\mu}$, that variations of the advection field are more relevant than variations of the Péclet number. This is because by varying the direction of the advection field, the solution shows strong variations in energy norm [40]. Since this, we have tried to considerably change the range of the angle of the advection field and to study the behaviour of the greedy algorithm in these cases. As maximum range for μ_2 we have chosen $\left[-0.75 \approx -\frac{\pi}{4}, 1.57 \approx \frac{\pi}{2}\right]$, while μ_1 will always be in $\left[10^4, 10^5\right]$.
As we can see in figure 2.16, with a small range of μ_2, as $\left[0.5 \approx \frac{\pi}{6}, 1 \approx \frac{\pi}{3}\right]$, the greedy algorithm converges quickly, with a reduced basis space of dimension 30, we have an error around 10^{-6}, while, widening the range to our maximum range, the convergence is heavily slowed (the error decrease really slowly and after 30 steps of Greedy algorithm, the error is still over 0.1). Clearly, this range is excessive and this results are not acceptable, but are useful to optimize our range, to get to $[0, 1.57]$, where we have all the possible positive angles and a good decrease of the error.



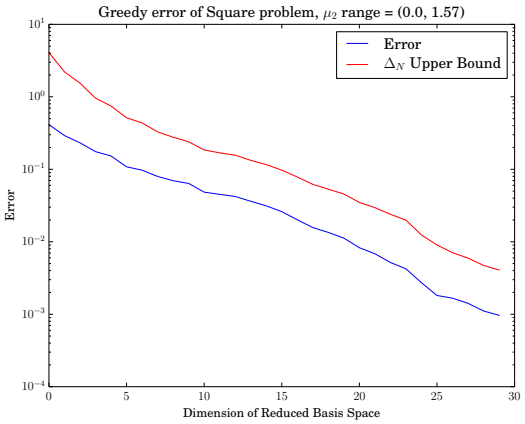
(a) $\mu_2 \in (0.5, 1)$



(b) $\mu_2 \in (0, 1.3)$



(c) $\mu_2 \in (-0.75, 1.57)$



(d) $\mu_2 \in (0, 1.57)$

Figure 2.16: RB error and Δ_N error bound varying μ_2 range

Chapter 3

Stabilized reduced basis method for parabolic equations

In this chapter, we will apply a stabilized RB method to parabolic (time dependent) problems. This approach is similar to one introduced for steady problems of chapter 2. The RB method for time dependent problems has been already studied in several works, e.g. [14, 19, 47, 52]. Stabilization of advection diffusion parabolic equations with high Péclet number have been studied in several works with different stabilization methods. We will adapt SUPG stabilization for FE methods on parabolic equations to RB method, as suggested in [40, 41, 42, 43]. Moreover, we will propose two different algorithms to compute the Greedy procedure [18, 47] of the RB method and we will compare them.

As the previous chapter, this one is a prerequisite to chapter 4, where we will deal with stochastic equations with random input parameters, defined by a prescribed random variable.

3.1 Reduced basis methods for linear parabolic equations

Like for elliptic equations, we define a *parameter domain* $\mathcal{D}$ as a closed subset of $\mathbb{R}^P$ and we call $\boldsymbol{\mu}$ any general P-tuple belonging to $\mathcal{D}$. Again, let Ω be a bounded open subset of $\mathbb{R}^d\,(d = 1, 2, 3)$ with regular boundary $\partial\Omega$ and let X be a functional space such that $H_0^1(\Omega) \subset X \subset H^1(\Omega)$. For each admissible value of the parameter, i.e. for each $\boldsymbol{\mu} \in \mathcal{D}$, we define the continuous bilinear forms

$$\begin{aligned} &a(\cdot,\cdot;\boldsymbol{\mu}) : X \times X \to \mathbb{R} \\ &m(\cdot,\cdot;\boldsymbol{\mu}) : X \times X \to \mathbb{R}. \end{aligned} \tag{3.1}$$

We suppose that the form a satisfies the *affinity* and coercivity assumptions (1.9) and (1.7), respectively. We need m to be coercive, i.e.

$$\exists\, \alpha_m \text{ s.t. } \alpha_m \leq \alpha_m(\boldsymbol{\mu}) := \inf_{v \in X} \frac{m(v, v; \boldsymbol{\mu})}{||v||_X^2} \quad \forall \boldsymbol{\mu} \in \mathcal{D} \tag{3.2}$$

and we assume also that the *mass* form m satisfy the *affinity* assumption

$$m(v, w; \boldsymbol{\mu}) = \sum_{q=1}^{Q_m} \Theta_q^m(\boldsymbol{\mu}) m^q(v, w) \tag{3.3}$$

where, as in (1.9), $\Theta_q^m : \mathcal{D} \to \mathbb{R}$, $q = 1, \ldots, Q_m$, are smooth functions whereas $m^q : X \times X \to \mathbb{R}$, $q = 1, \ldots, Q_m$, are continuous $\boldsymbol{\mu}$-independent bilinear forms. Finally, for each $\boldsymbol{\mu} \in \mathcal{D}$, we define the right-hand side continuous form $F(\cdot; \boldsymbol{\mu}) : X \to \mathbb{R}$ which satisfies the *affine* assumption (1.10). Let us finally denote our time domain with $I = [0, T]$, where T is the final time.

We want to spend some words to explain why we need the affine assumption (3.3) also on the mass term. As we saw in chapter 1, the parameter can be geometrical, that is the original domain of the problem $\Omega_p(\boldsymbol{\mu})$ might depend on the parameter. As in section 1.1.1, let us then suppose that both the original domain and the reference one are divided in subdomains, like in (1.17) and (1.18). The original *mass* form, that is the L^2 scalar product on the original domain, becomes:

$$m_p(u_p, v_p; \boldsymbol{\mu}) = \sum_{l=1}^{L_{dom}} \int_{\Omega_p^l(\boldsymbol{\mu})} u_p v_p dx \quad \forall u_p, v_p \in X_p(\boldsymbol{\mu}) \tag{3.4}$$

where $X_p(\boldsymbol{\mu})$ is the original test function space. Tracking back the latter integrals on the reference domain Ω trough the map T defined in (1.20) and (1.21), as we did in section 1.1.2 for the bilinear form a_p, we obtain:

$$m(v, w; \boldsymbol{\mu}) = \sum_{l=1}^{L_{dom}} \int_{\Omega^l} uv J^l(\boldsymbol{\mu}) dx \quad \forall u, v \in X \tag{3.5}$$

where $J^l(\boldsymbol{\mu})$ is the (local) Jacobian of the transformation T.

We can now define our continuous problem:

$$\begin{aligned} &\text{find } u(\cdot; \boldsymbol{\mu}) \in C^0(I; L^2(\Omega)) \cap L^2(I; X) \text{ s.t.} \\ &m(\partial_t u(t; \boldsymbol{\mu}), v) + a(u(t; \boldsymbol{\mu}), v; \boldsymbol{\mu}) = g(t) F(v; \boldsymbol{\mu}) \quad \forall v \in X, \quad \forall t \in I \\ &\text{given the initial value } u(0; \boldsymbol{\mu}) = u_0 \in L^2(\Omega) \end{aligned} \tag{3.6}$$

where $g : I \to \mathbb{R}$ is a *control function* such that $g \in L^2(I)$. We need such a *control function* for problems of the form

$$\begin{cases} \partial_t u(\boldsymbol{\mu}) + Lu(\boldsymbol{\mu}) = h_1(\cdot, t) & \text{in } \Omega \\ u(\cdot, t\ \boldsymbol{\mu}) = h_2(\cdot, t) & \text{on } \partial\Omega, \ \forall t \in I \\ u(\cdot, 0; \boldsymbol{\mu}) = u_0(\cdot, t) \end{cases} \tag{3.7}$$

where L is a differential operator and $h_1 \in (\Omega \times I)$ and $h_2 : \partial\Omega \times I \to \mathbb{R}$ are sufficiently regular. If we suppose that $h_1(x, t) = g_1(t) f_1(x)$ and $h_2(x, t) = g_2(t) f_2(x)$ with $g_{1,2} \in L^2(I)$, $f_1 \in L^2(\Omega)$ and $f_2 \in H^{\frac{1}{2}}(\partial\Omega)$, we can obtain a weak formulation like (3.6) adding a right-hand side term that satisfies lifting boundary conditions.

3.1.1 Discretization and RB formulation

To discretize the time-dependent problem (3.6) we follow the approach used in [16, 19, 39, 47], that is to use finite differences in time and FE in space discretization [48].

We start by discretizing the spatial part of the problem. We thus define the FE *truth* approximation space $X^{\mathcal{N}}$ and we denote its basis with $\{\phi_i\}_{i=1}^{\mathcal{N}}$. The *semi-discretized* problem reads as

$$\begin{aligned}
&\text{for each } t \in I, \text{ find } u^{\mathcal{N}}(t;\boldsymbol{\mu}) \in X^{\mathcal{N}} \text{ s.t.} \\
&m(\partial_t u^{\mathcal{N}}(t;\boldsymbol{\mu}), v^{\mathcal{N}};\boldsymbol{\mu}) + a(u^{\mathcal{N}}(t;\boldsymbol{\mu}), v^{\mathcal{N}};\boldsymbol{\mu}) = g(t)F(v^{\mathcal{N}};\boldsymbol{\mu}) \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}}, \quad \forall t \in I \\
&\text{given the initial value } u_0^{\mathcal{N}} \text{ s.t.} \\
&(u_0^{\mathcal{N}}, v^{\mathcal{N}})_{L^2(\Omega)} = (u_0, v^{\mathcal{N}})_{L^2(\Omega)} \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}}.
\end{aligned} \tag{3.8}$$

To obtain a fully discretized problem, we subdivide the time interval I into J subintervals of length $\Delta t = \frac{T}{J}$ and we define $t_j = j\Delta t$, $j = 1, \dots, J$. We then replace the time derivative in (3.8) with a backward finite difference approximation. The fully discretized problem we are considering is:

$$\begin{aligned}
&\text{for each } 1 \le j \le J, \text{ find } u_j^{\mathcal{N}}(\boldsymbol{\mu}) \in X^{\mathcal{N}} \text{ s.t.} \\
&\frac{1}{\Delta t} m(u_j^{\mathcal{N}}(\boldsymbol{\mu}) - u_{j-1}^{\mathcal{N}}(\boldsymbol{\mu}), v^{\mathcal{N}};\boldsymbol{\mu}) + a(u_j^{\mathcal{N}}(t;\boldsymbol{\mu}), v^{\mathcal{N}};\boldsymbol{\mu}) = g(t_j)F(v^{\mathcal{N}};\boldsymbol{\mu}) \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}}, \\
&\text{given the initial condition } u_0^{\mathcal{N}} \text{ s.t.} \\
&(u_0^{\mathcal{N}}, v^{\mathcal{N}})_{L^2(\Omega)} = (u_0, v^{\mathcal{N}})_{L^2(\Omega)} \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}}.
\end{aligned} \tag{3.9}$$

The latter problem is the *Backward Euler-Galerkin* discretization of (3.6). Of course, this is not the only way to discretize the time-dependent problem (3.6), for example we can resort to other theta-methods (e.g. Crank-Nicholson) or to high order method (e.g. Runge–Kutta) [48].
We will denote with $\boldsymbol{u}^{\mathcal{N}}(\boldsymbol{\mu})$ the solution array, that is:

$$\boldsymbol{u}^{\mathcal{N}}(\boldsymbol{\mu}) = (u_1^{\mathcal{N}}(\boldsymbol{\mu}), \dots, u_J^{\mathcal{N}}(\boldsymbol{\mu})) \in \left(X^{\mathcal{N}}\right)^J. \tag{3.10}$$

The RB formulation of the problem (3.9) is based on hierarchical RB space as we did for the steady case, that is: given an integer N_{max} we define a finite sequence $\{X_N^{\mathcal{N}}\}_{N=1}^{N_{max}}$ of subspaces of $X^{\mathcal{N}}$ such that (1.31) holds. To generate this subspaces there are different techniques that we will study in section 3.1.2, for the moment let us say that the basis functions of $X_N^{\mathcal{N}}$ are built by properly combining *snapshots* in time and space. As in chapter 1 we use the following notation

$$X_N^{\mathcal{N}} = \text{span}\{\zeta_n^{\mathcal{N}} | 1 \le n \le N\}. \tag{3.11}$$

We also assume that the functions $\zeta_n^{\mathcal{N}}$ are mutually orthonormal with respect to the scalar product $(\cdot,\cdot)_X$.

The RB problem is then:

$$\begin{aligned}&\text{for each } 1 \le j \le J, \text{ find } u_{N,j}^{\mathcal{N}}(\boldsymbol{\mu}) \in X_N^{\mathcal{N}} \text{ s.t.}\\ &\frac{1}{\Delta t} m(u_{N,j}^{\mathcal{N}}(\boldsymbol{\mu}) - u_{N,j-1}^{\mathcal{N}}(\boldsymbol{\mu}), v_N; \boldsymbol{\mu}) + a(u_{N,j}^{\mathcal{N}}(t;\boldsymbol{\mu}), v_N; \boldsymbol{\mu}) = g(t_j) F(v_N; \boldsymbol{\mu}) \quad \forall v_N \in X_N^{\mathcal{N}},\\ &\text{given the initial condition } u_{N,0}^{\mathcal{N}} \text{ s.t.}\\ &(u_{N,0}^{\mathcal{N}}, v_N)_{L^2(\Omega)} = (u_0^{\mathcal{N}}, v_N)_{L^2(\Omega)} \quad \forall v_N \in X_N^{\mathcal{N}}.\end{aligned} \tag{3.12}$$

Again, as in (3.10), we define

$$\boldsymbol{u}_N^{\mathcal{N}}(\boldsymbol{\mu}) = (u_{N,1}^{\mathcal{N}}(\boldsymbol{\mu}), \dots, u_{N,J}^{\mathcal{N}}(\boldsymbol{\mu})) \in \left(X_N^{\mathcal{N}}\right)^J. \tag{3.13}$$

Let us now try to obtain the matrix formulation of the RB problem (3.12). First of all we recall that, for each $j = 1, \dots, J$, the RB solution $u_{N,j}^{\mathcal{N}}(\boldsymbol{\mu}) \in X_N^{\mathcal{N}}$ can be expressed as:

$$u_{N,j}^{\mathcal{N}}(\boldsymbol{\mu}) = \sum_{m=1}^{N} u_{N,j,m}^{\mathcal{N}}(\boldsymbol{\mu}) \zeta_m^{\mathcal{N}}. \tag{3.14}$$

Then, by taking $v_N = \zeta_n^{\mathcal{N}}$, $\forall n = 1, \dots, N$, in the RB formulation (3.12) we have:

$$\frac{1}{\Delta t} m(u_{N,j}^{\mathcal{N}}(\boldsymbol{\mu}), \zeta_n^{\mathcal{N}}; \boldsymbol{\mu}) + a(u_{N,j}^{\mathcal{N}}(t;\boldsymbol{\mu}), \zeta_n^{\mathcal{N}}; \boldsymbol{\mu}) = g(t_j) F(\zeta_n^{\mathcal{N}}; \boldsymbol{\mu}) + \frac{1}{\Delta t} m(u_{N,j-1}^{\mathcal{N}}(\boldsymbol{\mu}), \zeta_n^{\mathcal{N}}; \boldsymbol{\mu}) \tag{3.15}$$

that is, recalling the affine assumptions (1.9), (1.10) and (3.3):

$$\begin{aligned}&\sum_{m=1}^{N} \left(\frac{1}{\Delta t} \sum_{q=1}^{Q_m} \Theta_m^q(\boldsymbol{\mu}) m^q(\zeta_m^{\mathcal{N}}, \zeta_n^{\mathcal{N}}) + \sum_{q=1}^{Q_a} \Theta_a^q(\boldsymbol{\mu}) a^q(\zeta_m^{\mathcal{N}}, \zeta_n^{\mathcal{N}}) \right) u_{N,j}^{\mathcal{N}}(\boldsymbol{\mu}) =\\ &= g(t_j) \sum_{q=1}^{Q_F} \Theta_F^q(\boldsymbol{\mu}) F^q(\zeta_n^{\mathcal{N}}) + \sum_{m=1}^{N} \left(\frac{1}{\Delta t} \sum_{q=1}^{Q_m} \Theta_m^q(\boldsymbol{\mu}) m^q(\zeta_m^{\mathcal{N}}, \zeta_n^{\mathcal{N}}) \right) u_{N,j-1}^{\mathcal{N}}(\boldsymbol{\mu}).\end{aligned} \tag{3.16}$$

We can thus obtain the matrix formulation:

$$\begin{aligned}&\left(\frac{1}{\Delta t} \sum_{q=1}^{Q_m} \Theta_m^q(\boldsymbol{\mu}) \mathbf{M}_N^q + \sum_{q=1}^{Q_a} \Theta_a^q(\boldsymbol{\mu}) \mathbf{A}_N^q \right) \mathbf{u}_{N,j}^{\mathcal{N}}(\boldsymbol{\mu}) =\\ &= g(t_j) \sum_{q=1}^{Q_F} \Theta_F^q(\boldsymbol{\mu}) \mathbf{F}_N^q + \left(\frac{1}{\Delta t} \sum_{q=1}^{Q_m} \Theta_m^q(\boldsymbol{\mu}) \mathbf{M}_N^q \right) \mathbf{u}_{N,j-1}^{\mathcal{N}}(\boldsymbol{\mu})\end{aligned} \tag{3.17}$$

where $\mathbf{A}_N, \mathbf{F}_N$ are defined in (1.42) e (1.40), while

$$\left(\mathbf{u}_{N,j}^{\mathcal{N}}(\boldsymbol{\mu})\right)_m = u_{N,m,j}^{\mathcal{N}}(\boldsymbol{\mu}), \quad \left(\mathbf{M}_N^q\right)_{nm} = m^q(\zeta_n^{\mathcal{N}}, \zeta_m^{\mathcal{N}}), \tag{3.18}$$

for $n, m = 1, \dots, N$, and $j = 1, \dots, J$. Denoting with $\mathcal{Z}$ the $\mathcal{N} \times N$ matrix whose columns are the coordinates of the reduced basis $\zeta_1^{\mathcal{N}}, \dots, \zeta_N^{\mathcal{N}}$ with respect to $\{\phi_i\}_{i=1}^{\mathcal{N}}$, it holds that:

$$\begin{aligned}&\mathbf{M}_N^q = \mathcal{Z}^T \mathbf{M}_{\mathcal{N}}^q \mathcal{Z} \quad 1 \le q \le Q_m\\ &\text{with } (\mathbf{M}_{\mathcal{N}}^q)_{ij} = m^q(\phi_i, \phi_j) \quad 1 \le i, j \le \mathcal{N}, \quad 1 \le q \le Q_m.\end{aligned} \tag{3.19}$$

Like in the steady case, during the *Offline* stage, we have to compute and store the FE matrices, the *snapshots* solutions and the RB matrices. The only difference between the time-dependent case and the steady case is that in the former we have also to deal with matrices associated to the mass term, which arise as a consequence of the time-dependency.

The Online operation count is the following [19, 39, 47]:

- $O((Q_a + Q_m)N^2)$ to get the left-hand side matrix;
- $O(Q_F N + Q_m N^2)$ to get the right-hand side;
- $O(N^3 + JN^2)$ to perform a factorization of the left-hand side matrix and to solve the J linear systems (3.17);
- $O(JN)$ to perform the scalar products (3.14).

Once again, we stress the point that the *Online* computational cost is independent of the dimension $\mathcal{N}$ of the underlying FE element *truth* approximation.

3.1.2 Sampling strategies and *a posteriori* error estimates

To construct the reduced basis in the time-dependent case, we will try two different methods. The first will be the so called *POD–Greedy* approach [19, 39, 47], while the second will be a mix between the first one and the one proposed in [18] that we will call *$GS^{\perp}$–POD–Greedy*.

In both algorithms we need to introduced the POD (proper orthogonal decomposition) technique (which is also called PCA (principal component analysis) in machine learning environment)[30, 31].

POD

POD algorithm starts from K vectors w_k, $k = 1, \dots, K$, in a linear space W (we will call $W_K := \text{span}\{w_k | 1 \leq k \leq K\}$), and it returns M vectors χ_m, $m = 1, \dots, M$, with $M < K$, that are orthonormal with respect to a given scalar product $(\cdot, \cdot)$ and such that the space

$$\mathcal{P}_M = \text{span}\{\chi_m | 1 \leq m \leq M\} \tag{3.20}$$

is optimal in the sense that:

$$\mathcal{P}_M = \underset{Y_M \subset W_K}{\arg\min} \left(\frac{1}{K} \sum_{k=1}^{K} \inf_{v \in Y_M} ||w_k - v||^2 \right), \tag{3.21}$$

where Y_M denotes an M-dimensional linear space and $|| \cdot ||$ is the norm induced by scalar product $(\cdot, \cdot)$. We also note that it also holds:

$$\inf_{v \in Y_M} ||w_k - y||^2 = ||w_k - \pi_{Y_M} v||^2 \tag{3.22}$$

where π_{Y_M} is the orthogonal projection on Y_M, with respect to the scalar product $(\cdot, \cdot)$. Roughly speaking we are searching the M-dimensional linear subspace which minimize

the sum of all the "errors" between our vectors w_k and their projections onto Y_M. It is easy to prove that it is equivalent to find the M-dimensional linear subspace included in W_K which maximize the variance of vectors χ_m, $m = 1, \dots, M$.

We show now an effective procedure to compute the orthonormal basis $\{\chi_m | 1 \leq m \leq M\}$ [30, 31].

1. Given snapshots $\{w_k | 1 \leq k \leq K\}$, we compute the $K \times K$ symmetric and positive definite matrix C defined by
$$C_{ij} = \frac{1}{K}(w_i, w_j). \tag{3.23}$$
2. We compute the first M eigenvalues λ_m, $1 \leq m \leq M$, of C and the associated eigenvectors $\{\psi, 1 \leq m \leq M\}$.
3. We obtain the orthonormal basis $\{\chi_m | 1 \leq m \leq M\}$ using the formula:
$$\chi_m = \frac{1}{\sqrt{\lambda_m}} \sum_{k=1}^{K} (\psi_m)_k w_k \quad 1 \leq m \leq M, \tag{3.24}$$
where $(\psi_m)_k$ is the k-th component of the eigenvector ψ_m.

As regards the error, we can define

$$\mathcal{E}_M := \frac{1}{K} \sum_{k=1}^{K} ||w_k - \pi_{\mathcal{P}_M} w_k||^2 = \sum_{k=M+1}^{K} \lambda_k \tag{3.25}$$

where the second equality can be proven as in [30].
It was already obvious from the definition that $\mathcal{E}_M \to 0$ as $M \to K$, but the equality (3.25) turns out to be useful for the choice of M. If we fix a tolerance ε_{tol}^{POD}, we can set M as:

$$M = \min\left\{ R \,\middle|\, \left(\sum_{k=R+1}^{K} \lambda_k \right)^{1/2} \leq \varepsilon_{tol}^{POD} \right\}. \tag{3.26}$$

To indicate the POD procedure, we adopt the following compact notation:

$$\{\chi_m | 1 \leq m \leq M\} = POD(W_K, M). \tag{3.27}$$

POD–Greedy method

We introduce now the POD–Greedy method [18, 19, 47], used to build the reduced basis for the time dependent problem (3.12). First of all, let us define the norm:

$$|||\boldsymbol{v}^{\mathcal{N}}(\boldsymbol{\mu})|||_{t-dep} = \left(m(v_J^{\mathcal{N}}(\boldsymbol{\mu}), v_J^{\mathcal{N}}(\boldsymbol{\mu}); \boldsymbol{\mu}) + \sum_{j=1}^{J} a^{sym}(v_j^{\mathcal{N}}(\boldsymbol{\mu}), v_j^{\mathcal{N}}(\boldsymbol{\mu}); \boldsymbol{\mu}) \Delta t \right)^{\frac{1}{2}} \tag{3.28}$$

for all sequences $\boldsymbol{v}^{\mathcal{N}}(\boldsymbol{\mu}) = (v_1^{\mathcal{N}}(\boldsymbol{\mu}), \dots, v_J^{\mathcal{N}}(\boldsymbol{\mu})) \in (X^{\mathcal{N}})^J$.
Let us denote with $\boldsymbol{e}(\boldsymbol{\mu})$, $\boldsymbol{\mu} \in \mathcal{D}$, the difference between the *truth* solution $\boldsymbol{u}^{\mathcal{N}}(\boldsymbol{\mu})$ and the

RB one $\boldsymbol{u}_N^{\mathcal{N}}(\boldsymbol{\mu})$. In order to pursue an effective greedy strategy, we assume that we have a sharp and a computationally inexpensive *a posteriori* error estimator $\boldsymbol{\mu} \mapsto \Delta_N^t(\boldsymbol{\mu})$, as in appendix A, such that:

$$|||\boldsymbol{e}(\boldsymbol{\mu})|||_{t-dep} \leq \Delta_N^t(\boldsymbol{\mu}) \quad \forall \boldsymbol{\mu} \in \mathcal{D}, \quad 1 \leq N \leq N_{max}. \tag{3.29}$$

Like in the steady case, we define a finite subset Ξ_{train} of $\mathcal{D}$, large enough to be considered an approximation of the parameter space $\mathcal{D}$.

The N-th step of the *POD–Greedy* algorithm can be described as:

1. find the value $\tilde{\boldsymbol{\mu}} \in \Xi_{train}$ that maximize the estimator Δ_{N-1}^t;
2. compute the FE solution to problem (3.9) for $\tilde{\boldsymbol{\mu}}$;
3. apply a first POD method to time snapshots of this solution, i.e., $\{u_j^{\mathcal{N}}(\tilde{\boldsymbol{\mu}})\}_{j=1}^J$ and obtain $\{\chi_m\}_{m=1}^{M_1}$, for some prescribed M_1, that represents the number of retained modes;
4. apply a second POD method to old reduced basis together with the results of the first POD, i.e. $\{\chi_m\}_{m=1}^{M_1} \cup \{\zeta_n^{\mathcal{N}}\}_{n=1}^{N-1}$ and retain only the first M_2 modes.

Data: $\boldsymbol{\mu}^1$, N_{max}, ε_{tol}^*, M_1, ε_{tol}^1, M_2, ε_{tol}^2
Result: A reduced space $\mathcal{Z}$
$\tilde{\boldsymbol{\mu}} = \boldsymbol{\mu}^1$;
compute $\boldsymbol{u}^{\mathcal{N}}(\tilde{\boldsymbol{\mu}})$;
$\mathcal{Z} = POD(\{u_j^{\mathcal{N}}(\tilde{\boldsymbol{\mu}})\}_{j=1}^J, M_1, \varepsilon_{tol}^1)$;
$N = M_1$;
compute $\Delta_1^t(\boldsymbol{\mu})$, $\forall \boldsymbol{\mu} \in \Xi_{train}$;
set $\tilde{\boldsymbol{\mu}} = \arg\max\{\Delta_N^t(\boldsymbol{\mu}),\ \boldsymbol{\mu} \in \Xi_{train}\}$;
while $\Delta_N^t(\tilde{\boldsymbol{\mu}}) > \varepsilon_{tol}^*$ *and* $N < N_{max}$ **do**
  $\{\chi_m\} = POD(\{u_j^{\mathcal{N}}(\tilde{\boldsymbol{\mu}})\}_{j=1}^J, M_1, \varepsilon_{tol}^1)$;
  $\mathcal{Z} = POD(\{\chi_m\} \cup \mathcal{Z}, M_2, \varepsilon_{tol}^2)$;
  $N =$dimension of $\mathcal{Z}$;
  compute $\Delta_n^t(\boldsymbol{\mu})$, $\forall \boldsymbol{\mu} \in \Xi_{train}$;
  set $\tilde{\boldsymbol{\mu}} = \arg\max\{\Delta_N^t(\boldsymbol{\mu}),\ \boldsymbol{\mu} \in \Xi_{train}\}$;
  compute $\boldsymbol{u}^{\mathcal{N}}(\tilde{\boldsymbol{\mu}})$;
end

Algorithm 3: POD–Greedy

Concerning stopping criteria, we will consider 2 different conditions: a tolerance on the greedy, i.e. we stop the algorithm when $\Delta_N^t(\boldsymbol{\mu}) \leq \varepsilon_{tol}^*$, $\forall \boldsymbol{\mu} \in \Xi_{train}$; a maximum RB dimension N_{max}. The POD–Greedy procedure is shown in algorithm 3. In this algorithm there are two "tuning" values, M_1 and M_2. Moreover, thanks to (3.26), we can use two tolerances ε_{tol}^1, ε_{tol}^2 as stopping criteria of the two PODs. So we will have 4 parameter to set.

Several problems can arise from this choice, for example, if we take an M_1 too small, it can happen that the choices of the parameter by Greedy algorithm is repetitive and that a single parameter is chosen many times. If we take a too large M_2 we may keep more

information than what we need. If ε^2_{tol} is too small, the algorithm can fall in a infinite loop before reaching N_{max}.

For our computation we have chosen $\varepsilon^1_{tol} = 10^{-3}$ and $M_1 \in \{5, 6, 7, 8\}$, while $\varepsilon^2_{tol} = 0$ and $M_2 \in \{1, 2, 3\}$.

GS$^\perp$–POD–Greedy method

To avoid this choice, we can remove some parameter choices changing the algorithm to the one that we will call *GS$^\perp$–POD–Greedy* (Gram-Schmidt, orthogonalization, POD, Greedy)[18]. The main problem of the *POD–Greedy* algorithm is the possibility of an infinite loop, due to the fact that the first POD can provide to the reduced basis space the same information from the same parameter $\tilde{\boldsymbol{\mu}}$ at each step. The main idea of this improvement is to delete the information we already have from reduced basis and then doing the POD on the chosen parameter. It means that instead of doing a POD on $\boldsymbol{u}^{\mathcal{N}}(\tilde{\boldsymbol{\mu}})$, we will do it on the projection of this vectors onto the orthogonal space $\mathcal{Z}^\perp$. Computationally it is not a great effort, it is sufficient to perform $\boldsymbol{u}^{\perp,\mathcal{N}} := \boldsymbol{u}^{\mathcal{N}} - \mathcal{Z}\mathcal{Z}^T\boldsymbol{u}^{\mathcal{N}}$. After this we will add the result of this POD to the old reduced basis space, and the we perform a Gram-Schmidt orthonormalization to stabilize the further steps, as we can see in algorithm 4.

Data: $\boldsymbol{\mu}^1$, N_{max}, ε^*_{tol}, M_1, ε^1_{tol}
Result: A reduced space $\mathcal{Z}$
$\tilde{\boldsymbol{\mu}} = \boldsymbol{\mu}^1$;
compute $\boldsymbol{u}^{\mathcal{N}}(\tilde{\boldsymbol{\mu}})$;
$\mathcal{Z} = POD(\{u^{\mathcal{N}}_j(\tilde{\boldsymbol{\mu}})\}^J_{j=1}, M_1, \varepsilon^1_{tol})$;
$N = M_1$;
compute $\Delta^t_1(\boldsymbol{\mu})$, $\forall \boldsymbol{\mu} \in \Xi_{train}$;
set $\tilde{\boldsymbol{\mu}} = \arg\max\{\Delta^t_N(\boldsymbol{\mu}),\ \boldsymbol{\mu} \in \Xi_{train}\}$;
while $\Delta^t_N(\tilde{\boldsymbol{\mu}}) > \varepsilon^*_{tol}$ *and* $N < N_{max}$ **do**
  compute $\boldsymbol{u}^{\perp,\mathcal{N}} = \boldsymbol{u}^{\mathcal{N}} - \mathcal{Z}\mathcal{Z}^T\boldsymbol{u}^{\mathcal{N}}$;
  $\{\chi_m\} = POD(\{u^{\perp,\mathcal{N}}_j(\tilde{\boldsymbol{\mu}})\}^J_{j=1}, M_1, \varepsilon^1_{tol})$;
  $\mathcal{Z} = GS(\mathcal{Z} \bigoplus \{\chi_m\})$;
  $N =$dimension of $\mathcal{Z}$;
  compute $\Delta^t_n(\boldsymbol{\mu})$, $\forall \boldsymbol{\mu} \in \Xi_{train}$;
  set $\tilde{\boldsymbol{\mu}} = \arg\max\{\Delta^t_N(\boldsymbol{\mu}),\ \boldsymbol{\mu} \in \Xi_{train}\}$;
  compute $\boldsymbol{u}^{\mathcal{N}}(\tilde{\boldsymbol{\mu}})$;
end

Algorithm 4: GS$^\perp$–POD–Greedy

Also in this algorithm we have to set M_1 and ε^1_{tol}, but we are sure that from every POD we will not take any redundant information. So, even if we fail the choice of parameter, the algorithm should work. Problems that we can have due to this setting are that the algorithm may be slow, or not the best to choose the RB space. Anyway, we will see that this algorithm is better than *POD–Greedy* algorithm, even in simulations.

For computations of the error estimator $\Delta_N^t(\boldsymbol{\mu})$ we refer to appendix A.

To test these two algorithms on our parabolic advection–diffusion problems, we have to resort a stabilization technique for our equations. Otherwise, we will get some really unstable results as in chapter 2. To avoid these problems we have to introduce a stabilization method valid for parabolic equations.

3.2 SUPG stabilization method for parabolic problems

In this section we briefly introduce the SUPG method for time-dependent problems [5, 29]. The idea is the same of the steady case: we have to add terms like $s^{(0)}$ and $\phi^{(0)}$, defined in (2.19) and (2.20) to the left-hand side and to the right-hand side of (3.9) respectively. More precisely, the right-hand side term is the same, whereas we have to slightly redefine the term $s^{(\rho)}$ in order to consider the time dependence and to guarantee the strong consistency. We thus set

$$s^{(\rho)}(v^{\mathcal{N}}(t), w^{\mathcal{N}}) = \sum_{K\in\mathcal{T}_h} \delta_K \left(\partial_t v^{\mathcal{N}}(t) + L v^{\mathcal{N}}(t), \frac{h_k}{|\boldsymbol{\beta}|}(L_{SS} + \rho L_S) w^{\mathcal{N}} \right)_K \tag{3.30}$$

where $v^{\mathcal{N}}(t) \in X^{\mathcal{N}}$ for each $t \in I$ and $w^{\mathcal{N}} \in X^{\mathcal{N}}$. To get the SUPG stabilization we take $\rho = 0$.

We note that if either the coefficients of the equation or its domain are $\boldsymbol{\mu}$-dependent, then the stabilization terms will depend on $\boldsymbol{\mu}$ too, as we have actually shown in section 2.2. Assuming the parametric dependence, we can write the *Backward Euler–SUPG* formulation as follows:

$$\begin{aligned}
&\text{for each } 1 \le j \le J, \text{ find } u_j^{\mathcal{N}}(\boldsymbol{\mu}) \in X^{\mathcal{N}} \text{ s.t.} \\
&\frac{1}{\Delta t} m_{stab}(u_j^{\mathcal{N}}(\boldsymbol{\mu}) - u_{j-1}^{\mathcal{N}}(\boldsymbol{\mu}), v^{\mathcal{N}}; \boldsymbol{\mu}) + a_{stab}(u_j^{\mathcal{N}}(t;\boldsymbol{\mu}), v^{\mathcal{N}}; \boldsymbol{\mu}) = \\
&\qquad = g(t_j) F_{stab}(v^{\mathcal{N}}; \boldsymbol{\mu}) \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}}, \\
&\text{given the initial condition } u_0^{\mathcal{N}} \text{ s.t.} \\
&(u_0^{\mathcal{N}}, v^{\mathcal{N}})_{L^2(\Omega)} = (u_0, v^{\mathcal{N}})_{L^2(\Omega)} \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}}.
\end{aligned} \tag{3.31}$$

where the stabilized terms are

$$\begin{aligned}
m_{stab}(v^{\mathcal{N}}, w^{\mathcal{N}}; \boldsymbol{\mu}) &= m(v^{\mathcal{N}}, w^{\mathcal{N}}; \boldsymbol{\mu}) + \sum_{K_p(\boldsymbol{\mu})\in\mathcal{T}_{h,p}(\boldsymbol{\mu})} \delta_{K_p(\boldsymbol{\mu})} \left(v^{\mathcal{N}}, \frac{h_{K_p(\boldsymbol{\mu})}}{|\boldsymbol{\beta}(\boldsymbol{\mu})|} L_{SS} w^{\mathcal{N}} \right)_{K_p(\boldsymbol{\mu})} \\
a_{stab}(v^{\mathcal{N}}, w^{\mathcal{N}}; \boldsymbol{\mu}) &= a(v^{\mathcal{N}}, w^{\mathcal{N}}; \boldsymbol{\mu}) + \sum_{K_p(\boldsymbol{\mu})\in\mathcal{T}_{h,p}(\boldsymbol{\mu})} \delta_{K_p(\boldsymbol{\mu})} \left(L v^{\mathcal{N}}, \frac{h_{K_p(\boldsymbol{\mu})}}{|\boldsymbol{\beta}(\boldsymbol{\mu})|} L_{SS} w^{\mathcal{N}} \right)_{K_p(\boldsymbol{\mu})} \\
F_{stab}(v^{\mathcal{N}}; \boldsymbol{\mu}) &= F(v^{\mathcal{N}}; \boldsymbol{\mu}) + \sum_{K_p(\boldsymbol{\mu})\in\mathcal{T}_{h,p}(\boldsymbol{\mu})} \delta_{K_p(\boldsymbol{\mu})} \left(f, \frac{h_{K_p(\boldsymbol{\mu})}}{|\boldsymbol{\beta}(\boldsymbol{\mu})|} L_{SS} v^{\mathcal{N}} \right)_{K_p(\boldsymbol{\mu})}
\end{aligned} \tag{3.32}$$

where $K_p(\boldsymbol{\mu})$ are the elements which form the mesh $\mathcal{T}_{h,p}$ defined on the original domain $\Omega_p(\boldsymbol{\mu})$.

For the analysis of stability and convergence of this method, we refer to [27].

After the introduction of this stabilization, we can use RB formulation (3.15) with new forms:

$$\begin{aligned}\frac{1}{\Delta t}m_{stab}(u^{\mathcal{N}}_{N,j}(\boldsymbol{\mu}),\zeta^{\mathcal{N}}_n;\boldsymbol{\mu}) + a_{stab}(u^{\mathcal{N}}_{N,j}(t;\boldsymbol{\mu}),\zeta^{\mathcal{N}}_n;\boldsymbol{\mu}) = \\ = g(t_j)F_{stab}(\zeta^{\mathcal{N}}_n;\boldsymbol{\mu}) + \frac{1}{\Delta t}m_{stab}(u^{\mathcal{N}}_{N,j-1}(\boldsymbol{\mu}),\zeta^{\mathcal{N}}_n;\boldsymbol{\mu})\end{aligned} \tag{3.33}$$

and test it on following examples.

3.3 Numerical Results

We are now showing some numerical results of the stabilized RB method for parabolic PDEs. The first one is the time-dependent Poiseuille-Graetz problem and the second will be the time-dependent case of the square problem of section 2.2.2.

3.3.1 Parabolic Poiseuille-Graetz problem

In this section we want to test the stabilized RB method for a time dependent Poiseuille-Graetz problem [14, 25, 47, 52]. The settings are analogous to the steady case in section 2.2.1.

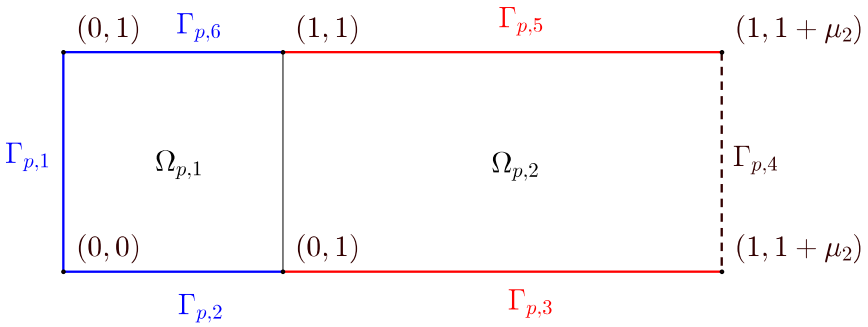


Figure 3.1: Geometry of Graetz problem. Parametrized domain. Boundary conditions: homogeneous Dirichlet on blu sides, u=1 on red sides, homogeneous Neumann on the dashed side

Let $\boldsymbol{\mu} = (\mu_1, \mu_2) \in \mathbb{R}^2$ such that $\mu_1, \mu_2 > 0$. For each value of the parameter $\boldsymbol{\mu}$, let $\Omega_p(\boldsymbol{\mu})$ be the rectangle in $\mathbb{R}^2$ sketched in figure 3.1. We first subdivide $\Omega_p(\boldsymbol{\mu})$ into two subdomains, $\Omega_{p,2}(\boldsymbol{\mu})$ and $\Omega_{p,1}(\boldsymbol{\mu})$ and then subdivide the boundary into 6 parts $\Gamma_{p,i}$, $i = 1, \dots, 6$. Then we define the time interval $I = [0, T]$.

The problem is to find the temperature distribution $u(\boldsymbol{\mu})$ such that:

$$\begin{cases} \partial_t u(\boldsymbol{\mu}) - \frac{1}{\mu_1}\Delta u(\boldsymbol{\mu}) + 4y(1-y)\partial_x u(\boldsymbol{\mu}) = 0 & \text{in } \Omega_p(\boldsymbol{\mu}) \\ u(\cdot, t; \boldsymbol{\mu}) = 0 & \text{on } \Gamma_{p,1}(\boldsymbol{\mu}) \cup \Gamma_{p,2}(\boldsymbol{\mu}) \cup \Gamma_{p,6}(\boldsymbol{\mu}) \\ u(\cdot, t; \boldsymbol{\mu}) = 1 & \text{on } \Gamma_{p,3}(\boldsymbol{\mu}) \cup \Gamma_{p,5}(\boldsymbol{\mu}) \\ \frac{\partial u}{\partial \nu}(\cdot, t; \boldsymbol{\mu}) = 0 & \text{on } \Gamma_{p,4}(\boldsymbol{\mu}) \\ u(\cdot, 0; \boldsymbol{\mu}) \equiv 1 & \text{in } \Omega_p(\boldsymbol{\mu}) \end{cases} \tag{3.34}$$

Here boundary conditions are steady. Only in the next section we will use time-dependent boundary conditions.

As in section 2.2.1, we have a reference domain $\Omega = \Omega_p(\cdot, 1)$ with its subdomains Ω_i, $i = 1, 2$, and boundary regions Γ_i, $i = 1, \dots, 6$, and a linear transformation $T(\boldsymbol{\mu}) : \Omega \to \Omega_p(\boldsymbol{\mu}) \subset \mathbb{R}^2$ defined as in the steady case (2.49) and (2.50). We build two triangulations over subdomains Ω_i:$\mathcal{T}_h^i$, $i = 1, 2$ such that their union is a proper triangulation $\mathcal{T}_h$ on Ω. We can define the approximation space $X^{\mathcal{N}} = \mathbb{P}^1(\mathcal{T}_h) \cap H_0^1(\Omega)$.

We define the lifting of the boundary data l as a function in $\mathbb{P}^1(\mathcal{T}_h)$ such that $l|_{\Gamma_{1,2,6}} \equiv 0$ and $l|_{\Gamma_{3,5}} \equiv 1$.

We can write the weak formulation of the problem (3.34) and then track it back on the reference domain, similarly to section 2.2.1. We obtain the following *Backward Euler Stabilized* FE problem:

for each $1 \le j \le J$, find $u_j^{\mathcal{N}}(\boldsymbol{\mu}) \in X^{\mathcal{N}}$ s.t.

$$\frac{1}{\Delta t} m_{stab}(u_j^{\mathcal{N}}(\boldsymbol{\mu}) - u_{j-1}^{\mathcal{N}}(\boldsymbol{\mu}), v^{\mathcal{N}}; \boldsymbol{\mu}) + a_{stab}(u_j^{\mathcal{N}}(t; \boldsymbol{\mu}), v^{\mathcal{N}}; \boldsymbol{\mu}) = g(t_j) F_{stab}(v^{\mathcal{N}}; \boldsymbol{\mu}) \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}},$$

given the initial condition $u_0^{\mathcal{N}}$ s.t.

$$(u_0^{\mathcal{N}}, v^{\mathcal{N}})_{L^2(\Omega)} = (1, v^{\mathcal{N}})_{L^2(\Omega)} \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}}. \tag{3.35}$$

where the stabilized terms are

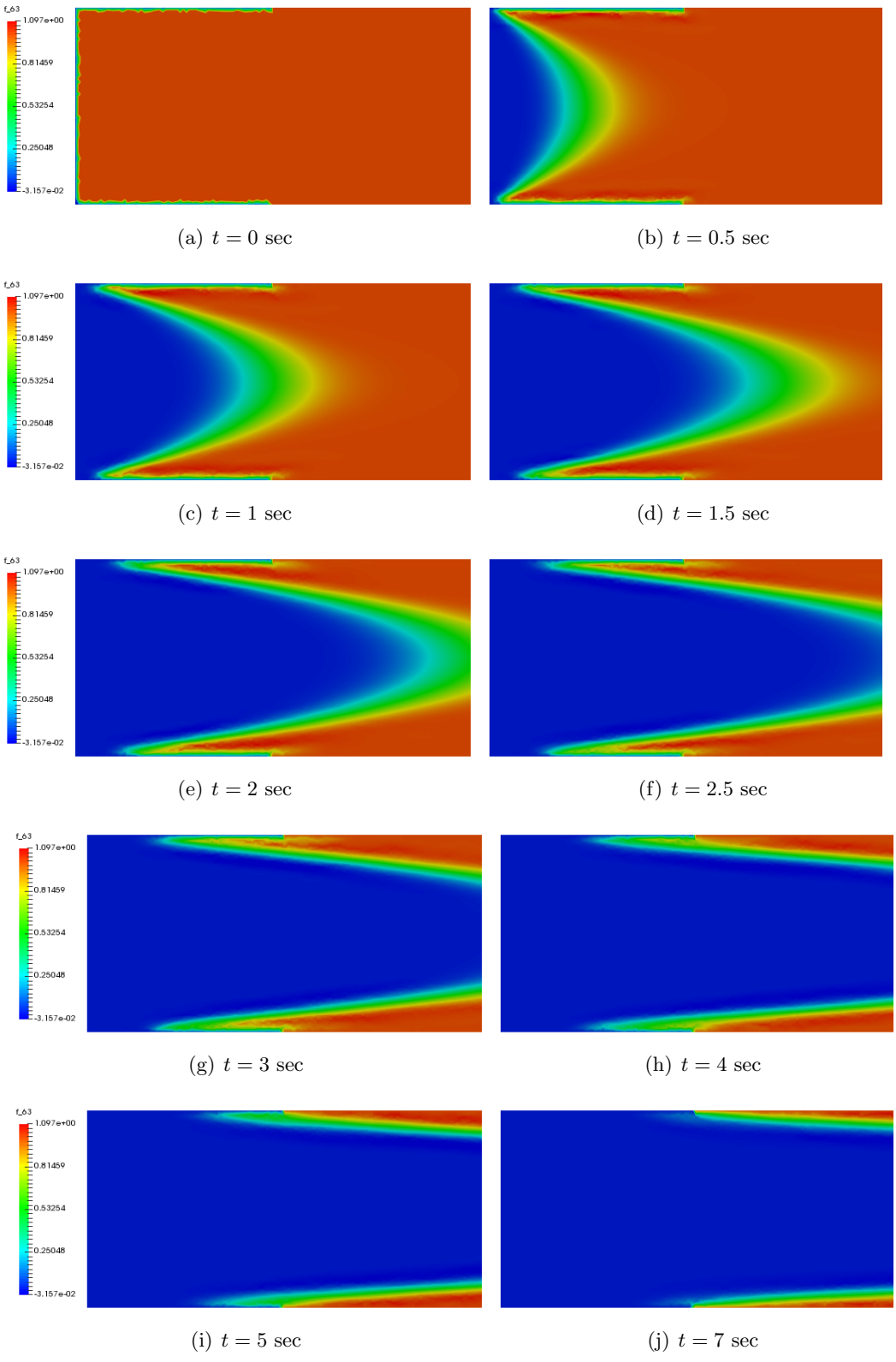
$$\begin{aligned} m_{stab}(v^{\mathcal{N}}, w^{\mathcal{N}}; \boldsymbol{\mu}) = & \int_{\Omega_1} v^{\mathcal{N}} w^{\mathcal{N}} + \sum_{K \in \mathcal{T}^1} \delta_K h_K \int_K v^{\mathcal{N}} \partial_x w^{\mathcal{N}} + \\ & + \int_{\Omega_2} \frac{\mu_2}{\mu_1} v^{\mathcal{N}} w^{\mathcal{N}} + \sum_{K \in \mathcal{T}^2} \delta_K \frac{h_K}{\sqrt{\mu_2}} \int_K v^{\mathcal{N}} \partial_x w^{\mathcal{N}} \\ a_{stab}(v^{\mathcal{N}}, w^{\mathcal{N}}; \boldsymbol{\mu}) = & \int_{\Omega_1} \frac{1}{\mu_1} \nabla v^{\mathcal{N}} \cdot \nabla w^{\mathcal{N}} + 4y(1-y)\partial_x v^{\mathcal{N}} w^{\mathcal{N}} + \\ & + \sum_{K \in \mathcal{T}^1} \delta_K h_K \int_K 4y(1-y)\partial_x v^{\mathcal{N}} \partial_x w^{\mathcal{N}} + \\ & + \int_{\Omega_2} \frac{1}{\mu_1 \mu_2} \partial_x v^{\mathcal{N}} \partial_y w^{\mathcal{N}} + \frac{\mu_2}{\mu_1} \partial_x v^{\mathcal{N}} \partial_x w^{\mathcal{N}} + \mu_1 \mu_2 4y(1-y)\partial_x v^{\mathcal{N}} w^{\mathcal{N}} + \\ & + \sum_{K \in \mathcal{T}^2} \delta_K \frac{h_K}{\sqrt{\mu_2}} \int_K 4y(1-y)\partial_x v^{\mathcal{N}} \partial_x w^{\mathcal{N}} \\ F_{stab}(v^{\mathcal{N}}; \boldsymbol{\mu}) = & - a_{stab}(l, v^{\mathcal{N}}; \boldsymbol{\mu}) \end{aligned} \tag{3.36}$$

for all $v^{\mathcal{N}}, w^{\mathcal{N}} \in X^{\mathcal{N}}$.

Now, we can see in picture 3.2 some snapshots at different time for stabilized FE solution (our *truth* solution) at $\mu_1 = 1$ and $\mu_2 = 1 \cdot 10^4$. We can notice that in parabolic case, instabilities near the boundary layer after the stabilization are almost disappeared. In this tests we have used a timestep of $\Delta t = 0.3$ seconds, with the same mesh used for the steady case ($h_{max} = 0.037$).

Finally, we can compute the RB method and test the different methods and compare them. In figure 3.3 we can see that the GS$^{\perp}$–POD–Greedy algorithm, with respect to the POD–Greedy one, can not increase its delta or error as the dimension of the reduced basis space increases. In this Graetz case this fact is an advantage, indeed, it happens that the POD–Greedy algorithm increases Δ_N and the error, while the other is always decreasing in a more *regular* way.
The most important disadvantage of the POD–Greedy algorithm is that it does not use the information collected during the Greedy algorithm in the first POD step. During this step, it happens that often it choose some samples that are very similar to others already chosen and that during the second POD it simply take a little information from basis chosen, or even nothing new. While GS$^{\perp}$–POD–Greedy considers only information that are not already used in the previous part of the algorithm and this guarantees an always decreasing error and Δ_N.
In other cases, this could be also a disadvantage, for example, when the algorithm take in consideration some basis that are less important than others, the POD–Greedy algorithm can get rid of these ones during the second POD, while the GS$^{\perp}$–POD–Greedy keeps all the information.

We want to notice that the error in absolute value, is a lot larger than the steady case one. With a reduced basis space of dimension 20 the error is still around 1, while for the steady case, with same dimension reduced basis space, we have an error of 10^{-3}. If we want, for example, to set a tolerance on the greedy algorithm and reach an error of 10^{-2} we have to compute a reduced basis space of dimension 83. This is due to the large diversity between snapshots taken at different times and different parameter values.



(a) $t = 0$ sec (b) $t = 0.5$ sec

(c) $t = 1$ sec (d) $t = 1.5$ sec

(e) $t = 2$ sec (f) $t = 2.5$ sec

(g) $t = 3$ sec (h) $t = 4$ sec

(i) $t = 5$ sec (j) $t = 7$ sec

Figure 3.2: Plot of FE solution for parabolic Graetz problem at different times at $\mu_1 = 1$ and $\mu_2 = 1 \cdot 10^4$

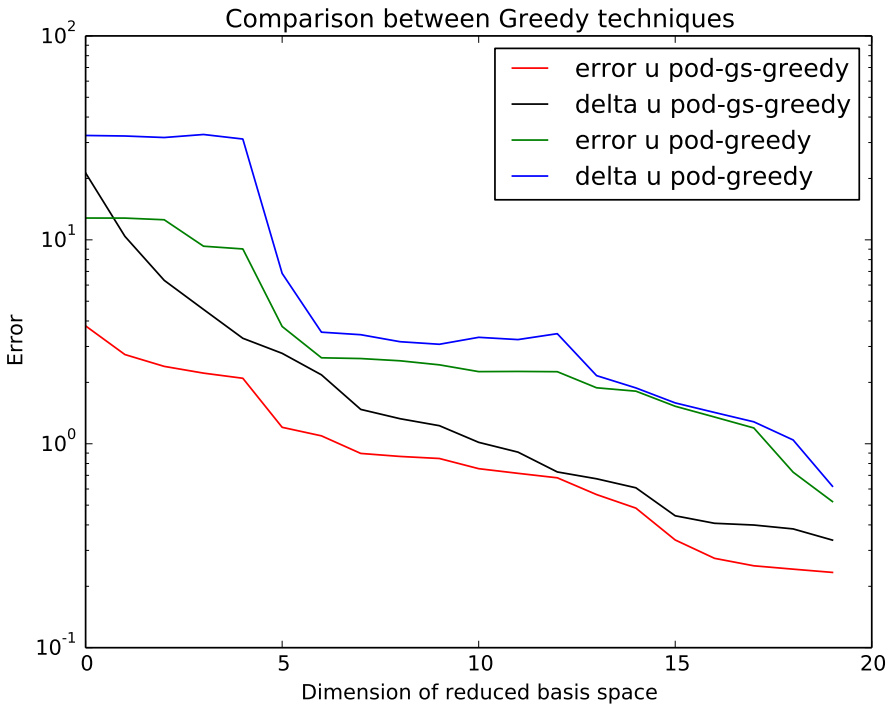


Figure 3.3: Error of POD–Greedy algorithm and $GS^{\perp}$–POD–Greedy algorithm

3.3.2 Propagating front in a square parabolic problem

Let consider the same geometry of figure 3.4, i.e. a square $\Omega \subset \mathbb{R}^2$, with boundary subdivided in two zone: red one composed by $\Gamma_1 \cup \Gamma_2$ and the blue one $\Gamma_3 \cup \Gamma_4 \cup \Gamma_5$.
Let us denote I the time interval $[0, T]$. Let $\boldsymbol{\beta} = (\mu_1, \mu_2)$, with $\mu_1, \mu_2 \in \mathbb{R}$. The problem

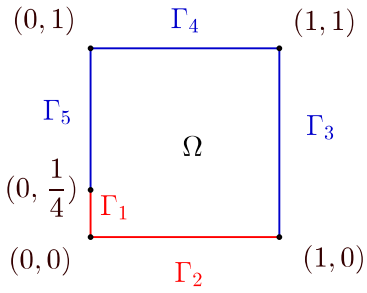


Figure 3.4: Propagating front problem geometry

we are dealing with is the following:

$$\begin{cases} \partial_t u(\boldsymbol{\mu}) - \frac{1}{\mu_1}\Delta u(\boldsymbol{\mu}) + (\cos\mu_2, \sin\mu_2) \cdot \nabla u(\boldsymbol{\mu}) = 0 & \text{in } \Omega \times I \\ u(\cdot, t; \boldsymbol{\mu}) = g(t) & \text{on } \Gamma_1 \cup \Gamma_2 \\ u(\cdot, t; \boldsymbol{\mu}) = 0 & \text{on } \Gamma_3(\boldsymbol{\mu}) \cup \Gamma_4 \cup \Gamma_5 \\ u(\cdot, 0; \boldsymbol{\mu}) \equiv 0 & \text{in } \Omega_p(\boldsymbol{\mu}) \end{cases} \tag{3.37}$$

where g is a control function, $g(t) = cos(t)$.

To build our approximation procedure, we first define a triangulation $\mathcal{T}_h$, with which we can define the polynomial approximation space $\mathbb{P}^1$ as in the steady case (section 2.2.2). Moreover we define $X^{\mathcal{N}} = \mathbb{P}^1 \cap H_0^1(\Omega)$. And we obtain the stabilized *Backward-Euler* FE formulation (3.31), that we recall

$$\begin{aligned} &\text{for each } 1 \le j \le J, \text{ find } u_j^{\mathcal{N}}(\boldsymbol{\mu}) \in X^{\mathcal{N}} \text{ s.t.} \\ &\frac{1}{\Delta t} m_{stab}(u_j^{\mathcal{N}}(\boldsymbol{\mu}) - u_{j-1}^{\mathcal{N}}(\boldsymbol{\mu}), v^{\mathcal{N}}; \boldsymbol{\mu}) + a_{stab}(u_j^{\mathcal{N}}(t; \boldsymbol{\mu}), v^{\mathcal{N}}; \boldsymbol{\mu}) = \\ &\qquad = g(t_j) F_{stab}(v^{\mathcal{N}}; \boldsymbol{\mu}) \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}}, \\ &\text{given the initial condition } u_0^{\mathcal{N}} \text{ s.t.} \\ &(u_0^{\mathcal{N}}, v^{\mathcal{N}})_{L^2(\Omega)} = (u_0, v^{\mathcal{N}})_{L^2(\Omega)} \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}}. \end{aligned} \tag{3.38}$$

where, for all $v^{\mathcal{N}}, w^{\mathcal{N}} \in X^{\mathcal{N}}$, we have:

$$\begin{aligned} m_{stab}(v^{\mathcal{N}}, w^{\mathcal{N}}; \boldsymbol{\mu}) &= \int_\Omega v^{\mathcal{N}} w^{\mathcal{N}} + \sum_{K \in \mathcal{T}} \delta_K h_K \left(v^{\mathcal{N}}, (\cos\mu_2, \sin\mu_2) \cdot \nabla w^{\mathcal{N}} \right)_K \\ a_{stab}(v^{\mathcal{N}}, w^{\mathcal{N}}; \boldsymbol{\mu}) &= \int_\Omega \frac{1}{\mu_1} \nabla v^{\mathcal{N}} \cdot \nabla w^{\mathcal{N}} + (\cos\mu_2, \sin\mu_2) \cdot \nabla v^{\mathcal{N}} w^{\mathcal{N}} + \\ &\quad + \sum_{K \in \mathcal{T}} \delta_K h_K \left((\cos\mu_2, \sin\mu_2) \nabla v^{\mathcal{N}} \cdot \nabla w^{\mathcal{N}} \right)_K \\ F_{stab}(v^{\mathcal{N}}; \boldsymbol{\mu}) &= \sum_{K \in \mathcal{T}} \delta_K h_K \left(f_h, (\cos\mu_2, \sin\mu_2) \cdot \nabla w^{\mathcal{N}} \right)_K \end{aligned} \tag{3.39}$$

where f_h is a lifting function corresponding to the boundary condition $u = 1$ on $\partial\Omega$. We will use, as in section 2.2.2, $\delta_K = 1$ for every element. We are using a timestep $\Delta t = 0.2$ with a maximum time $T = 3$ seconds.
In figure 3.6 we can see some stabilized FE solution at a finer time step ($\Delta t = 0.08$) at different times. We can see that even in this case the instabilities along boundary layers are well smoothed by the stabilization, indeed at each time, the solution is always under 1, while in the steady–case, it was often over (physically senseless).

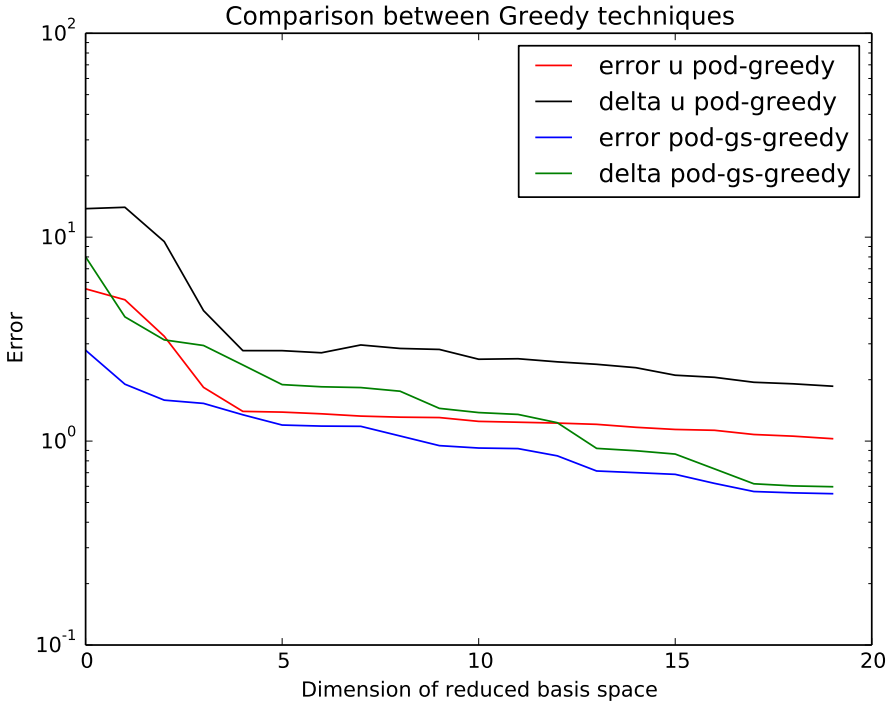
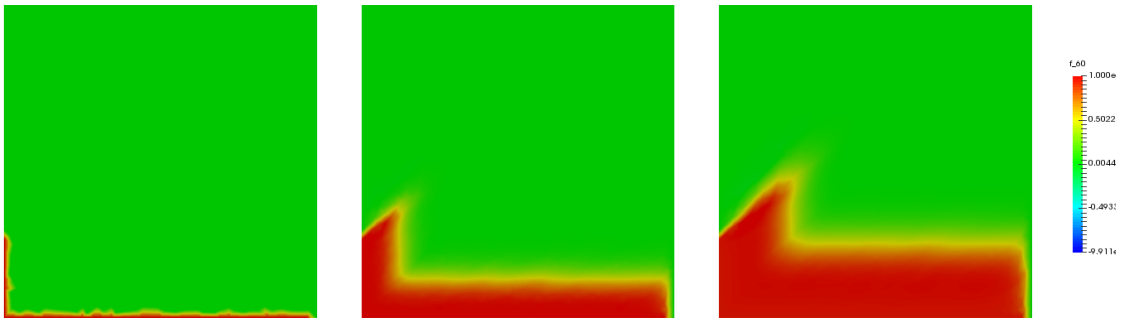
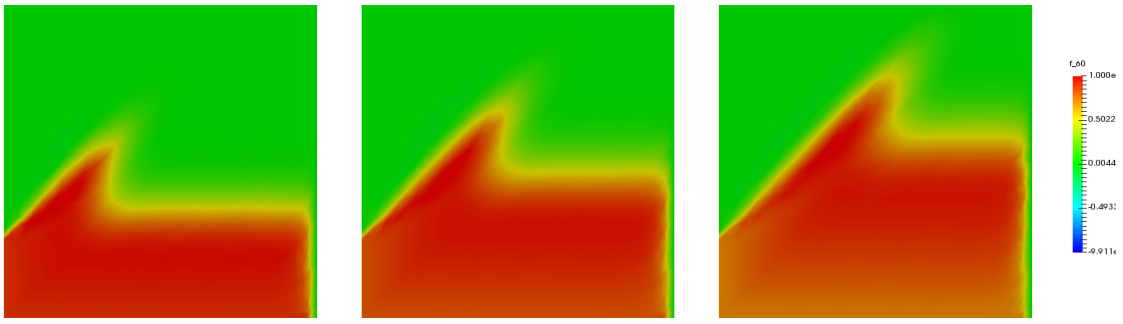


Figure 3.5: Error of POD–Greedy algorithm and $GS^{\perp}$–POD–Greedy algorithm

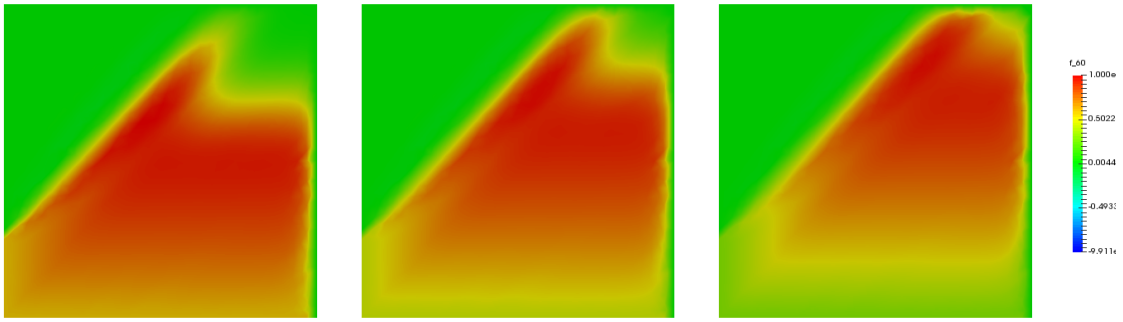
As for the Graetz problem, we can compare the two algorithms proposed in figure 3.5. We can notice the same behaviour of the previous case. The $GS^{\perp}$–POD–Greedy algorithm is getting lower errors with same dimension of reduced basis space with respect to POD–Greedy algorithm.



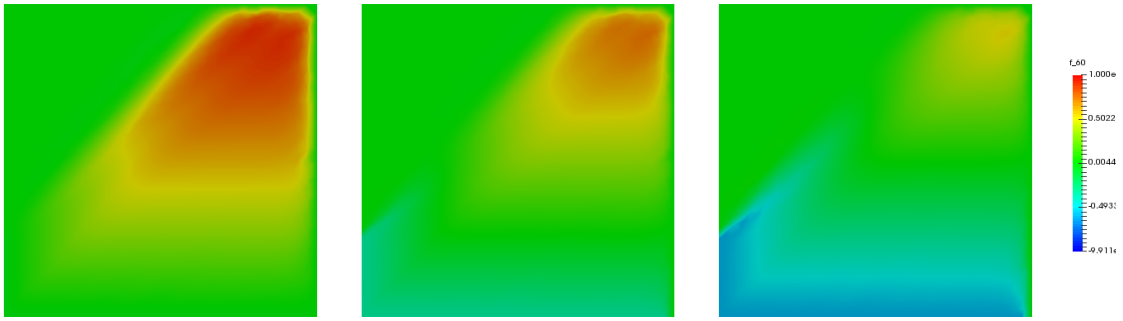
(a) $t = 0$ sec (b) $t = 0.16$ sec (c) $t = 0.32$ sec



(d) $t = 0.48$ sec (e) $t = 0.64$ sec (f) $t = 0.8$ sec



(g) $t = 0.96$ sec (h) $t = 1.12$ sec (i) $t = 1.28$ sec



(j) $t = 1.6$ sec (k) $t = 1.92$ sec (l) $t = 2.24$ sec

Figure 3.6: Plot of FE solution for parabolic Square problem at different times, $\mu_1 = 2 \cdot 10^4$, $\mu_2 = 0.8$

Chapter 4

Weighted reduced basis algorithm for stabilized transport PDEs

In this chapter we will discuss a refinement of the reduced basis method in particular situations. Until now, we have dealt with parameters that were uniformly distributed on their ranges. In order to deal with more general uncertainty problems with random distributed parameters, we will propose an extended version of the reduced basis method called "weighted reduced basis method" [6].
The main idea of this method is to suitably assign a larger weight to those samples that are more important. We expect a better convergence of this algorithm over a distributed test set.

Moreover, we will compare this method to the standard reduced basis method for advection–diffusion problems with high Péclet number. In order to do this, we will use all the stabilization techniques used in chapters 2 and 3.

We will also provide an offline/online stabilization approach that can be useful in case when stabilization involves large computations.

Finally, we will provide some numerical examples on problems presented in previous chapters.

Similar methods are studied even for nonlocal diffusion problems. In [17] we can see both reduced basis methods and uncertainty quantification in sense of random input parameters applied to nonlocal diffusion problems. This tell us that our weighted reduced basis method can be extended to other problems different from PDEs.

4.1 Weighted Reduced Basis Method

In this section, we will introduce stochastic partial differential equations and then, we will discuss the weighted Reduced Basis method [7].

First of all, we have to set an appropriate mathematical environment to define our problems. As in previous chapters, let Ω be an open set of $\mathbb{R}^2$ with Lipschitz boundary $\partial\Omega$ and let $H_0^1(\Omega) \subset X \subset H^1(\Omega)$ a functional space.

Let $(A, \mathcal{F}, P)$ denote a complete probability space, where A is a set of outcomes $\omega \in A$, $\mathcal{F}$ is a σ-algebra of events and $P : \mathcal{F} \to [0, 1]$ with $P(A) = 1$ is a probability measure [11].

A real-valued *random variable* is defined as a measurable function $Y : (A, \mathcal{F}) \to (\mathbb{R}, \mathcal{B})$, being $\mathcal{B}$ the Borel σ-algebra on $\mathbb{R}$.
The distribution function of a random variable $Y : A \to \mathcal{D} \subset \mathbb{R}$, being $\mathcal{D}$ the image of Y, is defined as $F_Y : \mathcal{D} \to [0, 1]$ such that $\forall y \in \mathcal{D}, F_Y(y) = P(\omega \in A : Y(\omega) \leq y)$. Let $dF_Y(y)$ denote the distribution measure, i.e., for all $B \subset \mathcal{D}$ $P(F \in B) = \int_B dF_Y(y)$. Provided that $dF_Y(y)$ is absolutely continuous with respect to the Lebesgue measure dy, which we assume hereafter to be the case, there exists a probability density function $\rho : \mathcal{D} \to \mathbb{R}$ such that $\rho(y)dy = dF_Y(y)$. Note that the new measure space $(\mathcal{D}, \mathcal{B}(\mathcal{D}), \rho(y)dy)$ is isometric to $(A, \mathcal{F}, P)$ under the random variable Y.

We define the probability Hilbert space $L^2(A) := \{v : A \to \mathbb{R} : \int_A v^2(\omega)dP(\omega) < \infty\}$ and $L^2_\rho(\mathcal{D}) := \{w_\mathcal{D} \to \mathbb{R} | \int_\mathcal{D} w^2(y)\rho(y)dy < \infty\}$, equipped with the equivalent norms (by noting that $v(\omega) = w(y(\omega))$)

$$||v||_{L^2(A)} := \left(\int_A v^2(\omega)dP(\omega)\right)^{1/2} = \left(\int_\mathcal{D} w^2(y)\rho(y)dy\right)^{1/2} =: ||w||_{L^2_\rho(\mathcal{D})}. \tag{4.1}$$

Let $v : \Omega \times A \to \mathbb{R}$ be a real-valued *random field*, which is a real-valued random variable defined on A for each $x \in \Omega$. We define the Hilbert space $S(\Omega) := L^2(A) \bigotimes H^1(\Omega)$, equipped with the inner product

$$(w, v) = \int_A \int_\Omega (wv + \partial_1 v \partial_1 w + \partial_2 v \partial_2 w)dx_1 dx_2 dP(\omega) \quad \forall w, v \in S(\Omega), \tag{4.2}$$

where ∂_i is the partial derivative along the ith coordinate of Ω. The associated norm is defined as $||v||_{S(\Omega)} = \sqrt{(v, v)}$.

Now we can introduce *stochastic partial differential equations*. Given random vector field $\boldsymbol{\mu} : A \to \mathbb{R}^p$, our stochastic advection-diffusion problem will be finding a random field $u(\boldsymbol{\mu}(\omega), x)$ such that

$$-\varepsilon(\boldsymbol{\mu}(\omega))\Delta u(\boldsymbol{\mu}(\omega)) + \boldsymbol{\beta}(\boldsymbol{\mu}(\omega)) \cdot \nabla u(\boldsymbol{\mu}(\omega)) = 0 \qquad \text{on } \Omega(\boldsymbol{\mu}(\omega)) \tag{4.3}$$

according to boundary conditions.

As in previous chapter, we have to consider the weak formulation of this problem, for example (2.52), then a discretized version of this weak formulation with FE, for example (2.60), and we have to remember the RB approach and the greedy algorithm of chapter 1.

Now, we want to develop an algorithm that gives more importance to parameters with higher probability of been chosen. The basic idea is to assign different weights to every values of parameter $\boldsymbol{\mu} \in \mathcal{D} \subset \mathbb{R}^P$ inside the RB algorithm according to a prescribed weight function $w(\boldsymbol{\mu})$, during the RB procedure of construction of the RB space. The motivation is that when the parameter $\boldsymbol{\mu}$ has various weights $w(\boldsymbol{\mu})$ at different values $\boldsymbol{\mu} \in \mathcal{D}$, e.g., stochastic problems with random inputs obeying probability distribution far from uniform type, the weighted approach can considerably attenuate the computational effort for large scale computational problems. The weighted reduced basis method consists of the same elements, namely greedy algorithm, a posteriori error estimate and offline-online decomposition, as presented in chapter 1. In this chapter, we only highlight the new weighted scheme.

Let $X^{\mathcal{N}}$ be a high-fidelity approximation space of X, equipped with the norm $||u||_X = \sqrt{a(u,u;\tilde{\boldsymbol{\mu}})} \quad \forall u \in X$ at some reference value $\tilde{\boldsymbol{\mu}} \in \mathcal{D}$. Let $X_w^{\mathcal{N}}$ be a weighted approximation space (actually $X^{\mathcal{N}}$) with norm

$$||u(\boldsymbol{\mu})||_w = w(\boldsymbol{\mu})||u(\boldsymbol{\mu})||_X \quad \forall u \in X^{\mathcal{N}}, \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{4.4}$$

where $w : \mathcal{D} \to \mathbb{R}^+$ is a weighted function taking positive real values.

Recalling the greedy algorithm (1), at each step we want to find $\boldsymbol{\mu}^*$ such that it maximize the error between the *truth* solution $u^{\mathcal{N}}(\boldsymbol{\mu})$ and the reduced basis solution $u_N(\boldsymbol{\mu})$. Similarly, the greedy weighted algorithm essentially deals with the $L^\infty(\mathcal{D}; X_w)$ optimization problem in a greedy way, seeking a new parameter $\boldsymbol{\mu}^N \in \mathcal{D}$ such that

$$\boldsymbol{\mu}^N = \arg\sup_{\boldsymbol{\mu}\in\mathcal{D}} ||u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})||_w, \tag{4.5}$$

where again u_N is the reduced basis approximation of the *truth* solution $u^{\mathcal{N}}$. As before we want to replace the parameter space $\mathcal{D}$ with its discretized version Ξ_{train}. Instead of performing the true error, we use a weighted *a posteriori* error estimator Δ_N^w such that

$$||u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})||_w \leq \Delta_N^w(\boldsymbol{\mu}). \tag{4.6}$$

The choice of the weight function $w(\boldsymbol{\mu})$ is aimed by the desire of minimizing the square norm error of the RB approximation in the space $L^\infty(\mathcal{D}; X_w)$, i.e.

$$\begin{aligned}\mathbb{E}[(u^{\mathcal{N}} - u_N)^2] &= \int_A \int_\Omega (u^{\mathcal{N}}(\boldsymbol{\mu}(\omega)) - u_N(\boldsymbol{\mu}(\omega)))^2 dx dP(\omega) = \\ &= \int_{\mathcal{D}} \int_\Omega (u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu}))^2 dx \rho(\boldsymbol{\mu}) d\boldsymbol{\mu},\end{aligned} \tag{4.7}$$

that we can bound with

$$\mathbb{E}\left[(u^{\mathcal{N}} - u_N)^2\right] \leq \int_{\mathcal{D}} \Delta_N(\boldsymbol{\mu})^2 \rho(\boldsymbol{\mu}) d\boldsymbol{\mu}, \tag{4.8}$$

where Δ_N is the error estimator (1.53).
So we will use as weighted function $w(\boldsymbol{\mu}) = \sqrt{\rho(\boldsymbol{\mu})}$ and we set $\Delta_N^w(\boldsymbol{\mu}) := \Delta_N(\boldsymbol{\mu})\sqrt{\rho(\boldsymbol{\mu})}$.[7]

Another important aspect in the RB algorithm is the choice of the training set Ξ_{train}. Usually we use Uniform Monte Carlo sampling methods to choose elements from $\mathcal{D}$, while, in this context, we can use a Monte Carlo sampling according to the distribution $\rho(\boldsymbol{\mu})$. We will see in numerical test that this choice is important to improve the convergence of the error.
For further convergence analysis we refer to [6, 7].

4.2 Stabilized weighted reduced basis

The weighted reduced basis method has been studied in [6, 7] and in other works, but it was never tested on advection–diffusion equations with high Péclet number.
We want to test it with our examples (Graetz problem 2.2.1 and propagating front problem

2.2.2). As before we will have a domain $\Omega_p(\boldsymbol{\mu})$ a strong formulation of the problems (2.48) (2.68) and their FE weak formulations (2.55) (2.71). As in section 2.1, for the moment, we will add the SUPG stabilization terms getting formulations (2.63) (2.70). The difference will be $\boldsymbol{\mu} : A \to \mathcal{D}$ that, instead of being a parameter in a range, will be a random vector. For example, if we write the Graetz example of this formulation, it will be:

$$\begin{aligned} &\text{find } u^{\mathcal{N}}(\boldsymbol{\mu}) \in X^{\mathcal{N}} \text{ s.t.} \\ &a_{stab}(u^{\mathcal{N}}(\boldsymbol{\mu}(\omega)), v^{\mathcal{N}}; \boldsymbol{\mu}(\omega)) = F_{stab}(v^{\mathcal{N}}; \boldsymbol{\mu}(\omega)) \quad v^{\mathcal{N}} \in X^{\mathcal{N}}, \forall \omega \in A \end{aligned} \tag{4.9}$$

where a_{stab} and F_{stab} are defined in (2.65), $\boldsymbol{\mu}$ is a random vector. The distribution of $\boldsymbol{\mu}$ for our test will be a Beta(α, β) distribution, with different values of α and β, over the ranges that we used in previous chapters:

$$\begin{aligned} \mu_1 &\sim \mu_{1,min} + (\mu_{1,max} - \mu_{1,min}) X_1 \\ \mu_2 &\sim \mu_{2,min} + (\mu_{2,max} - \mu_{2,min}) X_2 \\ X_1 &\sim \text{Beta}(\alpha_1, \beta_1) \\ X_2 &\sim \text{Beta}(\alpha_2, \beta_2). \end{aligned} \tag{4.10}$$

We choose this distribution because it takes values in a compact set and because we can give more importance to a certain piece of the range (for example, the one with higher Péclet number).
In figure 4.1 we can see the density function of a Beta$(4, 2)$ distribution.

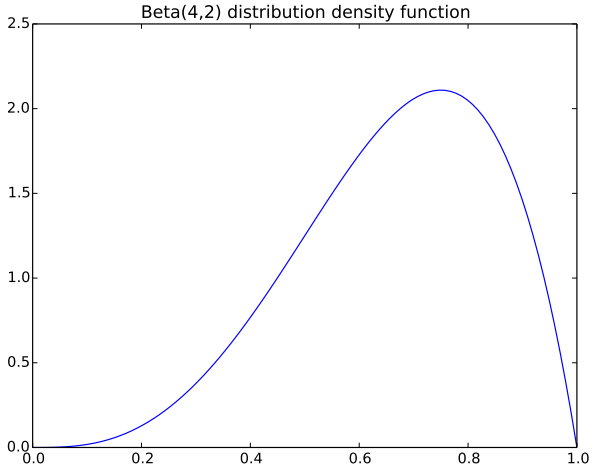


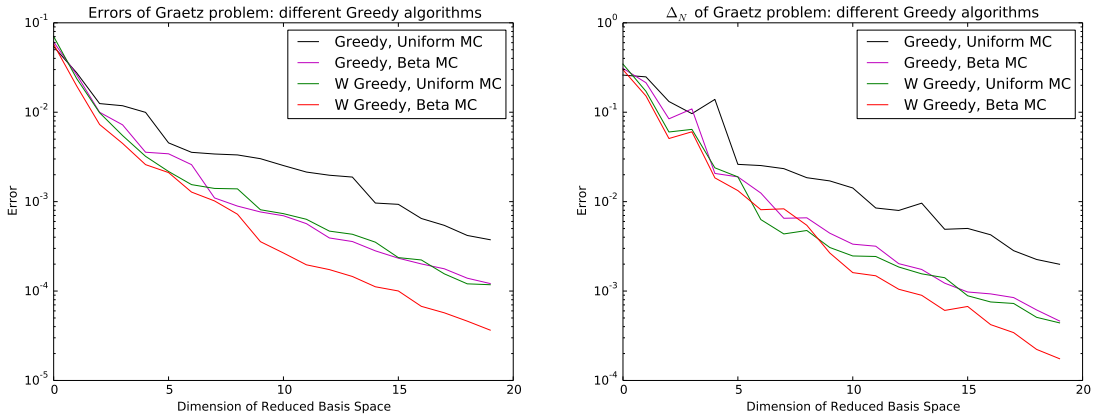
Figure 4.1: Beta(4,2)

4.2.1 Numerical test: Poiseuille-Graetz problem

For Poiseuille-Graetz problem, we consider the already used range $\mathcal{D} = \left[10^1, 10^6\right] \times [0.5, 4]$ for the parameter $\boldsymbol{\mu}$. To give more importance to parameter with $\mu_1 \approx 10^5$, we use $X_1 \sim \text{Beta}(4, 2)$ and $\mu_1 \sim 10^{1+5 \cdot X_1}$, while $X_2 \sim \text{Beta}(3, 4)$ and $\mu_2 \sim 0.5 + 3.5 X_2$.

In previous chapters we, implicitly used this algorithm with a uniform distribution over $\mathcal{D}$[1].
Before using weighted Greedy algorithm on weighted stabilized weak formulation (4.9), we have to decide how to discretize $\mathcal{D}$. Doing so in different ways, we will compare 4 algorithms:

1. Classical Greedy with Uniform Monte Carlo sampling (black in figure 4.2)
2. Classical Greedy with Beta Monte Carlo sampling (purple in figure 4.2)
3. Weighted Greedy with Uniform Monte Carlo sampling (green in figure 4.2)
4. Weighted Greedy with Beta Monte Carlo sampling (red in figure 4.2).



(a) Error Comparison

(b) Δ_N comparison

Figure 4.2: Greedy algorithms comparison for Graetz problem

We used 200 samples for Ξ_{train} in each algorithm and to evaluate errors and Δ_N we used a test set Ξ_{test} of 100 samples distributed according to $\boldsymbol{\mu}$. We can see in figure 4.2 the comparison between the errors and Δ_N between these algorithms. Both the improvements are important to make the weighted method converge faster than the classical Greedy method [55]. Clearly, putting together the two improvements (weighted error estimator and Monte Carlo sampling) we get the best result.

Overall, the stabilization of the weak formulation have not great influence on weighted method and its convergence, indeed, we have that with a reduced basis space of dimension 20, we have that its error is one tenth of the error of the classical Greedy algorithm.
Anyway, in this situation, we should be more interested in the mean of the error in a probability sense, while the one that we have plotted before was the uniform mean of errors. So, we would like to compute

$$\mathbb{E}[||u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})||_X] = \int_{\mathcal{D}} ||u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})||_X \rho(\boldsymbol{\mu}) d\boldsymbol{\mu}, \tag{4.11}$$

[1]We change μ_1 through its logarithm otherwise we will have that most (90%) of our parameters μ_1 are in $[10^5, 10^6]$

that we can approximate using some quadrature method, in particular, we will use Monte Carlo method, i.e.

$$\mathbb{E}[||u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})||_X] \approx \frac{1}{M}\sum_{i=1}^{M} ||u^{\mathcal{N}}(\boldsymbol{\mu}_i) - u_N(\boldsymbol{\mu}_i)||_X \rho(\boldsymbol{\mu}_i), \tag{4.12}$$

where $\boldsymbol{\mu}_i$, $i = 1, \dots, M$ are randomly chosen in $\mathcal{D}$.
If we want to compare for this error the classical Greedy method and the weighted reduced one, we obtain that the former is $4.5485 \cdot 10^{-4}$, while the latter is $1.2807 \cdot 10^{-4}$.

4.2.2 Numerical test: propagating front in a square problem

We can proceed in the same way for the propagating front problem. Given geometry, functional spaces and stabilization as in section 2.2.2, the stabilized stochastic weak formulation will be:

$$\begin{aligned} &\text{find } u^{\mathcal{N}}(\boldsymbol{\mu}(\omega)) \in X^{\mathcal{N}} \text{ s.t.} \\ &a_{stab}(u^{\mathcal{N}}(\boldsymbol{\mu}(\omega)), v^{\mathcal{N}}; \boldsymbol{\mu}(\omega)) = F_{stab}(v^{\mathcal{N}}; \boldsymbol{\mu}(\omega)) \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}}, \forall \omega \in A \end{aligned} \tag{4.13}$$

where all forms are defined as in (2.71).

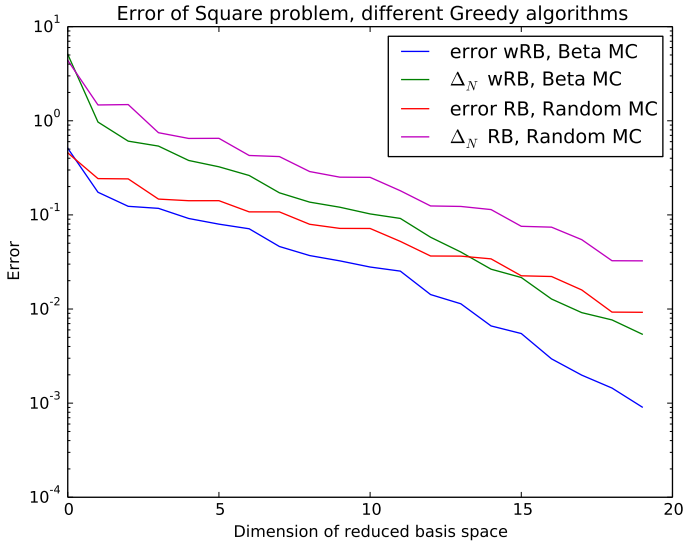


Figure 4.3: Greedy algorithms comparison for Square problem

In this section, the parameter range that we will use $\mathcal{D}$ will be $\left[10^4, 10^5\right] \times [0, 1.5]$. Even here we have that parameters μ_1 and μ_2 are random Beta variables, in particular $X_1 \sim \text{Beta}(3, 4)$ while $X_2 \sim \text{Beta}(4, 2)$ and $\mu_1 \sim 10^4 + 9 \cdot 10^4 \cdot X_1$ while $\mu_2 \sim 1.5 \cdot X_2$.
As for Graetz problem we can apply the weighted reduced basis method of section 4.1 and compare it with classical Greedy method.
In figure 4.3 we can see results very similar to Graetz problem. We have tested only

two algorithms (classical Greedy Uniform MC and weighted Greedy Beta MC) on a Ξ_{test} sample distributed as $\boldsymbol{\mu}$: the weighted RB method is converging faster than the classical one.
If we want to compare them in sense of (4.12)

$$\mathbb{E}[||u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})||_X], \tag{4.14}$$

we will have that the new method produces an error of $1.7803 \cdot 10^{-3}$, while the old one gives an error of $7.9362 \cdot 10^{-3}$ with a reduced basis space of dimension $N = 20$.

4.2.3 Numerical test: Parabolic problems

Even for parabolic cases we can proceed as before. We have stabilized weak formulations for both problems (3.31) with respectively forms (3.36) and (3.39). Here $\boldsymbol{\mu}$ is distributed as in previous sections 4.2.1 and 4.2.2.
The main difference is that our solutions will be J random fields $u_j^{\mathcal{N}}(\boldsymbol{\mu}(\omega))$, for $j = 1, \dots, J$.

$$\begin{aligned}
&\text{for each } 1 \le j \le J, \text{ find } u_j^{\mathcal{N}}(\boldsymbol{\mu}(\omega)) \in X^{\mathcal{N}} \text{ s.t.} \\
&\frac{1}{\Delta t} m_{stab}(u_j^{\mathcal{N}}(\boldsymbol{\mu}(\omega)) - u_{j-1}^{\mathcal{N}}(\boldsymbol{\mu}(\omega)), v^{\mathcal{N}}; \boldsymbol{\mu}(\omega)) + a_{stab}(u_j^{\mathcal{N}}(t; \boldsymbol{\mu}(\omega)), v^{\mathcal{N}}; \boldsymbol{\mu}(\omega)) = \\
&\qquad = g(t_j) F_{stab}(v^{\mathcal{N}}; \boldsymbol{\mu}(\omega)) \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}}, \\
&\text{given the initial condition } u_0^{\mathcal{N}} \text{ s.t.} \\
&(u_0^{\mathcal{N}}, v^{\mathcal{N}})_{L^2(\Omega)} = (u_0, v^{\mathcal{N}})_{L^2(\Omega)} \quad \forall v^{\mathcal{N}} \in X^{\mathcal{N}},
\end{aligned} \tag{4.15}$$

where stabilized forms are defined in (3.32).

As in chapter 3, the error will be the sum of errors at each time, for plot in figure 4.4 we still use the norm of the error defined in (3.28), i.e.

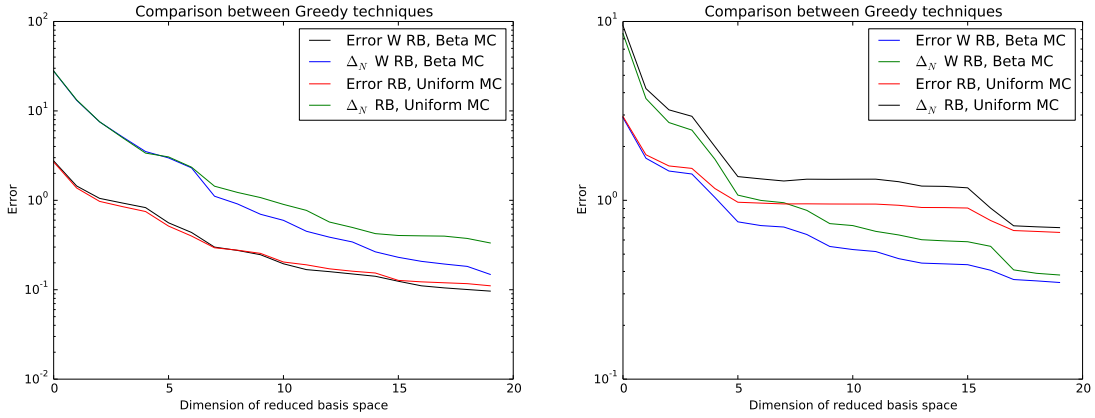
$$|||\boldsymbol{e}^{\mathcal{N}}(\boldsymbol{\mu})|||_{t-dep} = \left(m(e_J^{\mathcal{N}}(\boldsymbol{\mu}), e_J^{\mathcal{N}}(\boldsymbol{\mu}); \boldsymbol{\mu}) + \sum_{j=1}^{J} a^{sym}(e_j^{\mathcal{N}}(\boldsymbol{\mu}), e_j^{\mathcal{N}}(\boldsymbol{\mu}); \boldsymbol{\mu}) \Delta t \right)^{\frac{1}{2}}, \tag{4.16}$$

where $\boldsymbol{e}^{\mathcal{N}} = \boldsymbol{u}^{\mathcal{N}}(\boldsymbol{\mu}) - \boldsymbol{u}_N^{\mathcal{N}}(\boldsymbol{\mu})$.
We can see in figure 4.4 that for Graetz problem the difference between classical Greedy method and weighted one is not so relevant for the first 20 dimension of the reduced basis space in terms of the error, while the Δ_N error bound is improved with the modified algorithm. For propagation front problem we can see important improvements both in error and Δ_N.

If we want to consider an error analogous to (4.11), we can define

$$\mathbb{E}[\boldsymbol{u}^{\mathcal{N}} - \boldsymbol{u}_N^{\mathcal{N}}] := \sum_{j=1}^{J} \int_{\mathcal{D}} ||u_j^{\mathcal{N}} - u_{N,j}^{\mathcal{N}}||_X \rho(\boldsymbol{\mu}) d\boldsymbol{\mu} \tag{4.17}$$

and approximate it with Monte Carlo quadrature procedure.
By doing this we obtain for Graetz problem with a reduced basis space of dimension 20 an error of $8.3248 \cdot 10^{-2}$ for classic Greedy algorithm and $7.6318 \cdot 10^{-2}$ for weighted reduced



(a) Graetz Problem

(b) Propagating front problem

Figure 4.4: Greedy algorithms comparison for parabolic problems

basis algorithm. For propagating front problem we have that the classic Greedy algorithm produce an error of 0.3196 while the weighted algorithm gets 0.2343.

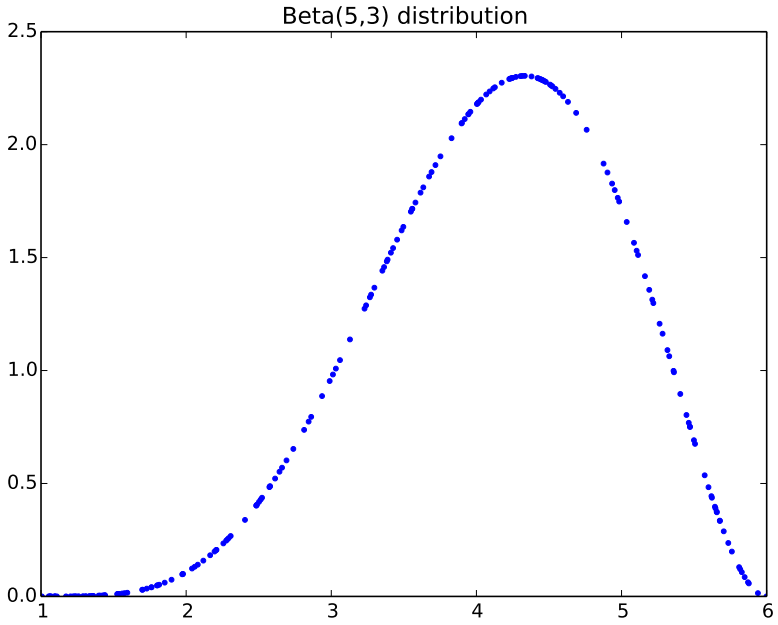
In conclusion we can say that the weighted reduced basis method can be useful in situation where we know the distribution of parameters. In the particular case where the parameters are concentrated around a high Péclet zone, we can use this algorithm to reduce our Online computation.

4.3 Offline/Online stabilized weighted reduced basis method

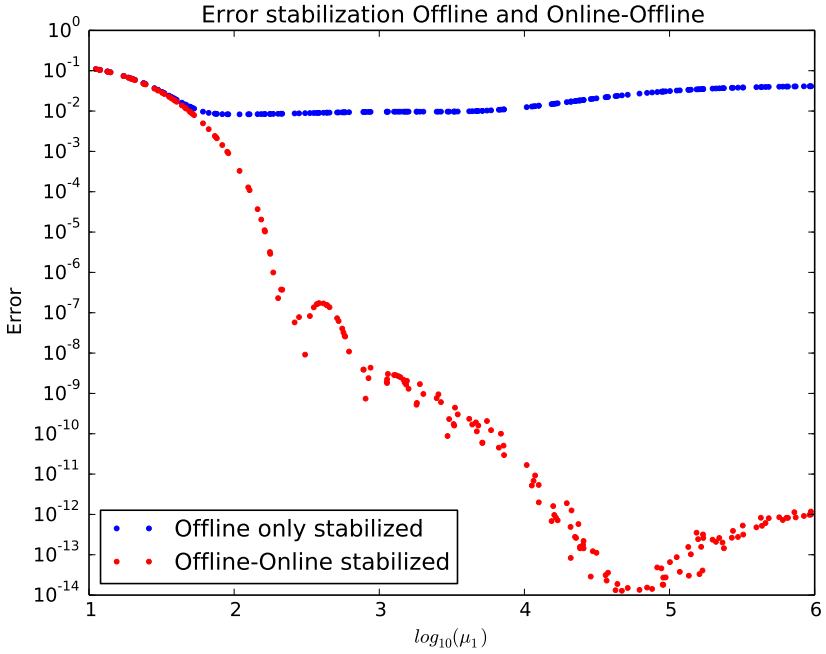
In this section we want to optimize computational costs in the Online phase of RB method. The aim of the following algorithm is to reduce long computations that stabilization procedures can have during the Online phase (for example large Q_a or Q_F). The SUPG stabilization in our examples does not have this problem (stabilization affine terms are at most 3), but, anyway, we want to test these improvements on these examples to understand if we can avoid some stabilization procedures (even for other stabilization techniques). We know that errors computed with the Offline-only stabilized method are often very large, compared with Offline-Online stabilization (for example for Graetz problem in figure 2.13), especially for high Péclet numbers. But, working in context of stochastic partial differential equation and using the error (4.11), we want to combine the Offline–only and the Offline–Online approach to reduce computational costs and not to worsen the error too much.

Let us consider the Poisuille-Graetz example, with Beta distribution over parameter $\boldsymbol{\mu}$. We change a little bit parameters to have more comprehensible plots, but results are similar to distribution used in previous example (4.10). To simplify more, we will not use the second parameter μ_2 that we fix at $\mu_2 = 1$. For μ_1, we use range $[10, 10^6]$ with $X_1 \sim \text{Beta}(3, 5)$ as in figure 4.5(a).

To sum up what we have done in the previous section, we have taken $\boldsymbol{\mu}$ as a random vector, we have taken the stochastic equation (4.3), its weak formulation and the dis-



(a) Beta(5,3) distribution on interval (1,6)



(b) Errors with stabilization Offline and Offline-Online

Figure 4.5: Error and density of Uniform Monte Carlo test set

cretized version. Then, we applied the Offline stage of the weighted RB algorithm (section 4.1) to SUPG stabilized version of this equation (4.9).
As a result of this procedure we can apply stabilized and non-stabilized Online phase to a test set (that we have taken with a Uniform Monte Carlo sampling). In figure 4.5(b) we can see which error this algorithm produces. We can observe that for low Péclet number ($\mu_1 \leq 10^2$), stabilized and non-stabilized Online phase produce almost the same error (which is a lot bigger then high Péclet numbers one due to the density of $\boldsymbol{\mu}$ and weighted RB algorithm).
We have that the zone with lower Pèclet numbers coincides with lower density $\rho(\boldsymbol{\mu})$. So, we should consider the idea of not stabilizing during the Online phase the reduced solution of parameters with low density and low Péclet numbers. Indeed, they are less relevant in terms of error (4.11) and their difference between Online–Offline stabilized error and Offline only stabilized error will be smaller than one of solutions of high Péclet numbers as we can see in figure 4.5(b).

Let us start considering the case we want to stabilize Online solutions depending on Péclet numbers. First, we establish a threshold at a certain diffusivity $\varepsilon = \frac{1}{\mu_1}$, for example $\tilde{\mu}_1 = 10^2$. For parameters $\mu_1 > \tilde{\mu}_1$ we will use both stabilization Online and Offline, while for parameter $\mu_1 \leq \tilde{\mu}_1$ we will use only stabilization Offline as we see in figure 4.6.

After this we can compute the error in sense of (4.11) and we can see that, if during

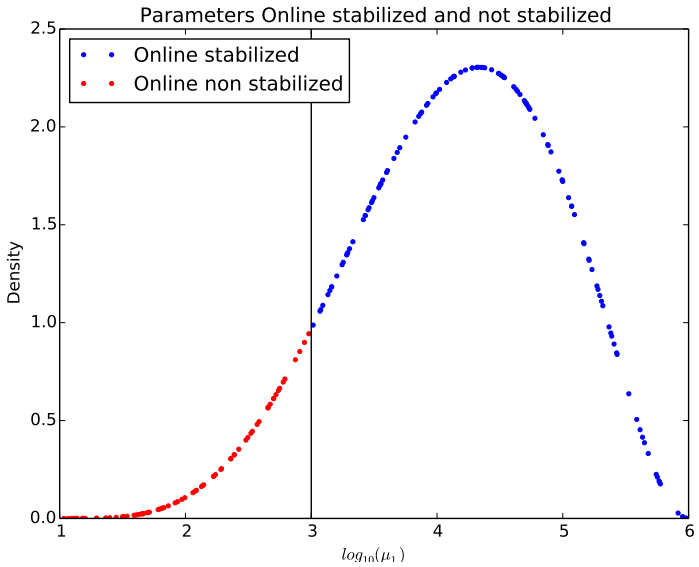


Figure 4.6: Péclet discriminant, black line is the Péclet threshold

the Online stage we do not stabilize every solution, we have an error of 0.021128, while stabilizing everything we get an error of $7.967 \cdot 10^{-4}$. For different thresholds we can

compute errors as we can see in the following table.

Threshold	Error	Percentage non-stabilized
0	$7.9673 \cdot 10^{-4}$	0%
$10^{1.5}$	$8.0704 \cdot 10^{-4}$	10%
10^{2}	$10.0060 \cdot 10^{-4}$	20%
$10^{2.5}$	$18.2806 \cdot 10^{-4}$	33%
10^{3}	$33.4593 \cdot 10^{-4}$	45%
∞	0.021128	100%

Considering that the original error was of $7.967 \cdot 10^{-4}$, we can say that until $10^{2.5}$ we are not worsening considerably the error. Anyway, even with a threshold of 10^2 we can save computations of stabilization for 19% of our test (that was Uniformly distributed), which is a good result.

The other natural gauge to decide whether stabilize the Online solution or not is the density $\rho(\boldsymbol{\mu})$. We can choose different thresholds $\tilde{\rho}$ and if $\rho(\boldsymbol{\mu}) \leq \tilde{\rho}$ we will not stabilize the Online solution. In this situation, we will decide the threshold through the percentage of integral of the density we want to stabilize or not. It means that if we want not to stabilize the 10% of the integral with lowest density, we will take the range I of parameters that are with less density such that

$$\int_I \rho(\boldsymbol{\mu}) d\boldsymbol{\mu} = 0.10, \tag{4.18}$$

in this way, I will be the set $\{\boldsymbol{\mu} : \rho(\boldsymbol{\mu}) \leq \tilde{\rho}\}$ and we will know, in probability, how much this will weigh on the whole parameter set. In figure 4.7 we can see an example of 10% of integral non stabilized online.

In the following table, we can see some results to test at different thresholds.

$\int_I \rho(\boldsymbol{\mu}) d\boldsymbol{\mu}$	Threshold $\tilde{\rho}$	Error	Percentage non-stabilized
0	0	$7.9673 \cdot 10^{-4}$	0%
0.001	0.02233	$9.3222 \cdot 10^{-4}$	15%
0.002	0.04423	$9.6456 \cdot 10^{-4}$	17%
0.005	0.09094	$14.7861 \cdot 10^{-4}$	21%
0.01	0.13877	$15.9482 \cdot 10^{-4}$	25%
0.02	0.21433	$25.6017 \cdot 10^{-4}$	30%
0.05	0.38244	$49.1931 \cdot 10^{-4}$	38%
0.1	0.89068	$66.7488 \cdot 10^{-4}$	45%
1	∞	0.021128	100%

We have that errors computed using density discriminant are less accurate than ones computed with Péclet discriminant. This is due to the enormous difference between Online stabilized and Online non stabilized solution for high Péclet numbers (figure 4.5(b)). Indeed, for the same percentage of non-stabilized solution (for example 45%) we have bigger errors in density discriminant approach ($66 \cdot 10^{-4}$ instead of $33 \cdot 10^{-4}$).

Nevertheless, we can not always use the Pèclet discriminant approach. For example, in propagating front problem we had a limited μ_1 range, which was $[10^4, 10^5]$, important

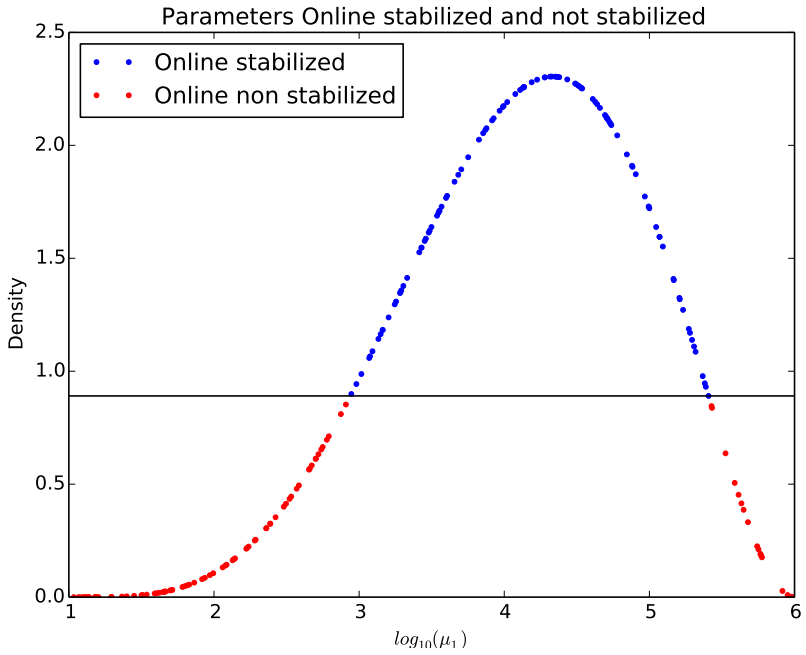


Figure 4.7: Density discriminant, black line is the density threshold

differences were given by angle μ_2 range, that was $[0, 1.5]$. Even in this case, to simplify the problem, we can fix $\mu_1 = 10^5$ and we take a Beta(4,2) distribution over the angle parameter μ_2.

We can see error of stabilized and not stabilized Online phase over a Uniform Monte Carlo test set of 200 elements in figure 4.8. We can notice that Offline–Online stabilized errors of solutions with small angles (left side of figure 4.8) are even bigger than Offline only stabilized errors (this is due to the Beta distribution and the weighted Greedy algorithm that do not give much importance to that set of the parameter range). So, we would like to stabilize only angles greater that a certain threshold, example in figure 4.9. The error in sense of (4.11) of all stabilized online solutions is of 0.01416, while with all not stabilized is of 0.82998. In next table we show different errors at different angle thresholds.

Threshold μ_2	Error	Percentage non-stabilized
0	0.01416	0%
0.1	0.01416	6%
0.2	0.01506	16%
0.3	0.04056	23%
0.4	0.11810	30%
0.5	0.20365	37%
1.5	0.82998	100%

We can observe that at the beginning the error is decreasing as the threshold increase, as we expected, while, passed a certain angle, it slowly increase. For example, a threshold of 0.2 is optimal not to increase the error and save 16% of stabilization computations.

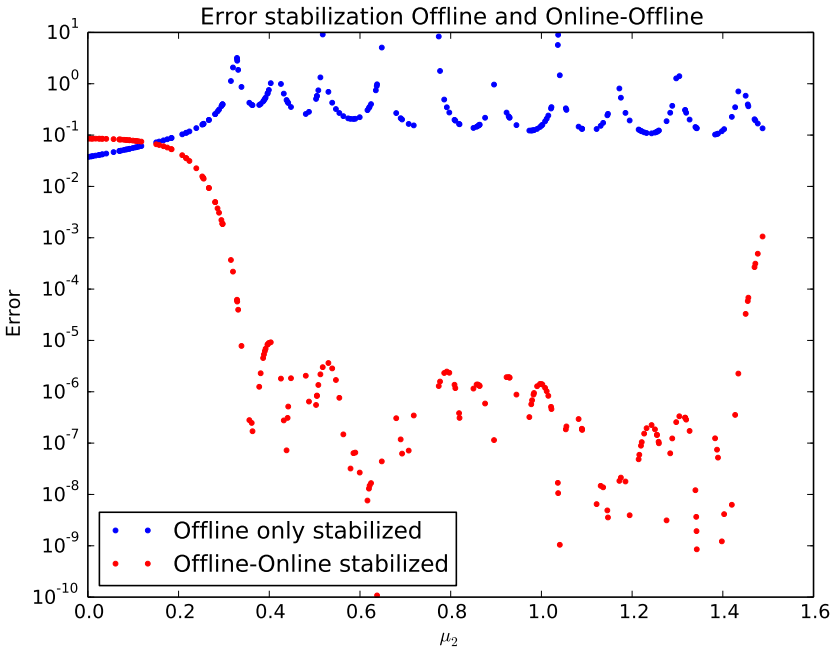


Figure 4.8: Errors stabilized and not stabilized Online phase

As for Graetz example, we can test the density threshold and see if it works better in this case. In figure 4.10 we can see an example of density threshold. In the following table, we are showing different errors for different density thresholds.

$\int_I \rho(\boldsymbol{\mu})d\boldsymbol{\mu}$	Threshold $\tilde{\rho}$	Error	Percentage non-stabilized
0	0	0.01416	0%
0.001	0.02271	0.01400	13%
0.002	0.04600	0.01506	16%
0.005	0.10237	0.02269	20%
0.01	0.13598	0.04658	25%
0.02	0.26309	0.11158	30%
0.05	0.51855	0.20613	38%
0.1	0.72557	0.32034	46%
1	∞	0.82998	100%

In this case the two methods produce almost the same results, because even big angles are not well approximated through the weighted RB method. Anyway we can say that without adding almost any error we can save 15% of stabilized Online computations, while if we want to sacrifice a little bit the error, we can arrive to 25% of computations saved (and 1% of the integral $\int_I \rho(\boldsymbol{\mu})d\boldsymbol{\mu}$).

We want to remark that, in a deterministic point of view, the Offline-Online-discriminant stabilized approach can not guarantee a good approximation for a single snapshot taken during the Online phase of the RB method. Conversely, in a stochastic error analysis it can give a lot of computational advantages.

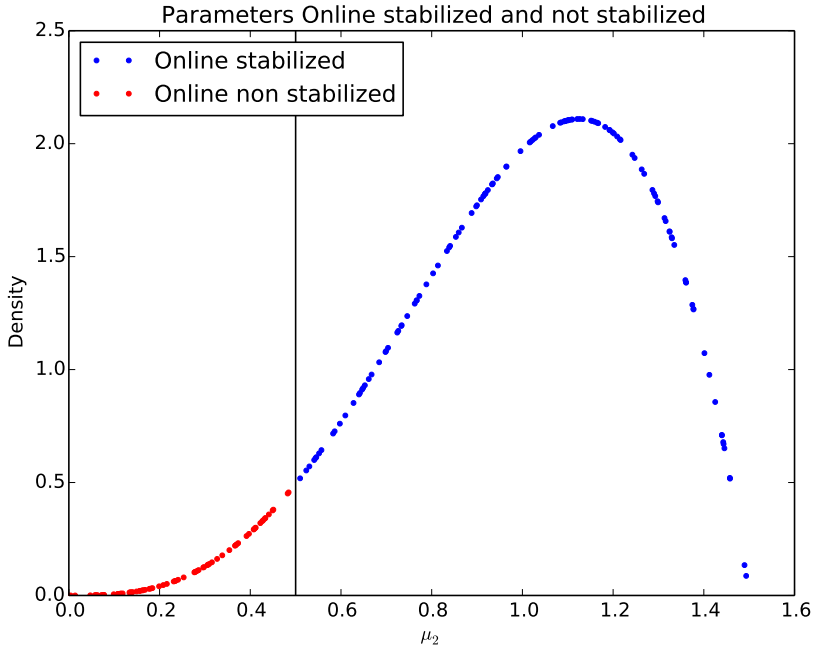


Figure 4.9: Density discriminant, black line is the density threshold

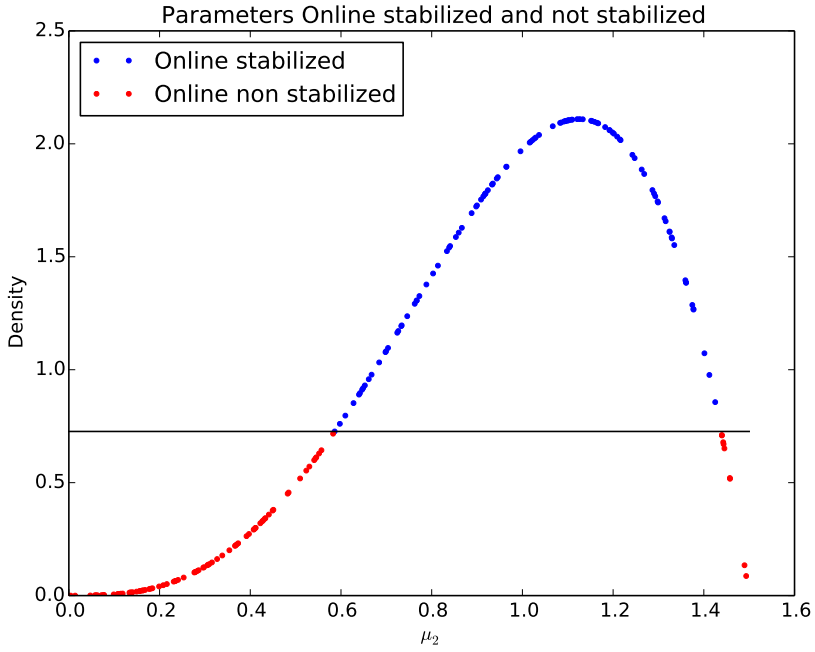


Figure 4.10: Density discriminant, black line is the density threshold

Conclusions

In this thesis we have dealt with stabilization techniques for approximation of advection dominated problems using a reduced basis approach into a stochastic framework, both in steady and unsteady case.

To perform a stabilization in reduced basis algorithm, we have studied the SUPG [49] stabilization for FE method and introduced two reduced basis stabilization algorithms. The *Online–Offline* stabilization, which uses SUPG stabilized forms in both stages (*Offline* and *Online*) and the *Offline–only* stabilization, which uses the original (not stabilized) forms for the Online stage. The underlying idea was to obtain a stable RB approximation, from the stable FE approximation, with reasonable computational times and, at the same time, a very good accuracy.

We have tested the two methods on some examples (the Poisuille–Graetz problem and the propagating front in a square problem) which have shown that the *Offline–Online* stabilization produces stable results and that the *a posteriori* error bound of RB method still guarantees a convergence. The *Offline–only* shows some instabilities and the error between RB approximation and FE one can not always converge because of declared inconsistencies between *Online* and *Offline* phase forms.

We have improved these methods to parabolic problems producing a stabilized RB approach for unsteady cases [41], starting from SUPG stabilized parabolic FE methods [5, 29]. We have proposed two different algorithms to perform the parabolic RB algorithm and we have compared them. We have tested successfully our algorithms on time dependent equations with also time dependent boundary conditions (for example Poiseuille–Graetz problem).

Then we introduced stochastic equations and weighted reduced basis method [7]. We tested a weighted stabilization reduced basis method for random input parameters and, thanks to error produced by this algorithm, we introduced a class of *Offline–Online / Offline-only* discriminant methods which allows to reduce quite well the computational load, with a negligible worsening of the error, which remains of the same order of the previous strategies one.

Possible developments of this topic can be applications of these methods to more complex geometries that requires more affine terms for forms. An other step, that can be done, could be to use non–affinely parametrized geometries, which requires an empirical interpolation pre–processing [4, 33].

Moreover, the method could be tested on larger dimension parameter spaces $\mathcal{D}$, using again Monte Carlo sampling or improving it implementing quasi-Monte Carlo strategies. Other possibilities can be to test the algorithms on other type of probability distributions

and compare different results.

Appendix A

Parabolic error estimator

In this appendix we will discuss the *a posteriori* error estimator that we will use for parabolic Greedy algorithm, proposed in [16].

First of all, we want to notice that the error estimator Δ_N^t should depend also on the control function g. Then we have to choose a particular control function with which the greedy algorithm is performed. Recalling that our problem is linear, for every input control function g we can recover the solution $u^{\mathcal{N}}(\boldsymbol{\mu})$ by convolution

$$u_{\tilde{j}}^{\mathcal{N}}(\boldsymbol{\mu}) = \sum_{j=1}^{\tilde{j}} g(t^j)\tilde{u}_{\tilde{j}-j}^{\mathcal{N}}(\boldsymbol{\mu}) \tag{A.1}$$

where $\tilde{u}^{\mathcal{N}}(\boldsymbol{\mu})$ is the *impulse response*, that is the solution of (3.9) in which it is used an *impulse control* $\tilde{g}$, such that:

$$\tilde{g}(t^0) = 1, \qquad \tilde{g}(t^j) = 0 \qquad 1 \leq j \leq J. \tag{A.2}$$

It is important to note, in (A.1), that the function $u_{\tilde{j}}^{\mathcal{N}}$, $j = \ldots, J$, is a linear combination of the impulse response $\tilde{u}_j^{\mathcal{N}}$, $j = \ldots, J$. This means that, to obtain a good approximation of the solution corresponding to any control function g, it is sufficient that the RB method approximates well the (parametric) impulse response [16].
As a consequence of the presence of an "unknown" control function that can be set Online, theoretically the error $e(\boldsymbol{\mu})$ should depend also on g. We must point out that the following inequality is not valid:

$$|||e(\boldsymbol{\mu}; g)||_{t-dep} \leq \varepsilon_{tol}^* \quad \forall \boldsymbol{\mu} \in \Xi_{train} \tag{A.3}$$

for every choice of g, but we can provide error estimators such that

$$|||e(\boldsymbol{\mu}; g)||_{t-dep} \leq \Delta_N^t(\boldsymbol{\mu}; g) \quad \forall \boldsymbol{\mu} \in \Xi_{train}, \forall g. \tag{A.4}$$

A posteriori error estimates

We are dealing now with the a posteriori error estimators to be used in the greedy algorithm. In our work we will follow the choice presented in [16], but other possibility have been proposed [54].

The first ingredient we have to introduce is the dual norm of the residual:

$$\varepsilon_N(t^j;\boldsymbol{\mu};g) := \sup_{v^{\mathcal{N}}\in X^{\mathcal{N}}} \frac{r_N(v^{\mathcal{N}};t^j;\boldsymbol{\mu};g)}{||v^{\mathcal{N}}||_X}, \quad 1\le j\le J, \tag{A.5}$$

where r_N is the residual of the RB approximation, that is:

$$\begin{aligned} r_N(v^{\mathcal{N}};t^j;\boldsymbol{\mu};g) = g(t^j)f(v^{\mathcal{N}}) - \frac{1}{\Delta t}m(u^{\mathcal{N}}_{j,N}-u^{\mathcal{N}}_{j-1,N},v^{\mathcal{N}};\boldsymbol{\mu}) - a(u^{\mathcal{N}}_{j,N},v^{\mathcal{N}};\boldsymbol{\mu}) \\ \forall v^{\mathcal{N}}\in X^{\mathcal{N}},\ 1\le j\le J. \end{aligned} \tag{A.6}$$

Let us define $\mathcal{F}_q,\mathcal{A}^n_q,\mathcal{M}^n_q \in X^{\mathcal{N}}$ respectively for $q=1,\dots,Q_F$, $q=1,\dots,Q_a$, $n=1,\dots,N$ and $q=1,\dots,Q_m$, $n=1,\dots,N$ as following:

$$\begin{aligned} (\mathcal{F}_q,v^{\mathcal{N}})_X &= F^q(v^{\mathcal{N}}) \quad \forall v\in X^{\mathcal{N}} \\ (\mathcal{A}^n_q,v^{\mathcal{N}})_X &= a^q(\zeta^{\mathcal{N}}_n,v^{\mathcal{N}}) \quad \forall v\in X^{\mathcal{N}} \\ (\mathcal{M}^n_q,v^{\mathcal{N}})_X &= m^q(\zeta^{\mathcal{N}}_n,v^{\mathcal{N}}) \quad \forall v\in X^{\mathcal{N}}. \end{aligned} \tag{A.7}$$

All these quantities are $\boldsymbol{\mu}$-independent. Now, we need to define some scalar products as follows:

$$\begin{aligned} &\Lambda^{FF}_{pq} = (\mathcal{B}_p,\mathcal{B}_q)_X, && 1\le p,q\le Q_F; \\ &\Lambda^{aF}_{pqn} = -2(\mathcal{B}_p,\mathcal{A}^n_q)_X, && 1\le p\le Q_F,\ 1\le p\le Q_a,\ 1\le n\le N; \\ &\Lambda^{MF}_{pqn} = -\frac{2}{\Delta t}(\mathcal{B}_p,\mathcal{M}^n_q)_X, && 1\le p\le Q_F,\ 1\le p\le Q_m,\ 1\le n\le N; \\ &\Lambda^{aa}_{pnqm} = (\mathcal{A}^n_p,\mathcal{A}^m_q)_X, && 1\le p,q\le Q_a,\ 1\le n,m\le N; \\ &\Lambda^{aM}_{pnqm} = \frac{2}{\Delta t}(\mathcal{A}^n_p,\mathcal{M}^m_q)_X, && 1\le p\le Q_a,\ 1\le q\le Q_m,\ 1\le n,m\le N; \\ &\Lambda^{MM}_{pnqm} = \frac{1}{\Delta t^2}(\mathcal{M}^n_p,\mathcal{M}^m_q)_X, && 1\le p,q\le Q_m,\ 1\le n,m\le N. \end{aligned}$$

Finally, we can compute, thanks to affine assumptions (1.9), (1.10) and (3.3), the residual

squared [16]:

$$\begin{aligned}
\varepsilon_N(t^j;\boldsymbol{\mu};g)^2 = &\sum_{p,q=1}^{Q_F}\Theta_F^p(\boldsymbol{\mu})\Theta_F^q(\boldsymbol{\mu})g(t^j)^2\Lambda_{pq}^{FF}+\\
&+\sum_{p=1}^{Q_F}\sum_{n=1}^{N}\Theta_F^p(\boldsymbol{\mu})g(t^j)\left(\sum_{q=1}^{Q_a}\Theta_a^q(\boldsymbol{\mu})u_{N,n,j}^{\mathcal{N}}(\boldsymbol{\mu})\Lambda_{pqn}^{aF}+\right.\\
&\left.+\sum_{q=1}^{Q_m}\Theta_M^q(\boldsymbol{\mu})\left(u_{N,n,j}^{\mathcal{N}}(\boldsymbol{\mu})-u_{N,n,j-1}^{\mathcal{N}}(\boldsymbol{\mu})\right)\Lambda_{pqn}^{MF}\right)+\\
&+\sum_{n,m=1}^{N}\left\{\sum_{p,q=1}^{Q_a}\Theta_a^p(\boldsymbol{\mu})\Theta_a^q(\boldsymbol{\mu})u_{N,m,j}^{\mathcal{N}}(\boldsymbol{\mu})u_{N,n,j}^{\mathcal{N}}(\boldsymbol{\mu})\Lambda_{pmqn}^{aa}+\right.\\
&+\sum_{p,q=1}^{Q_m}\Theta_M^p(\boldsymbol{\mu})\Theta_M^q(\boldsymbol{\mu})\Lambda_{pmqn}^{MM}\cdot\\
&\quad\cdot(u_{N,m,j}^{\mathcal{N}}(\boldsymbol{\mu})-u_{N,m,j-1}^{\mathcal{N}}(\boldsymbol{\mu}))(u_{N,n,j}^{\mathcal{N}}(\boldsymbol{\mu})-u_{N,n,j-1}^{\mathcal{N}}(\boldsymbol{\mu}))+\\
&\left.+\sum_{p=1}^{Q_a}\sum_{q=1}^{Q_m}\Theta_a^p(\boldsymbol{\mu})\Theta_M^q(\boldsymbol{\mu})u_{N,m,j}^{\mathcal{N}}(\boldsymbol{\mu})(u_{N,n,j}^{\mathcal{N}}(\boldsymbol{\mu})-u_{N,n,j-1}^{\mathcal{N}}(\boldsymbol{\mu}))\Lambda_{pmqn}^{aM}\right\}.
\end{aligned}\tag{A.8}$$

We then need again a lower bound $\boldsymbol{\mu}\mapsto\alpha_{LB}^{\mathcal{N}}(\boldsymbol{\mu})$ for the coercivity constant of the discretized bilinear form a as in (1.52). To do so we can resort to the SCM [23] introduced in section 1.2.5 and used in the steady case.

After these preliminaries, we are finally able to define the *a posteriori* error estimator which satisfies (3.29):

$$\Delta_N^t(\boldsymbol{\mu};g)=\left(\frac{\Delta t}{\alpha_{LB}^{\mathcal{N}}(\boldsymbol{\mu})}\sum_{j=1}^{J}\varepsilon_N(t^j;\boldsymbol{\mu};g)^2\right)^{\frac{1}{2}}.\tag{A.9}$$

The *a posteriori* error estimator used during the greedy is actually

$$\Delta_N^t(\boldsymbol{\mu}):=\Delta_N^t(\boldsymbol{\mu};\tilde{g}),\tag{A.10}$$

where $\tilde{g}$ defined in (A.1).

Bibliography

[1] F. Ballarin. *Reduced-order models for patient-specific haemodynamics of coronary artery bypass grafts*. PhD thesis, Politecnico di Milano, 2015.

[2] F. Ballarin, E. Faggiano, S. Ippolito, A. Manzoni, A. Quarteroni, G. Rozza, and R. Scrofani. Fast simulations of patient-specific haemodynamics of coronary artery bypass grafts based on a POD–Galerkin method and a vascular shape parametrization. *Journal of Computational Physics*, 315:609 – 628, 2016.

[3] F. Ballarin, A. Sartori, and G. Rozza. RBniCS - reduced order modelling in FEniCS. `http://mathlab.sissa.it/rbnics`, 2015.

[4] M. Barrault, Y. Maday, N.C. Nguyen, and A.T. Patera. An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. *C. R. Math. Acad. Sci.*, 339(9):667–672, 2004.

[5] A.N. Brooks and T.J.R. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Engrg.*, 32(1-3):199–259, 1982.

[6] P. Chen. *Model Order Reduction Techniques for Uncertainty Quantification Problems*. PhD thesis, École polytechnique fédérale de Lausanne EPFL, 2014.

[7] P. Chen, A. Quarteroni, and G. Rozza. A weighted reduced basis method for elliptic partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 51(6):3163–3185, 2013.

[8] Y. Chen, J. S. Hesthaven, Y. Maday, and J. Rodríguez. Improved successive constraint method based a posteriori error estimate for reduced basis approximation of 2D Maxwell's problem. *M2AN Math. Model. Numer. Anal.*, 43(6):1099–1116, 2009.

[9] L. Dedè. Reduced basis method for parametrized elliptic advection-reaction problems. *J. Comput. Math.*, 28(1):122–148, 2010.

[10] J.W. Demmel. *Applied numerical linear algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphi, PA, 1997.

[11] R. Durrett. *Probability Theory and Examples*. Cambridge University Press, Cambridge, UK, 2010.

[12] J.L. Eftang, M.A. Grepl, and A.T. Patera. A posteriori error bounds for the empirical interpolation method. *C. R. Math. Acad. Sci.*, 51(1):28–58, 2010.

[13] L.P. Franca, S.L. Frey, and T.J.R. Hughes. Stabilized finite element methods. I. Application to the advective-diffusive model. *Comput. Methods Appl. Mech. Engrg.*, 95(2), 1992.

[14] F. Gelsomino and G. Rozza. Comparison and combination of reduced-order modelling techniques in 3D parametrized heat transfer problems. *Math. Comput. Model. Dyn. Syst.*, 17(4):371–394, 2011.

[15] M.A. Grepl, Y. Maday, N.C. Nguyen, and A.T. Patera. Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. *M2AN Math. Model. Numer. Anal.*, 41(3):575–605, 2007.

[16] M.A. Grepl and A.T. Patera. A Posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations. *M2AN Math. Model. Numer. Anal.*, 1(39):157–181, 2005.

[17] Q. Guan, M. Gunzburger, C. G. Webster, and G. Zhang. Reduced basis methods for nonlocal diffusion problems with random input data. Technical report, Elsevier, 2016.

[18] Haasdonk, B. and Ohlberger, M. Reduced basis method for finite volume approximations of parametrized linear evolution equations. *ESAIM: M2AN*, 42(2):277–302, 2008.

[19] J.S. Hesthaven, G. Rozza, and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer, 2016.

[20] T.J.R. Hughes and A.N. Brooks. A multidimensional upwind scheme with no crosswind diffusion. In *Finite element methods for convection dominated flows*, volume 34, pages 19–35. Am. Soc. Mech. Engrs, New York, 1979.

[21] T.J.R. Hughes, L. P. Franca, and G. M. Hulbert. A new finite element formulation for computational fluid dynamics. VIII. The Galerkin/least-squares method for advective–diffusive equations. *Comput. Methods Appl. Mech. Engrg.*, 73(2), 1989.

[22] D.B.P. Huynh, D.J. Knezevic, Y. Chen, J.S. Hesthaven, and A.T. Patera. A natural-norm successive constraint method for inf-sup lower bounds. *Comput. Methods Appl. Mech. Engrg.*, 199(29-32):1963–1975, 2010.

[23] D.B.P. Huynh, G. Rozza, S. Sen, and A.T. Patera. A successive constraint linear optimization method for lower bounds of parametric coercivity and inf-sup stability constants. *C. R. Math. Acad. Sci.*, 345(8):473–478, 2007.

[24] L. Iapichino, A. Quarteroni, and G. Rozza. A reduced basis hybrid method for coupling of parametrized domains represented by fluidic networks. *Comput. Methods Appl. Mech. Engrg.*, 221-222:63–82, 2012.

[25] F.P. Incropera and D.P. DeWitt. *Fundamental of Heat and Mass Transfer*. John Wiley & Sons, 1990.

[26] K. Ito and S.S. Ravindran. A reduced-order method for simulation and control of fluid flows. *J. Comput. Phys.*, 2(143):403–425, 1998.

[27] V. John and J. Novo. Error analysis of the SUPG finite element discretization of evolutionary convection-diffusion-reaction equations. *SIAM J. Numer. Anal.*, 49(3):1149–1176, June 2011.

[28] C. Johnson and U. Nävert. An analysis of some finite element methods for advection-diffusion problems. In *Analytical and numerical approaches to asymptotic problems in analysis (Proc. Conf., Univ. Nijmegen, Nijmegen, 1980)*, volume 47 of *North-Holland Math. Stud.*, pages 99–116, North-Holland, Amsterdam, 1981.

[29] C. Johnson, U. Nävert, and J. Pitkäranta. Finite element methods for linear hyperbolic problems. *Comput. Methods Appl. Mech. Engrg.*, 45(1-3), 1984.

[30] I.T. Jolliffe. *Principal Component Analysis*. Springer New York, 2002.

[31] K. Kunisch and S. Volkein. Galerkin proper orthogonal decomposition methods for parabolic problems. *Numer. Math.*, 90(1):117–148, 2001.

[32] T. Lassila, A. Manzoni, and G. Rozza. On the approximation of stability factors for general parametrized partial differential equations with a two-level affine decomposition. *M2AN Math. Model. Numer. Anal.*, 46:1555–1576, 2012.

[33] T. Lassila and G. Rozza. Parametric free-form shape design with PDE models and reduced basis method. *Comput. Methods Appl Mech. Engrg.*, 199:1583–1592, 2010.

[34] A.E. Løvgren, Y. Maday, and E.M. Rønquist. *The reduced basis element method for fluid flows*, page 129–154. Adv. Math. Fluid Mech. Birkhäuser, Basel, 2007.

[35] Y. Maday, A. Manzoni, and A. Quarteroni. An online intrinsic stabilization strategy for the reduced basis approximation of parametrized advection-dominated problems. Technical report, MATHISCE, 2016.

[36] A. Manzoni, A. Quarteroni, and G. Rozza. Model reduction techniques for fast blood flow simulation in parametrized geometries. *Int. J. Numer. Meth. Biomed. Engng.*, 28:604–625, 2012.

[37] A. Manzoni, A. Quarteroni, and G. Rozza. Shape optimization for viscous flows by reduced basis methods and free-form deformation. *Int. J. Numer. Meth. Fluids*, 70:646–670, 2012.

[38] A. Manzoni and G. Rozza. Model order reduction by geometrical parametrization for shape optimization in computational fluid dynamics. In *Proceedings of the V European Conf. Computat. Fluid Dynamics*, Lisbon, Portugal, June 14-17 2010. ECCOMAS, CFD 2010, J.C.F. Pereira and A. Sequeira editors.

[39] N. C. Nguyen, G. Rozza, D. B. P. Huynh, and A. T. Patera. *Reduced Basis Approximation and a Posteriori Error Estimation for Parametrized Parabolic PDEs: Application to Real-Time Bayesian Parameter Estimation*, pages 151–177. John Wiley & Sons, Ltd, 2010.

[40] P. Pacciarini. Stabilized reduced basis method for parametrized advection-diffusion PDEs. Master's thesis, Università degli Studi di Pavia, 2012.

[41] P. Pacciarini and G. Rozza. Stabilized reduced basis method for parametrized advection–diffusion PDEs. *Comput. Methods Appl. Mech. Engrg.*, 274:1–18, 2014.

[42] P. Pacciarini and G. Rozza. Stabilized reduced basis method for parametrized scalar advection-diffusion problems at higher Péclet number: roles of the boundary layers and inner fronts. In *Proceedings of the jointly organized 11th World Congress on Computational Mechanics - WCCM XI, 5th European Congress on Computational Mechanics - ECCM V, 6th European Congress on Computational Fluid Dynamics - ECFD V*, pages 5614–5624, 2014.

[43] P. Pacciarini and G. Rozza. *Reduced Basis Approximation of Parametrized Advection-Diffusion PDEs with High Péclet Number*, pages 419–426. Springer International Publishing, Cham, 2015.

[44] T.A. Porsching. Estimation of the error in the reduced basis method solution of nonlinear equations. *Math. Comp.*, 45(172):487–496, 1985.

[45] A. Quarteroni. *Numerical models for differential problems*, volume 8 of *MS&A*. Springer-Verlag Italia, Milano, 2013.

[46] A. Quarteroni, G. Rozza, L. Dedè, and A. Quaini. Numerical approximation of a control problem for advection-diffusion processes. In *System modeling and optimization*, volume 199 in IFIP Int. Fed. Inf. Process., page 261–273, New York, 2006. Springer.

[47] A. Quarteroni, G. Rozza, and A. Manzoni. Certified reduced basis approximation for parametrized partial differential equations and applications. *J. Math. Ind.*, 1(3):1–44, 2011.

[48] A. Quarteroni, R. Sacco, and F Saleri. *Numerical mathematics*, volume 37 of *Texts in Applied Mathematics*. Springer-Verlag, Berlin, 2007.

[49] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations*. Springer, 1994.

[50] G. Rozza, D.B.P. Huynh, N.C. Nguyen, and A.T. Patera. Real-time reliable simulation of heat transfer phenomena. In *ASME -American Society of Mechanical Engineers - Heat Transfer Summer Conference Proceedings*, S. Francisco, CA, USA, 2009.

[51] G. Rozza, D.B.P. Huynh, and A.T. Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations: application to transport and continuum mechanics. *Arch. Comput. Methods Eng.*, 3(15):229–275, 2008.

[52] G. Rozza, N.C. Nguyen, A.T. Patera, and S. Deparis. Reduced basis methods and a posteriori error estimators for heat transfer problems. In *ASME -American Society of Mechanical Engineers - Heat Transfer Summer Conference Proceedings*, S. Francisco, CA, USA, 2009.

[53] G. Rozza and K. Veroy. On the stability of the reduced basis method for Stokes equations in parametrized domains. *Comput. Methods Appl. Mech. Engrg.*, 196(7):1244–1260, 2007.

[54] K. Urban and A.T. Patera. A new error bound for reduced basis approximation of parabolic partial differential equations. *C. R. Math. Acad. Sci. Paris*, 350(3-4):203–207, 2012.

[55] L. Venturi. Weighted reduced basis methods for parametrized PDEs in uncertainty quantification problems. Master's thesis, Università degli Studi di Trieste, Trieste, Italia, 2016.