Contributions to Real-Time Reliable Simulations and Certain Aspects of Scientific Computing

Christophe Prud'Homme

Université Pierre et Marie Curie, Paris VI **École doctorale des Sciences**
Mathématiques de Paris Centre
UFR 929

Contributions to Real-Time Reliable Simulations and Certain Aspects of Scientific Computing

MÉMOIRE

presented and defended publicly on December 9 2005

for the authorization of

Habilitation à Diriger des Recherches
de l'Université Pierre et Marie Curie – Paris VI
(Spécialité Mathématiques Appliquées)

par

Christophe Prud'homme

Composition du jury

Rapporteurs :	Charles-Henri Bruneau
	Andreas Griewank
	Spencer Sherwin
Examinateurs :	Silvia Bertoluzza
	Stéphane Cordier
	Yvon Maday
	Bertrand Maury
	Olivier Pironneau

Laboratoire Jacques Louis Lions — UMR 7598

Mis en page avec la classe thloria.

Remerciements

La vie est faite de rencontres et d'opportunités. À y bien réfléchir, j'ai été très bien servi de ce point de vue. Tout d'abord la rencontre avec Olivier Pironneau et le calcul scientifique: c'est en lisant la description de son cours que j'ai immédiatement changé d'orientation pendant mes études et si je suis ici aujourd'hui à défendre mon habilitation. Il y a ensuite Yvon Maday qui m'a embarqué dans l'aventure CEMRACS des premières années. Puis il y a ce jour de mois de mai ou juin 1999 où Olivier en me croisant me dit que Tony Patera est là, à Paris, et qu'il a peut être un post-doc à proposer. Encore une fois, je ne crois pas m'être donné le temps de la reflexion et je me suis retrouvé pour une brève discussion avec Tony puis un ou deux emails plus tard, je me suis retrouvé au MIT fin 1999 dans son groupe. Enfin, en juin 2003, lors de mon séjour au laboratoire Jacques-Louis Lions, Yvon me parle d'un poste chez Alfio Quarteroni à l'EPFL. Après un séminaire et quelques discussions, je me retrouve en poste à Lausanne dans le groupe d'Alfio. Au passage en Février 2003, Yvon me propose de postuler au concours du ministère de la recherche pour la création d'entreprises innovantes avec la technologie que j'avais commencée à développer au MIT autour des bases réduites. Au mois de juillet 2003, elle fut récompenser.

À chaque époque de ma carrière, j'ai eu la chance de pouvoir m'impliquer dans divers projets autour du calcul scientifique: mon sujet de thèse et FreeFEM avec Olivier, le CEMRACS avec Yvon, les bases réduites avec Tony et maintenant LifeV avec Alfio. Je tiens ici à tous les remercier.

Les travaux de cette habilitation ont essentiellement été conduits au MIT dans le groupe de Tony. À plusieurs reprises aux États-Unis, j'avais entendu parler de la "MIT experience", je crois avoir effectivement vécu une expérience scientifique et humaine très marquante, extrêmement positive et enrichissante au contact de Tony en particulier et du MIT en général.

Je remercie les rapporteurs de mon habilitation Charles-Henri Bruneau, Andreas Griewank et Spencer Sherwin qui ont accepté de rapporter dans des délais très courts. Je tiens également à remercier Silvia Bertoluzza, Stéphane Cordier et Bertrand Maury qui ont accepté chaleureusement de faire partie du jury.

J'aimerais adresser une pensée amicale aux différents collaborateurs que j'ai côtoyés ces dernières années, que ce soit au MIT, Martin Grepl, Thomas Leurent, Ivan Oliveira, Dimitrios Rovas, Karen Veroy, Yuri Solodukhov et surtout Debbie; ou à l'EPFL Erik Burman, Gilles Fourestey, Nicola Parolini et Christoph Winkelmann.

Je tiens à remercier ma famille pour son soutien constant et surtout Evelyn sans qui ces dernières années auraient été bien ternes et qui a toujours su trouver les mots justes d'encouragements. Elle a également accompli la tâche ingrate mais au combien indispensable de relire ce mémoire aussi bien dans sa version anglaise que française et ceci bien qu'elle vienne d'un monde complètement différent des mathématiques, un grand merci pour ton abnégation.

À ma famille
À Evelyn

Résumé

Ce document présente une synthèse des travaux de recherche effectués par l'auteur depuis 2000 au Massachusetts Institute Of Technology(MIT) puis à l'École Polytechnique Fédérale de Lausanne(EPFL). Ces travaux sont essentiellement axés sur les méthodes de bases réduites avec bornes d'erreur permettant la prédiction rapide et fiable de quantités issues de solutions d'équations aux dérivées partielles. L'implémentation de ces méthodes est complexe et l'auteur a également proposé et développé des outils innovants afin de tirer profit de celles-ci à moindre coût en termes de développement. Par ailleurs, ce document présente quelques travaux en cours et perspectives de recherche.

Abstract

This document presents a synthesis of the research work done by the author since 2000 at Massachusetts Institute Of Technology(MIT) and at École Polytechnique Fédérale de Lausanne(EPFL). This work is essentially focused on the reduced basis output bounds methods allowing the rapid yet reliable prediction of quantities derived from the solutions of partial differential equations. The implementation of these methods is complex and the author has also proposed and implemented innovating tools in order to use these methods at a low development costs. Furthermore, some recent investigations are presented and research perspectives discussed.

Introduction

The work which will be presented in this document is primarily justified by the current evolution and needs for the scientific computation in sciences and in particular in engineering.

First, we can observe an increasing demand[1] for computations with real-time response, for example in education, engineering design and control systems: indeed simulations integrated in components or electronic documents will only be interesting if the interaction is quasi instantaneous[2]. There is a lot of work from the scientific computing community to increase the performances of simulation techniques. However, the algorithms and paradigms for real-time computations are only appearing. One of my contributions was to develop in a more systematic way mathematical methods for rapid but reliable predictions for a wide class of problems.

Then, we can also observe a growing trend that uncertainty must be quantified if we effectively want to trust the results of numerical computations; it becomes all the more critical when we ask integrated interfaces to be accessible by non-specialists while requiring real-time responses. The later implies or suggests the diminution of the operation counts and degrees of freedom which can be conceived only if the error estimation is an integral part of the eventually adaptive prediction process. Similarly to the very rapid prediction methods, general frameworks for error estimation have been proposed recently and are actively developed. A second contribution of these research activities was to develop *a posteriori* error estimations of outputs of interest, in engineering and in the context of real-time simulations, which means that the error estimations are also available in real-time.

Finally, distributed and parallel computing as well as the Web have seen many new developments in various scientific domains. We can cite for example the projects such as Seti@home[3], Folding@Home[4], Grid.org[5], integrals.wolfram.com[6], Matlab and its Web server[7] and many other commercial or free environments. Moreover, new infrastructures for distributed and parallel computing are becoming standards such as Corba[8], Globus(Grid)[9], Condor[10] or Netsolve[11]. It is clear that from now on with the set of software available, the control algorithms, the data and computations can be disseminated on many clients and servers. Another objective of the research activity was to design and develop an original and cheap infrastructure that allows very simple interfaces for real-time computations requests through a network. Besides among the possible interfaces, we focused on electronic documents which replace more and

[1] A simple check of the major research themes will confirm this, but I also got the confirmation that industrials are indeed interested through a market study.

[2] The definition of quasi instantaneous is fuzzy here, we can distinguish two kinds of real-time : *(i) hard* real-time which puts strict constraints on the execution time of a command and *(ii)* the *soft* real-time for multimedia applications where temporal constraints are not enforced but a delay in the later would create a discomfort.

[3] http://setiathome.ssl.berkeley.edu/

[4] http://folding.stanford.edu/

[5] http://www.grid.org/

[6] http://integrals.wolfram.com/

[7] http://www.mathworks.com/products/webserver/

[8] http://www.omg.org/

[9] http://www.globus.org/

[10] http://www.cs.wisc.edu/condor/

[11] http://icl.cs.utk.edu/netsolve/

more standard archiving methods. It is indeed clear that electronic media offer new capabilities, in our case integrate text and computations. There are already applications in this area such as Mathcad, Mathematica or Matlab which have all been developing, in proprietary formats and so with restricted access, documents with integrated computations. This type of application is not yet frequently used or used as a standard. We have developed a set of tools — uniquely based on free software — allowing to write aesthetic documents where equations are *actionable* and which allow a user in a natural context to request and present real-time computations.

In this synthesis, the research work accomplished since 2000 while being at MIT in the group of A.T. Patera until now at EPFL in the group of A. Quarteroni will be presented in the coming sections. The main part of the research was done within the group of A.T. Patera. At the end of my time at MIT and during a few month in the laboratory Jacques Louis Lions in Paris, I have started the development of a technology which has received the price from the French Ministry of Research in an innovation contest in 2003 in the category "emergence". Since then, the technology has evolved and a market study permitted to confirm the interest of the industry for such a methodology — the reduced basis output bounds methods — and technology and to identify the sectors with strong potential. This has also permitted to orient many research activities: identification and development of generic components, adaptive algorithms, systematic comparison with similar methods. These activities will be eventually tested with the European project (STREP) Excalade[12].

The document is organized in four parts. In the first part, I present the synthesis of my research work which is divided in three chapters: the first one talks about the reduced basis output bounds methods. The second one presents some work around the programming tools that have been developped for these methods and the distributed environment in which they reside. And finally the third chapter briefly exposes some projects and research perspectives regarding the reduced basis. The next two parts contain the articles I contributed to : they are organized according to the order of the first two chapters of the synthesis. And finally, in the fourth part, I inserted my bibliography and my curriculum vitae. I will present only very few numerical results or experiments, albeit there are many of them in the publications, they mostly corroborate and illustrate the theoritical results.

[12]Flexible, Competitive and Agile Design of Low-Noise-Emission Aircraft in a Distributed Environment. Enregistré auprès de la commission européenne sous le numéro: FP6-031016

Contents

Part I

Research Work Synthesis

Chapter 1

Reduced Basis Methods

The components in mechanical engineering are becoming more and more complex (in topology and structure) and multi-functional : indeed, it is only through complexity and multi-functionality that we can satisfy the stringent performance constraints in many domains such as for example defense, medicine or embedded systems. However, these complex multi-functional systems admit no longer an intuitive compromise analysis: we must turn to optimization — optimal choice of material, configuration and deployment — to truly benefit from these new structures.

The optimization, the control as well as the characterization of these components necessitate the prediction of certain quantities of interest that we shall call outputs — for example displacements, mean temperatures, heat transfer or drag and lift. — These outputs are typically expressed as functionals of variables associated with partial differential equations which describe the mechanical behavior of the component or the system. The parameters, which we shall call inputs, serve to identify a particular *configuration*: these inputs can represent the design such as the geometry during the optimization analysis; control variables such as the power of a fan, for example in the context of real-time applications; or characterization variables such as physical properties. It follows then an *input/output* relationship whose evaluation requires the resolution of the underlying partial differential equations.

The methodology, see [62, 81, 80, 61], has two main components : a *very rapid* prediction, yet *reliable* thanks to *a posteriori* error estimation, of the behavior of the quantities of interest — the *forward* problem; — and very rapid mathematical programming techniques — the *inverse* problem.

Regarding the *forward* problem, the approach comprises three ingredients: (*i*) rapidly convergent reduced basis approximations — projection on a space W_N spanned by the solutions of parametrized partial differential equations selected at N points in the parameter space; (*ii*) *a posteriori* error estimations of the approximation by relaxing the error equation ; (*iii*) offline/online algorithms — methods allowing to decouple the generation and the projection of the approximation process and error estimation. As to the *inverse* problem, the approach is based on interior point or sequential quadratic programming methods for optimization under constraints — techniques which provide rapid convergence toward an optimal admissible solution. — These methods are described in more details in [62, 52, 60].

For a restricted class of problems, affinely or quasi-affinely parametrized partial differential equations, the algorithmic complexity of the *online* stage of the *forward* prediction depends

uniquely on N (typically small and less than 100) and on the parametric complexity of the problem which provide thus a real-time reliable prediction. Moreover, the rapid convergence of the optimization algorithm allows to preserve to some extent the real-time responses in the context of the *inverse* problem; as to the error bounds, they guarantee not only optimal but also admissible and safe results.

We shall see, in the coming section, the development of these methods — more precisely the development of the three preceding components: reduced basis approximation, error estimation and affine decomposition — for certain classes of equations: *(i)* elliptic symmetric coercive and *compliant* — the output functional coincides with the right hand side functional of the variational equation, — *(ii)* elliptic non-symmetric and/or non-compliant — the output functional does not coincide with the right hand side functional of the variational equation, — *(iii)* generalized eigenvalue problems, *(iv)* linear elliptic non-coercive, *(v)* non-linear, and finally *(vi)* approximately parametrized equations.

I present first with few details the methods in the case *(i)* then for each following type of equations, I indicate only the crucial ingredients or results. Finally I present a more recent work: a few indications on the generation of the reduced basis space and an *a priori* convergence result for the multi-dimensional reduced basis approximation.

1.1 Abstract Framework

We consider a regular domain $\Omega \subset \mathbb{R}^d$, $d = 1,\ 2$, or 3, and associated functional space $X \subset H^1(\Omega)$, where $H^1(\Omega) = \{v \in L^2(\Omega),\ \nabla v \in (L^2(\Omega))^d\}$, and $L^2(\Omega)$ is the space of square integrable function over Ω. The scalar product and norm associated to X are given by $(\cdot,\cdot)_X$ and $\|\cdot\|_X = (\cdot,\cdot)^{1/2}$, respectively. We define also a set of parameters $\mathcal{D} \in \mathbb{R}^P$, whereof a particulier point is denoted μ. We shall note that Ω does not depend on the parameters in the sense that Ω is not parametrized topologically.

We introduce now a bilinear form $a\colon\ X \times X \times \mathcal{D} \to \mathbb{R}$, and the linear forms $f\colon\ X \to \mathbb{R}$, $\ell\colon\ X \to \mathbb{R}$. We suppose that a is continuous, $a(w,v;\mu) \leq \gamma(\mu)\ \|w\|_X\ \|v\|_X \leq \gamma_0\ \|w\|_X\ \|v\|_X,\ \forall \mu \in \mathcal{D}$.

We require also the linear forms f and ℓ to be bounded; in what follows, we suppose that the output is compliant, $f(v) = \ell(v),\ \forall\, v \in X$ — we extend later the results to the non-compliant case.

We also suppose that a, f and ℓ depend affinely on the parameters. In particular, we suppose that for a finite integer Q, a can be expressed as

$$a(w,v;\mu)\ =\ \sum_{q=1}^{Q}\ \sigma^q(\mu)\ a^q(w,v), \quad \forall\ w,v \in X,\ \forall\ \mu \in \mathcal{D}, \tag{1.1}$$

with $\sigma^q\colon\ \mathcal{D} \to \mathbb{R}$ and $a^q\colon\ X \times X \to \mathbb{R}$, $q = 1,\ldots,Q$. Similarly for f and l although, in what follows, they won't depend on μ. This hypothesis on a, f and l is crucial to obtain real-time predictions. A large class of problem admits such a decomposition, and this restriction has been alleviated recently for certain non-affine and locally non-affine parametrizations.

Our problem under abstract form reads then as follows: for all $\mu \in \mathcal{D}$, find $s(\mu) \in \mathbb{R}$ given by

$$s(\mu) = \ell(u(\mu)), \tag{1.2}$$

where $u(\mu) \in X$ is solution of

$$a(u(\mu), v; \mu) \;=\; f(v), \quad \forall\, v \in X. \tag{1.3}$$

In the language of the introduction, a is the partial differential equation under weak form, $u(\mu)$ is our solution and $s(\mu)$ the output.

1.2 Symmetric Coercive Equations and Compliant Outputs

Publication: [62].

In this section, we suppose that a is coercive — we shall present results in the case where a is not coercive in section 1.5, —

$$0 < \alpha_0 \leq \;\alpha(\mu) = \inf_{w \in X} \frac{a(w, w; \mu)}{\|w\|_X^2}, \quad \forall\, \mu \in \mathcal{D}, \tag{1.4}$$

and symmetric, $a(w, v; \mu) = a(v, w; \mu)$, $\forall\, w, v \in X$, $\forall\, \mu \in \mathcal{D}$.

1.2.1 Reduced Basis Approximation

We introduce the sampling of the parameter space $S_N = \{\mu_1, \ldots, \mu_N, \mu_i \in \mathcal{D}\}$. The construction of this sampling can be done in several ways, we shall see some possible options in section 1.8.

Then we construct the reduced basis space $W_N = \text{span}\,\{\zeta_n \equiv u(\mu_n),\; n = 1, \ldots, N\}$, where $u(\mu_n) \in X$ is the solution of (1.3) for $\mu = \mu_n$. In practice $u(\mu_n)$ is replaced by a *truth* finite element approximation such that we can equate the finite element solution with the exact one.

The reduced basis approximation reads then as follows: for all $\mu \in \mathcal{D}$, find $s_N(\mu) = \ell(u_N(\mu))$, où $u_N(\mu) \in W_N$ is solution of

$$a(u_N(\mu), v; \mu) \;=\; \ell(v), \quad \forall\, v \in W_N. \tag{1.5}$$

A priori convergence properties will be presented in section 1.9. We refer to the tables 4.1 and 4.2 for numerical evidence that the reduced basis approximation is converging very fast.

1.2.2 Computational Procedure

Empirical and theoritical results, see section 1.9, suggest that N can effectively be chosen small — in practice $N = O(10)$. — We develop then two procedures for offline and online computations which shall exploit the reduction in the number of degrees of freedom.

We express naturally $u_N(\mu)$ in the reduced basis as

$$u_N(\mu) = \sum_{j=1}^{N} u_{N\,j}(\mu)\, \zeta_j = (\underline{u}_N(\mu))^T\, \underline{\zeta}, \tag{1.6}$$

where $\underline{u}_N(\mu) \in \mathbb{R}^N$; we choose the test functions as $v = \zeta_i$, $i = 1, \ldots, N$. We then obtain the algebraic representation for $\underline{u}_N(\mu) \in \mathbb{R}^N$,

$$\underline{A}_N(\mu)\, \underline{u}_N(\mu) \;=\; \underline{F}_N, \tag{1.7}$$

and we evaluate then our output as $s_N(\mu) = \underline{F}_N^T\, \underline{u}_N(\mu)$. In this case, $\underline{A}_N(\mu) \in \mathbb{R}^{N\times N}$ is a symmetric positive definite matrix with its entries as $A_{N\,i,j}(\mu) \equiv a(\zeta_j, \zeta_i; \mu)$, $1 \le i, j \le N$, and $\underline{F}_N \in \mathbb{R}^N$ has its entries as $F_{N\,i} \equiv f(\zeta_i)$, $i = 1, \ldots, N$.

We now use the affine decomposition (1.1) to write

$$A_{N\,i,j}(\mu) = a(\zeta_j, \zeta_i; \mu) = \sum_{q=1}^{Q} \sigma^q(\mu)\, a^q(\zeta_j, \zeta_i)\ , \tag{1.8}$$

or equivalently

$$\underline{A}_N(\mu) = \sum_{q=1}^{Q} \sigma^q(\mu)\, \underline{A}_N^q\ ,$$

where $\underline{A}_N \in \mathbb{R}^{N\times N}$ are given by $A^q_{N\,i,j} = a^q(\zeta_j, \zeta_i)$, $i \le i, j \le N$, $1 \le q \le Q$.

The offline/online decomposition is now clear:

- Offline, we compute $u(\mu_n)$ and we form the $\underline{A}_N^q$ and $\underline{F}_N$: this requires N (expensive) "a" finite element solves and $O(QN^2)$ scalar products;
- Online, for any given μ we form $\underline{A}_N$ from (1.8), then solve (1.7) for $\underline{u}_N(\mu)$, and finally evaluate $s_N(\mu) = \underline{F}_N^T\, \underline{u}_N(\mu)$: this requires $O(QN^2) + O(\frac{2}{3}N^3)$ operations and $O(QN^2)$ in memory storage.

We thus have constructed a very fast prediction of $s(\mu)$, $s_N(\mu)$, whose evaluation depends only on N and Q. The computational gains are particularly important at least $O(10)$, typically $O(100)$, and often $O(1000)$ if not more.

The originality of this work resides in the systematisation of the reduced basis approximation and the offline/online decomposition for new types of equations as we shall see later. Now, we turn to the *a posteriori* error estimation.

1.2.3 A Posteriori Error Estimation

In [62] and the publications presented later in this document, we have developed rigorous output bounds for $s(\mu)$ under the form

$$s_N(\mu) \;\le\; s(\mu) \;\le\; s_N(\mu) \;+\; \Delta_N(\mu) \quad \forall\, \mu \;\in\; \mathcal{D} \tag{1.9}$$

where $\Delta_N(\mu)$ is an upper bound for $|s(\mu) - s_N(\mu)|$. s_N is indeed a lower bound of s by applying symmetry and coercivity of a, as well as the Galerkin orthogonality, see the development 4.9 page 54.

Method I

The ingredients are *(i)* a function $g(\mu) : \mathcal{D} \to \mathbb{R}_+$, and *(ii)* a bilinear form symmetric continuous and coercive (independent of μ) $\hat{a} : X \times X \to \mathbb{R}$, such that

$$\underline{\alpha}_0 \|v\|_X^2 \;\leq\; g(\mu)\; \hat{a}(v,v) \;\leq\; a(v,v;\mu), \quad \forall\, v \in X, \forall\, \mu \in \mathcal{D} \tag{1.10}$$

for a real positive $\underline{\alpha}_0$. We then look for $\hat{e}(\mu) \in X$ such that

$$g(\mu)\; \hat{a}(\hat{e}(\mu), v) \;=\; R(v; u_N(\mu); \mu), \quad \forall v \in X, \tag{1.11}$$

where for a given $w \in X$, $R(v;w;\mu) = \ell(v) - a(w,v;\mu)$ is the weak form of the residual. The lower and upper bounds can then be evaluated as

$$s_N^-(\mu) \;\equiv\; s_N(\mu), \text{ et } s_N^+(\mu) \equiv s_N(\mu) + \Delta_N(\mu), \tag{1.12}$$

respectively, where

$$\Delta_N(\mu) \;\equiv\; g(\mu)\; \hat{a}(\hat{e}(\mu), \hat{e}(\mu)) \tag{1.13}$$

is the bound gap.

Next we prove the following result

$$1 \;\leq\; \eta_N(\mu) \;\leq\; \frac{\gamma_0}{\underline{\alpha}_0}, \quad \forall\, N. \tag{1.14}$$

where

$$\eta_N(\mu) \;=\; \frac{\Delta_N(\mu)}{s(\mu) - s_N(\mu)} \tag{1.15}$$

is the effectivity of the error estimation.

The offline/online decomposition procedure can be applied, using superposition, to $\Delta_N(\mu)$

$$\begin{aligned}\Delta_N(\mu) = \frac{1}{g(\mu)} \Bigg(c_0 + 2 \sum_{q=1}^{Q} \sum_{j=1}^{N} \sigma^q(\mu) u_{N\,j}(\mu) \Lambda_j^q + \\ \sum_{q=1}^{Q} \sum_{q'=1}^{Q} \sum_{j=1}^{N} \sum_{j'=1}^{N} \sigma^q(\mu) \sigma^{q'}(\mu) u_{N\,j}(\mu) u_{N\,j'}(\mu) \Gamma_{jj'}^{qq'} \Bigg)\end{aligned} \tag{1.16}$$

where $c_0 = \hat{a}(\hat{z}_0, \hat{z}_0)$, $\Lambda_j^q = \hat{a}(\hat{z}_0, \hat{z}_j^q)$, $\Gamma_{jj'}^{qq'} = \hat{a}(\hat{z}_j^q, \hat{z}_{j'}^{q'})$ and the $\hat{z}$ are solutions of $\hat{a}$ solves.

Again the online stage for error estimation does not depend on the finite element space dimension, but only N and Q. The complexity is about $O(N^2Q^2)$.

Method II

It is not always possible to derive rigorous error bounds for each type of equations, so a less rigorous method albeit easily applicable has been proposed — the recent developments, see section 1.9, on the a priori convergence properties of the reduced basis approximation make this method attractive.

This method is a simple application of the convergence properties of the reduced basis approximation. We pursue the following construction:

We choose $M > N$, and we introduce a sampling $S_M = \{\mu_1, \ldots, \mu_M\}$ and the associated reduced basis space $W_M = \text{span}\,\{\zeta_m \equiv u(\mu_m),\ m = 1, \ldots, M\}$; we require $S_N \subset S_M$ and thus $W_N \subset W_M$. We seek $u_M(\mu) \in W_M$ such that $a(u_M(\mu), v; \mu) = f(v), \forall\ v \in W_M$; we then evaluate $s_M(\mu) = \ell(u_M(\mu))$; and finally we construct the error bounds as

$$s^-_{N,M}(\mu) = s_N(\mu), \qquad s^+_{N,M}(\mu) = s_N(\mu) + \Delta_{N,M}(\mu), \tag{1.17}$$

where $\Delta_{N,M}(\mu)$, is the bound gap defined by

$$\Delta_{N,M}(\mu) = \frac{1}{\tau}\left(s_M(\mu) - s_N(\mu)\right) \tag{1.18}$$

for $\tau \in (0, 1)$. The effectivity of the estimation is defined as follows

$$\eta_{N,M}(\mu) = \frac{\Delta_{N,M}(\mu)}{s(\mu) - s_N(\mu)}. \tag{1.19}$$

In practice, we choose $M = 2N$.
The offline/online decomposition is readily obtained.
We prove also that the effectivity satisfies the following property, see section 4.4.2.0:

$$1 \ \leq\ \eta_N(\mu)\ \leq\ \text{const}, \quad \forall\ N. \tag{1.20}$$

1.3 Non-Symmetric Coercive Equations and Non-Compliant Outputs

Publication: [62].

The extension to non-symmetric equations and non-compliant outputs, $\ell(v) \neq f(v)\ \forall\ v\ \in X$, is a relatively straightforward extension of the previous section.

1.3.1 Reduced Basis Approximation

In order to reproduce an equivalent construction to the symmetric case, and in particular recover the convergence properties, we have to introduce the *dual* problem to the initial one (1.3) which we shall call the *primal* problem : for all $\mu \in X$, find $\psi(\mu) \in X$ such that

$$a(v, \psi(\mu); \mu) = -\ell(v), \quad \forall\ v \in X; \tag{1.21}$$

Then parallel to the primal problem (1.3), we form the reduced basis space associated to the dual problem at the same sampling points S_N, see section 4.5.1.0. Note that two strategies can be used at this stage: *(i)* the integrated one which builds $W_N = \text{span}\{(u(\mu_n), \psi(\mu_n)),\ n = 1, \ldots, N/2\}$ or *(ii)* the non-integrated one which builds $W^{\text{pr}}_{N/2} = \text{span}\{(u(\mu_n),\ n = 1, \ldots, N/2\}$ et $W^{\text{du}}_{N/2} = \text{span}\,\{(\psi(\mu_n),\ n = 1, \ldots, N/2\}$. The second one is more attractive from an implementation and cost point of view, because it allows to treat more easily problems with many non-compliant outputs, whereas the first strategy allow to expose the results more easily.

1.3.2 A Posteriori Error Estimation

Method I

Regarding the rigourous error estimator construction, it follows the same ideas of the previous section. We introduce only the residual associated to the dual problem, $R^{\mathrm{du}}(v;w;\mu) \equiv -\ell(v) - a(v,w;\mu), \forall\, v \in X$. We next define the bounds as follows

$$\bar{s}_N(\mu) = s_N(\mu) - \frac{g(\mu)}{2}\hat{a}(\hat{e}^{pr}(\mu), \hat{e}^{du}(\mu)), \text{ et} \tag{1.22}$$

$$\Delta_N(\mu) = \frac{g(\mu)}{2}\left[\hat{a}(\hat{e}^{\mathrm{pr}}(\mu), \hat{e}^{\mathrm{pr}}(\mu))\right]^{\frac{1}{2}}\left[\hat{a}(\hat{e}^{\mathrm{du}}(\mu), \hat{e}^{\mathrm{du}}(\mu))\right]^{\frac{1}{2}}. \tag{1.23}$$

And we have the following property on the effectivity of the error estimation

$$1 \leq \eta_N(\mu) \leq \frac{\gamma_0^2}{2\underline{C}\ \underline{\alpha}_0}. \tag{1.24}$$

where $\underline{C}$ is a constant independent of N.

Method II

The extension to the method II to non-compliant and non-symmetric operators follows also the same ideas as in the symmetric/compliant case, except that we introduce the dual space.

We recover also the good properties for the effectivity of the error estimation (greater or equal than 1 and bounded by a constant independent of N.)

1.3.3 Offline/Online Decomposition

The offline/online decomposition is also obtained, for the prediction as well as the error estimation, by following the same ideas as in the symmetric/compliant case.

1.4 Generalized Symmetric Eigenvalue Problems

Publication: [62].

The generalized eigenvalue problem is written under weak form as follows: find $(u_i(\mu), \lambda_i(\mu)) \in X \times \mathbb{R},\ i = 1, \ldots,$ such that

$$a(u_i(\mu), v;\mu) = \lambda_i(\mu) m(u_i(\mu), v;\mu),\ \forall v \in X,\ \text{et } m(u_i(\mu), u_i(\mu)) = 1. \tag{1.25}$$

where a is the symmetric, continuous, coercive form introduced earlier in the chapter, and m is, for example, the L^2 scalar product on Ω. The assumptions on a and m imply that the eigenvalues $\lambda_i(\mu)$ are real positive. We sort them in increasing order (as well as the associated eigenvectors u_i): $0 < \lambda_1(\mu) < \lambda_2(\mu) \leq \ldots$; and we suppose that $\lambda_1(\mu)$ and $\lambda_2(\mu)$ are distinct.

Our output of interest is $s(\mu) = \lambda_1(\mu)$ but we can consider other outputs, i.e. other eigenvalues.
The construction of the approximation using reduced basis follows the same principles already presented in previous sections except for the following differences:

- if we choose to develop method I for error estimation, we need to construct an approximation space containing also the second eigenmode (λ_2, u_2)
- the reduced basis approximation is an upper bound for the output of interest, conversely from the previous cases.

The properties of the effectivity of the error estimation are also obtained (greater than 1 and bounded by a constant independent of N.) Note that this application is useful to compute for example the rapid approximation of the coercivity or continuity constants as functions of μ.

1.5 Non-Coercive Equations

Publications: [62, 81].

We now turn to non-coercive equations, see chapter 5. A few ideas and developments were mentioned in [62], but only the publications [44, 72, 81] present the main results for the rapid predictions derived from non-coercive equations and the associated error quantification. Only the results from [81], which are less costly than those in [44], will be presented.
In what follows, a is a non-coercive bilinear form whereof we denote $\beta(\mu)$ the associated inf-sup constant.
The new ingredient is the construction of a lower bound $\hat{\beta}(\mu)$ for $\beta(\mu)$ that enters the definition of the error bounds as follows:

$$\Delta_N(\mu) \equiv \frac{\|r(\,\cdot\,;\mu)\|_{X'}}{\hat{\beta}(\mu)}\ , \tag{1.26}$$

where

$$r(v;\mu) = f(v) - a(u_N(\mu), v;\mu), \quad \forall\, v \in X\ , \tag{1.27}$$

is the residual associated to $u_N(\mu)$.
We verify also that the effectivity statisfies

$$1 \leq \eta_N(\mu) \leq \frac{\gamma(\mu)}{(1-\tau)\,\varepsilon_s}, \quad \forall\, \mu \in \mathcal{D}\ , \tag{1.28}$$

where $\gamma(\mu)$ is the continuity constant, $\tau \in]0;1[$ and $0 < (1-\tau)\epsilon_s \leq \hat{\beta}(\mu) \leq \beta(\hat{\mu})$, see proposition 5 page 78. The relation (1.28) establishes that Δ_N provides bounds for the approximation error.

The offline/online doesn't present any particular difficulties or new elements and is very similar to the previous sections.

1.5.1 Inf-Sup Lower Bound Construction

The construction of the inf-sup lower bound $\hat{\beta}(\mu)$ such that we have (1.28) is done the following way : we suppose that we have a set of J parameters, $\mathcal{L}_J \equiv \{\overline{\mu}^1 \in \mathcal{D}^\mu, \ldots, \overline{\mu}^J \in \mathcal{D}^\mu\}$, to which we associate the set of polygonal regions $\mathcal{R}^{\overline{\mu}^j,\tau}$, $1 \leq j \leq J$, where

$$\mathcal{R}^{\overline{\mu},\tau} \equiv \{\mu \in \mathcal{D}^\mu \,|\mathcal{B}_q^{\overline{\mu}}(\mu) \leq \frac{\tau}{C_X}\,\beta(\overline{\mu}),\ 1 \leq q \leq Q\}\ , \tag{1.29}$$

C_X is a finite constant, see 5.20 page 76 for more details, and

$$\mathcal{B}_q^{\overline{\mu}}(\mu) = \Gamma_q \sum_{p=1}^{P} D_{qp}\,|\mu_p - \overline{\mu}_p|\ ; \tag{1.30}$$

where Γ_q is a "continuity" constant associated to a_q and D_{qp} is defined as follows

$$D_{qp} = \max_{\mu \in \mathcal{D}^\mu} \left| \frac{\partial \sigma_q}{\partial \mu_p}\,(\mu) \right|\ , \tag{1.31}$$

We further suppose that

$$\bigcup_{j=1}^{J} \mathcal{R}^{\overline{\mu}^j,\tau} = \mathcal{D}^\mu\ . \tag{1.32}$$

Then we define $\mathcal{J} : \mathcal{D}^\mu \to \{1, \ldots, J\}$ such that, for a given μ, $\mathcal{R}^{\overline{\mu}^{\mathcal{J}(\mu)},\tau}$ is the region containing μ.

The lower bound is then defined as follows: given $\mu \in \mathcal{D}^\mu$,

$$\hat{\beta}(\mu) = \beta(\overline{\mu}^{\mathcal{J}(\mu)}) - C_X\,\mathcal{B}_{\max}^{\overline{\mu}^{\mathcal{J}(\mu)}}(\mu)\ , \tag{1.33}$$

$$\mathcal{B}_{\max}^{\overline{\mu}}(\mu) = \max_{q \in \{1,\ldots,Q\}} \mathcal{B}_q^{\overline{\mu}}(\mu) \tag{1.34}$$

with $\mathcal{B}_q^{\overline{\mu}}(\mu)$ defined by (1.30).

While studying the Helmholtz model problem, some enhancements are proposed. Indeed, we remark first that the number of regions J necessary to construct $\hat{\beta(\mu)}$ is about

$$J(\sigma^*, \Lambda, \varepsilon_s, \tau) \sim \frac{\sigma^* \ln(\frac{\Lambda}{\varepsilon_s})}{\tau} \tag{1.35}$$

where $\Lambda > \epsilon_s$ and σ^* is an eigenvalue — or resonance frequency — of $a_1(w,v) = \int_\Omega \nabla w \cdot \nabla v$ relative to $a_2(w,v) = -\int_\Omega wv$. We then observe that

- the dependency on σ^* is particularly debilitating, J becomes large and so the construction of $\hat{\beta}(\mu)$ becomes expensive when $\sigma^* >> 1$ which is the interesting case. To remedy this issue, we modify the construction of $\hat{a}$, see section 5.2.4.0

- the dependency on $\ln(\epsilon_s)$ is also a problem because the size of the region paving $\mathcal{D}^\mu$ goes to zero as we go near a resonance frequency. To remedy this issue, we modify the construction of the approximation space using a deflation technique, see section 5.2.4.0 page 84. This modification has a much lower cost than evaluating the dual norm of the residual, see (1.26) and (1.27), and is therefore advantageous to solve this issue.

1.6 Non-Linear Equations

Publications: [81, 80].

We turn now to non-linear equations. We apply the methodology to two types of non-linearities: *(i)* cubic and *(ii)* Burgers. In the first case, the results are published in [81]. In the second, preliminary results were published in [81], but the error bounds were not rigorous in the sense that they were providing a choice regarding the *proximity* of the reduced basis solution relative to the finite element one. A rigorous error bounds construction was published then in [80].

1.6.1 Poisson Equation with Cubic Non-Linearity

In what follows, for all $\mu \in \mathcal{D}^\mu \subset \mathbb{R}^2$, $a(\cdot\,,\cdot\,;\mu)\colon X \times X \to \mathbb{R}$ is given by

$$a(w,v;\mu) = a^{\mathrm{L}}(w,v;\mu) + a^{\mathrm{NL}}(w,v), \quad \forall\, w,v \in X\ , \tag{1.36}$$

where

$$a^{\mathrm{L}}(w,v;\mu) = \mu_1 \int_\Omega \nabla w \cdot \nabla v + \mu_2 \int_{\partial\Omega_{\mathrm{R}}} wv\ , \tag{1.37}$$

and

$$a^{\mathrm{NL}}(w,v) = \int_\Omega w^3 v \tag{1.38}$$

representing respectively a "Poisson-Robin" operator and a cubic non-linearity. We note that a^L is continuous, coercive and symmetric. The formulation of the problem and in particular of the reduced basis approximation is identical to the one in section 1.2.

However the a posteriori error estimation construction as well as the offline/online decomposition present some novelties.

A Posteriori Error Estimation

We construct $\Delta_N(\mu)$ as follows:

$$\Delta_N(\mu) \equiv \frac{\|r(\cdot\,;\mu)\|_{X'}}{\hat{\alpha}(\mu)}\ , \tag{1.39}$$

where $\hat{\alpha}(\mu)$ is a lower bound to the coercivity constant of a^L such that

$$\alpha(\mu) \geq \hat{\alpha}(\mu) \geq (1-\tau)\,\varepsilon_s, \quad \forall\, \mu \in \mathcal{D}^\mu\ , \tag{1.40}$$

for all $\tau \in]0,1[$ and $\varepsilon_s > 0$.
We show that $\Delta_N(\mu)$ provides indeed a bound for the error, see proposition 7 page 94.

We have used the ingredients from the construction of the error estimation in the non-coercive case, more precisely the construction of the lower bound of the inf-sup, and we applied it to the construction of the lower bound of the coercivity constant, $\hat{\alpha}(\mu)$. However we are in a simpler case if we take into account the monotonicity of $\alpha(\mu)$, coercivity constant of a^L.

Offline/Online Decomposition and Computing Cost

Our Poisson problem with cubic non-linearity admits an offline/online decomposition as well as the error estimator Δ_N. However the difficulty is now the computing cost due to the high order summations. In particular, we want to make sure that the online cost stays competitive as regards the finite element cost. A rapid estimation shows that the approximation cost is about N^4, which is not too depressing, but the evaluation dual norm of the residual $\|r(\cdot;\mu)\|_{X'}$ is about N^6 when applying a naive expansion of the cubic term. If we examine closely the computation of the dual norm of the residual, we observe that there are many symmetries that allow to reduce considerably the computing cost. In fact, when expanding the coefficients to obtain the affine decomposition of the non-linear term a^{NL}, we apply the multinomial formula[13]. We then obtain a complexity about $N^6/72$ with the cubic non-linearity which is considerable for a moderate N with respect to the naive expansion in N^6.
To obtain the numerical results, we used for the first time an adaptive generation of the reduced basis approximation space which allowed to reduce the high cost of the prediction and error estimation, see section 1.8 for more details.

1.6.2 Burgers Equation

Consider now the case of the Burgers equation. The abstract framework is the following:

We consider the domain $\Omega =]0,1[$. We define $X = H_0^1(\Omega)$, and $(\,\cdot\,,\,\cdot\,)_X = (\,\cdot\,,\,\cdot\,)_{H^1(\Omega)}$, $\|\cdot\|_X = \|\cdot\|_{H^1(\Omega)}$.

We shall consider only one parameter, $\mu \in \mathcal{D}^\mu \equiv [\mu^{\min} > 0, \mu^{\max}] \subset \mathbb{R}_+^{P=1}$. For all $\mu \in \mathcal{D}^\mu$. $a(\,\cdot\,,\,\cdot\,;\mu)\colon X \times X \to \mathbb{R}$ is given by

$$a(w,v;\mu) = a^{\mathrm{L}}(w,v;\mu) + a^{\mathrm{NL}}(w,w,v), \quad \forall\, v \in X\ , \tag{1.41}$$

where

$$a^{\mathrm{L}}(w,v;\mu) \equiv \mu a_0(w,v) = \mu \int_0^1 w_x v_x \tag{1.42}$$

$$a^{\mathrm{NL}}(w,z,v) = -\frac{1}{2}\int_0^1 w z v_x \tag{1.43}$$

are the bilinear and trilinear forms, respectively.

[13] http://mathworld.wolfram.com/MultinomialSeries.html

For $z \in X$, we define the bilinear form — associated to the derivative of the operator — $d_a(\cdot\,,\,\cdot\,;z;\mu)\colon X \times X \to \mathbb{R}$ such that

$$d_a(w,v;z;\mu) = a^{\mathrm{L}}(w,v;\mu) + 2a^{\mathrm{NL}}(z,w,v)\ . \tag{1.44}$$

We introduce the operator $T^{z;\mu}\colon X \to X$ such that, for all $w \in X$,

$$(T^{z;\mu}w, v)_X = d_a(w,v;z;\mu), \quad \forall\, v \in X\ ; \tag{1.45}$$

We have the relation

$$T^{z;\mu}w = \arg\sup_{v\in X} \frac{d_a(w,v;z;\mu)}{\|v\|_X}\ . \tag{1.46}$$

Moreover, for

$$\beta^z(\mu) \equiv \inf_{w\in X}\sup_{v\in X} \frac{d_a(w,v;z;\mu)}{\|w\|_X\,\|v\|_X}\ , \tag{1.47}$$

and

$$\gamma^z(\mu) \equiv \sup_{w\in X}\sup_{v\in X} \frac{d_a(w,v;z;\mu)}{\|w\|_X\,\|v\|_X}\ , \tag{1.48}$$

we have the following inequalities

$$\beta^z(\mu) = \inf_{w\in X} \sigma^z(w;\mu) \le \sup_{w\in X} \sigma^z(w;\mu) = \gamma^z(\mu)\ , \tag{1.49}$$

where

$$\sigma^z(w;\mu) \equiv \frac{\|T^{z,\mu}w\|_X}{\|w\|_X}\ . \tag{1.50}$$

The construction follows the Helmholtz case, except that we must now take into account the linearization point z in the definitions

Finally, we suppose that we have a constant ρ such that, for all $v \in X\ (= H^1_0(\Omega))$,

$$\|v\|_{L^4(\Omega)} \le \rho\|v\|_X\ ; \tag{1.51}$$

the existence of such a finite constant is guaranteed by the continuous embedding of X in $L^4(\Omega)$ [68].

Reduced Basis Approximation

The steps to construct the reduced basis approximation space are standard: first sampling then Galerkin projection. Note that the existence of bifurcation is not taken into account.

Why Burgers

The advantage of the reduced basis methods in the context of the Burgers equation regarding operation counts is very limited — we are in 1D. — However the construction of the approximation, then error estimation and affine decomposition apply identically to the Navier-Stokes equations when the geometry is not parametrized: in this case, since the basis functions associated to the velocity are divergence free, then the reduced basis approximation is also divergence-free and hence pressure-free. We then recover the Burgers case in the online stage.

Error Estimation

A First Estimation In [81], we propose an error estimator following closely the Helmholtz case:

$$\Delta_N(\mu) \sim \frac{\|r(\,\cdot\,;\mu)\|_{X'}}{\hat{\beta}(\mu)} \tag{1.52}$$

but we introduce also the following term

$$\Upsilon_N(\mu) \sim \frac{2\hat{\beta}(\mu)}{\rho^2} \;; \tag{1.53}$$

We then proved the following proposition:

Proposition 1. *Given $\mu \in \mathcal{D}^\mu$, for N sufficiently large such that*

$$\|r(\,\cdot\,;\mu)\|_{X'} \leq \frac{\hat{\beta}^2(\mu)}{2\rho^2} \;, \tag{1.54}$$

either

$$\|e(\mu)\|_{H^1(\Omega)} \leq \Delta_N(\mu) \;, \tag{1.55}$$

or

$$\|e(\mu)\|_{H^1(\Omega)} \geq \Upsilon_N(\mu) \;, \tag{1.56}$$

where $\Delta_N(\mu)$ and $\Upsilon_N(\mu)$ are given by (1.52) and (1.53), respectively.

This first result is not satisfactory because it does not guarantee that $\Delta_N(\mu)$ is a bound for $\|e(\mu)\|_{H^1(\Omega)}$.

Rigorous Error Estimation In [80], we construct a rigorous a posteriori error estimation for reduced basis predictions. The essential ingredient of the result is the usage of the theory developed by Brezzi-Rappaz and Raviart [15, 19]. We define a new quantity from the one already defined

$$\kappa = \sup_{z\in X}\sup_{w\in X}\sup_{v\in X} \frac{\int_0^1 zwv_x}{\|z\|_X\,\|w\|_X\,\|v\|_X} \;, \tag{1.57}$$

which allows us to bound the continuity constant $\gamma(z;\nu) \equiv \|d_a(z;\nu)\|_{X,X'}$ the following way $\gamma(z;\nu) \leq \nu + \kappa\|z\|_X$. Note that κ, and therefore $\gamma(z;\nu)$, is finite thanks to the continuous embedding of $H^1(\Omega)$ in $L^4(\Omega)$.

We then construct an upper bound (respectively, lower bound) of κ (respectively of $\beta_N(\nu)$). An upper bound for κ, $\tilde{\kappa}$, can be computed *a priori* and chosen to be $\tilde{\kappa} = \frac{1}{4}$. Regarding the lower bound construction of $\beta_N(\nu)$, $\tilde{\beta}(\nu)$, we introduce a set of parameters $U_J \equiv \{\tilde{\nu}^1 \in \mathcal{D}, \ldots, \tilde{\nu}^J \in \mathcal{D}\}$, a distance $d(\nu,\tilde{\nu};N) \equiv 2\beta_{N^{\max}}(\tilde{\nu})^{-1}[|\nu-\tilde{\nu}| + \frac{1}{2}\|u_N(\nu) - u_{N^{\max}}(\tilde{\nu})\|_{L^2(\Omega)}]$, and a mapping $\mathcal{I}_N\nu \equiv \arg\min_{\tilde{\nu}\in U_J} d(\nu,\tilde{\nu};N)$; we define finally $\tilde{\beta}(\nu) \equiv \frac{1}{2}\beta_{N^{\max}}(\mathcal{I}_N\nu)$.

We can then show the following lemma:

Lemma 1. *The inf-sup approximation* $\tilde{\beta}(\nu)$ *satisfies* $0 < \beta_0/2 \leq \tilde{\beta}(\nu) \leq \beta_N(\nu), \quad \forall\, \nu \in \tilde{\mathcal{D}}_N(U_J)$, *where* $\tilde{\mathcal{D}}_N(U_J) \equiv \{\nu \in \mathcal{D} \,|\, d(\nu, \mathcal{I}_N\nu; N) \leq 1\}$.

We have thus the confirmation that $\tilde{\beta}(\nu)$, for all ν in $\tilde{\mathcal{D}}_N(U_J) \equiv \{\nu \in \mathcal{D} \,|\, d(\nu, \mathcal{I}_N\nu; N) \leq 1\}$, is indeed a lower bound for $\beta_N(\nu)$.

We now define $\Delta_N(\nu) \equiv \frac{3}{2}\varepsilon_N(\nu)/\tilde{\beta}(\nu)$ and $\tau(\nu) \equiv \frac{3}{2}\tilde{\kappa}/\tilde{\beta}(\nu)$, where $\varepsilon_N(\nu) \equiv \|G(u_N(\nu); \nu)\|_{X'}$ is the dual norm of the residual.

Thanks to the previous lemma and the theory of Brezzi-Rappaz-Raviart [15, 19], $\Delta_N(\nu)$ is a rigorous and sharp error bound for the reduced basis approximation. The theorem reads as follows:

Theorem 1.6.1. *For* ν *in* $\tilde{\mathcal{D}}_N(U_J)$, *and* $\Delta_N(\nu) \leq \tau(\nu)^{-1}$, *(i) there exists a unique solution to equation 1.3 page 11,* $u(\nu)$, *in* $\overset{\circ}{\mathcal{B}}(u_N(\nu), \frac{3}{2}\tau(\nu)^{-1})$ *such that (ii)* $\|u(\nu) - u_N(\nu)\|_X \leq \Delta_N(\nu)$, *and (iii)* $\Delta_N(\nu) \leq \rho_N(\nu)\, \|u(\nu) - u_N(\nu)\|_X$. *Where* $\overset{\circ}{\mathcal{B}}(z, r) = \{w \in X \mid \|w - z\|_X < r\}$.

Numerical results corroborate the theorem in two distinct cases : *(i)* $\Delta_N(\nu) \leq \tau(\nu)^{-1}$ is always verified and *(ii)* $\Delta_N(\nu) \leq \tau(\nu)^{-1}$ is only verified for $N > N_0, N_0 > 1$. We note also that the rapid convergence of $\Delta_N(\nu)/\|u(\nu)\|_X$ to 0 and that the effectivity of the error estimation exhibits a good behavior.

1.7 Approximative Parametrization

Publication: [61].

Until now, we have developed a framework for *a posteriori* error estimation in the context of equations with exact parametrization. In [61], we focused on the situation where the mathematical model is incomplete: errors are introduced in the data that define the partial differential operators such as, for example, imperfect specifications, measures, computations or parametrized expansion. This article develops a mathematical framework for the rapid predictions of outputs of interest and rigorous error estimation. The later incorporates not only the reduced basis approximation error estimation but also the effects of the model truncation. Numerical tests were conducted on a relatively complex test case to illustrate the theoretical results.

The development being relatively technical and the exposition complex, only the main ideas will be presented.

1.7.1 Formulation

Given $\mu \in \mathcal{D}^\mu$, find $s(\mu) = \langle L, u(\mu)\rangle$, where $u(\mu) \in X$ satisfies

$$\langle \mathcal{A}(\mu)\, u(\mu), v\rangle = \langle F, v\rangle, \quad \forall\, v \in X\ . \tag{1.58}$$

where $\mathcal{A}(\mu)\colon X \to X'$ is our differential operator.

As mentioned in the introduction, we know only an approximation of $\mathcal{A}, \overline{\mathcal{A}}$. We have $\mathcal{A}(\mu) = \overline{\mathcal{A}}(\mu) + \widehat{\mathcal{A}}(\mu)$, where $\widehat{\mathcal{A}}(\mu)$ is small in a certain sense. We can consider the same for F : $F = \overline{F} + \widehat{F}$.

$\overline{\mathcal{A}}$ is coercive and continuous. $\overline{\mathcal{A}}$ and $\widehat{\mathcal{A}}$ admit an affine decomposition:

$$\overline{\mathcal{A}}(\mu) = \sum_{q=1}^{\overline{Q}} \alpha_q(\mu)\, A_q \ , \tag{1.59}$$

and

$$\widehat{\mathcal{A}}(\mu) = \sum_{q=1}^{\widehat{Q}} \beta_q(\mu)\, B_q \ ; \tag{1.60}$$

with

$$\beta_{\widehat{Q}}^{\max}(\mu) = \max_{q\in\{1,\ldots,\widehat{Q}\}} |\beta_q(\mu)| \ , \tag{1.61}$$

and

$$\beta_{\widehat{Q}}^{\max}(\mu) \le \hat{\beta}^{\max} \tag{1.62}$$

for $\hat{\beta}^{\max} \in \mathbb{R}_+$ independent of μ and $\widehat{Q}$ and preferably *small*. In general, we have only access to $\beta_{\widehat{Q}}^{\max}(\mu)$ — and we shall not have any details about $\beta_q(\mu)$.

We next define the scalar product and associated norms on $D_q \subset \Omega$, $((\,\cdot\,,\,\cdot\,))_q$ and $|||\,\cdot\,|||_q = ((\,\cdot\,,\,\cdot\,))_q^{1/2}$, respectively. We suppose that $|||\,\cdot\,|||_q$ is *uniformly* equivalent to $\|\cdot\|_{H^1(D_q)}$ for all functions in $H^1(D_q)$. It comes then from these hypothesis that there exists a finite positive constant $\hat{\rho}_\Sigma$ — *independent* of μ and $\widehat{Q}$ — such that

$$\sum_{q=1}^{\widehat{Q}} |||\, v|_{D_q} \,|||_q^2 < (\hat{\rho}_\Sigma)^2 \, \|v\|_X^2, \quad \forall\, v \in X \ . \tag{1.63}$$

We introduce also $\gamma_q \colon X \to \mathbb{R}$, $1 \le q \le \widehat{Q}$, such that

$$\gamma_q(w) \equiv \sup_{v\in X} \frac{\langle B_q w, v\rangle}{|||\, v|_{D_q} \,|||_q} \ , \tag{1.64}$$

and we require

$$\sup_{w\in X} \frac{\left(\sum_{q=1}^{\widehat{Q}} \gamma_q^2(w)\right)^{1/2}}{\|w\|_X} \le \widehat{\Gamma} \ , \tag{1.65}$$

for $\widehat{\Gamma} \in \mathbb{R}$ *independent* of μ and of $\widehat{Q}$.

1.7.2 Approximation

Two choices are possible concerning the construction of the approximation space: *(i)* $W_N = \text{span}\,\{\overline{u}(\mu_n),\, n = 1,\ldots,N\}$, where $\underline{u}_N(\mu) \in X$ satisfies

$$\langle \overline{\mathcal{A}}(\mu)\, \overline{u}_N(\mu), v\rangle = \langle F, v\rangle, \quad \forall v \in X_N \ . \tag{1.66}$$

which does not constrain us to incorporate the details of $\widehat{\mathcal{A}}(\mu)$, but obliges us to reconstruct W_N any time we improve $\overline{\mathcal{A}}(\mu)$ and *(ii)* $W_N = \text{span}\,\{u(\mu_n),\, n = 1, \ldots, N\}$ where $u(\mu_n)$ are solutions of (1.58), but which require that we know the details on $\beta_q(\mu)$, $1 \le q \le \widehat{Q}$. The choice of the construction of W_N will be problem dependent.

The formulation of the problem then reads as follows: given $\mu \in \mathcal{D}^\mu$, find $\overline{s}_N(\mu) = \underline{\overline{L}}_N^T\, \underline{\overline{u}}_N(\mu)$, where $\underline{\overline{u}}_N(\mu) \in \mathbb{R}^N$ satisfies

$$\underline{\overline{\mathcal{A}}}_N(\mu)\, \underline{\overline{u}}_N(\mu) = \underline{\overline{F}}_N \ . \tag{1.67}$$

where $\overline{u}_N(\mu)$ is a standard Galerkin approximation of $\overline{u}(\mu)$ in $W_N \subset X$.

We prove the first result on the approximation of $s(\mu)$, $\overline{s}_N(\mu)$:

Proposition 2. *for $\beta^{\max}_{\widehat{Q}}(\mu)$ and $\gamma_q(w)$ satisfying (1.61) and (1.64), respectively,*

$$\begin{aligned} |s(\mu) - \overline{s}_N(\mu)| \ \le \ & \|L\|_{X'} \Big(C_1 \inf_{v \in X_N} \|\overline{u}(\mu) - v\|_X \\ & + C_2\, \hat{\rho}_\Sigma\, \hat{\beta}^{\max} \widehat{\Gamma} \Big), \qquad \forall\, \mu \in \mathcal{D}^\mu \ ; \end{aligned} \tag{1.68}$$

where C_1 and C_2 depend uniquely on the coercivity and continuity constants and F.

1.7.3 Error Estimation

We introduce first a symmetric operator, coercive and continuous $\mathcal{C}(\mu)\colon X \to X'$ satisfying

$$1 \le \frac{\langle \mathcal{A}^S(\mu) v, v\rangle}{\langle \mathcal{C}(\mu) v, v\rangle} \le \rho, \quad \forall\, v \in X,\ \forall\, \mu \in \mathcal{D}^\mu, \tag{1.69}$$

with $\rho \in \mathbb{R}$ preferably small and $\mathcal{A}^S(\mu)$ is the symmetric part (positive definite) of $\mathcal{A}(\mu)$, defined by $\langle \mathcal{A}^S(\mu) w, v\rangle \equiv \frac{1}{2}(\langle \mathcal{A}(\mu) w, v\rangle + \langle \mathcal{A}(\mu) v, w\rangle)$, $\forall\, w, v \in X$.

$\mathcal{C}(\mu)$ is one of the essential ingredients of the error estimator and will play an identical role as the bilinear form $\hat{a}$ in the previous sections.

The a posteriori error estimator of $|s(\mu) - \overline{s}_N(\mu)|$ is given by:

$$\Delta_N(\mu) \equiv |||L|||_{X'} \left(\overline{\delta}_N(\mu) + \hat{\delta}_N(\mu) \right) \ , \tag{1.70}$$

where

$$|||L|||_{X'} = \sup_{v \in X} \frac{\langle L, v\rangle}{\langle \mathcal{C}(\mu) v, v\rangle^{1/2}} \ , \tag{1.71}$$

and

$$\overline{\delta}_N(\mu) \ \equiv \ \langle \mathcal{C}(\mu)\, \overline{E}(\mu), \overline{E}(\mu)\rangle^{1/2} \ , \tag{1.72}$$

$$\hat{\delta}_N(\mu) \ \equiv \ \hat{\rho}'_\Sigma(\mu)\, \beta^{\max}_{\widehat{Q}}(\mu) \left(\sum_{q=1}^{\widehat{Q}} \gamma_q^2(\overline{u}_N(\mu)) \right)^{1/2} \ . \tag{1.73}$$

où $\overline{E}(\mu) \in X$ satisfies

$$\langle \mathcal{C}(\mu)\, \overline{E}(\mu), v\rangle = \langle F, v\rangle - \langle \overline{\mathcal{A}}(\mu)\, \overline{u}_N(\mu), v\rangle,\ \forall\, v \in X\ ; \tag{1.74}$$

and $\beta^{\max}_{\widehat{Q}}(\mu) \in \mathbb{R}$ is the model truncation error (implicitly linked to $\widehat{Q}$ and to $\beta_q(\mu)$ through (1.61)).
It is clear that $\overline{\delta}_N(\mu)$ measures the reduced basis discretization error; and $\hat{\delta}_N(\mu)$ measures the model error truncation — approximation of $\mathcal{A}(\mu)$ by $\overline{\mathcal{A}}(\mu)$.
The following proposition proves that $\Delta_N(\mu)$ thus constructed is indeed a bound for $|s(\mu) - \overline{s}_N(\mu)|$:

Proposition 3. *For $\mathcal{C}\colon X \to X'$ satisfying (1.69), and $\hat{\rho}'_\Sigma(\mu)$ such that*

$$\sum_{q=1}^{\widehat{Q}} |||\, v|_{D_q}\, |||_q^2 \le (\hat{\rho}'_\Sigma(\mu))^2\, \langle \mathcal{C}(\mu)v, v\rangle,\ \forall\, v \in X\ . \tag{1.75}$$

we then have, $|s(\mu) - \overline{s}_N(\mu)| \le \Delta_N(\mu)$, $\forall\, \mu \in \mathcal{D}^\mu$, and $\forall\, N \in \mathbb{N}$.

We also prove that the effectivity verifies $1 \le \eta_N(\mu) = \frac{\Delta_N(\mu)}{|s(\mu)-\overline{s}_N(\mu)|} \le \Upsilon$, Υ not too large and we establish hence that Δ_N provides a sharp error estimation of the output of interest.

1.7.4 Offline/Online Decomposition

Δ_N contains essentially three components: $|||L|||_{X'}$; $\overline{\delta}_N(\mu)$; and $\hat{\delta}_N(\mu)$.
The offline/online decomposition of $|||L|||_{X'}$ and $\overline{\delta}_N(\mu)$ are standards in the context of reduced basis.
However the offline/online decomposition of $\hat{\delta}_N(\mu)$ is not so standard. We introduce

$$\hat{\hat{\delta}}^2_N(\mu) \equiv \sum_{q=1}^{\widehat{Q}} \gamma_q^2(\overline{u}_N(\mu))\ ,$$

in terms of which $\hat{\delta}_N(\mu)$ can be calculated as $\hat{\delta}_N(\mu) = \hat{\rho}'_\Sigma(\mu)\, \beta^{\max}_{\widehat{Q}}(\mu)\, \hat{\hat{\delta}}_N(\mu)$.

- The computation of $\hat{\rho}'_\Sigma(\mu)$ can be done exclusively offline by remarking that $\hat{\rho}'_\Sigma(\mu)$ is the maximum eigenvalue of the Rayleigh quotient:

$$\frac{\sum_{q=1}^{\widehat{Q}} |||\, v|_{D_q}\, |||_q^2}{\langle \mathcal{C}(\mu)v, v\rangle}\ ; \tag{1.76}$$

  and by using the offline/online decomposition of $\mathcal{C}(\mu)$.

- We suppose that $\beta^{\max}_{\widehat{Q}}(\mu)$ is given ;

- Finally we turn to $\hat{\hat{\delta}}_N(\mu) \equiv \sum_{q=1}^{\widehat{Q}} \gamma_q^2(\overline{u}_N(\mu))$, with

$$\gamma_q^2(\overline{u}_N(\mu)) = ||| \sigma_q(\mu) |||_q^2 \ , \tag{1.77}$$

where $\sigma_q(\mu) \in X_q$ satisfies

$$\big((\sigma_q(\mu), v|_{D_q})\big)_q = \langle B_q \, \overline{u}_N(\mu), v|_{D_q} \rangle, \quad \forall \, v \in X \ . \tag{1.78}$$

Here X_q is the space of functions in $H^1(D_q)$ which vanish on the part of ∂D_q which coincide with the Dirichlet part of $\partial\Omega$. If we then define the functions independent of the parameters $\sigma_{nq} \in X_q$, $1 \le n \le N$, $1 \le q \le \widehat{Q}$, such that

$$\big((\sigma_{nq}, v|_{D_q})\big)_q = \langle B_q \, \zeta_n, v|_{D_q} \rangle, \quad \forall \, v \in X \ , \tag{1.79}$$

we then have that, by using the reduced basis expansion, (1.78) and (1.79)

$$||| \sigma_q(\mu) |||_q^2 = \sum_{n=1}^{N} \sum_{n'=1}^{N} \overline{u}_{Nn}(\mu) \, \overline{u}_{Nn'}(\mu) \, \big((\sigma_{nq}, \sigma_{n'q})\big)_q \tag{1.80}$$

and thus that

$$\hat{\hat{\delta}}_N^2(\mu) = \sum_{n=1}^{N} \sum_{n'=1}^{N} \overline{u}_{Nn}(\mu) \, \overline{u}_{Nn'}(\mu) \left(\sum_{q=1}^{\widehat{Q}} \big((\sigma_{nq}, \sigma_{n'q})\big)_q \right) \ . \tag{1.81}$$

The offline/online decomposition is now clear, we need only to precompute offline

$$\Xi_{nn'} = \sum_{q=1}^{\widehat{Q}} \big((\sigma_{nq}, \sigma_{n'q})\big)_q \quad 1 \le n, n' \le N \ . \tag{1.82}$$

and given μ, $\hat{\rho}'_\Sigma(\mu)$, and $\beta_{\widehat{Q}}^{\max}(\mu)$, evaluate online

$$\hat{\delta}_N(\mu) = \tag{1.83}$$

$$\hat{\rho}'_\Sigma(\mu) \, \beta_{\widehat{Q}}^{\max}(\mu) \left(\sum_{n=1}^{N} \sum_{n'=1}^{N} \overline{u}_{Nn}(\mu) \, \overline{u}_{Nn'}(\mu) \, \Xi_{nn'} \right)^{1/2} \ . \tag{1.84}$$

The algorithmic complexity of this term, $\hat{\delta}_N(\mu)$, is only $O(N^2)$ and is hence negligible with respect to the complexity of computing $\overline{\delta}_N(\mu)$.

1.8 Reduced Basis Generation

In preparation: [59]

The construction of the sampling S_N has not yet been explained. In most articles [62, 61, 81, 80], S_N is generated by sampling log-randomly the parameter space $\mathcal{D}^\mu$, and following the algorithm 1

Algorithm 1 Log-Random generation of the reduced basis space

$N := 0$
$S_0 := \emptyset$
$N^{\max}$:= maximal number of basis functions to be generated
while $N < N^{\max}$ **do**
  $N := N + 1$
  pick log-randomly $\mu_N \in D^\mu$
  construct $S_N := \{\mu_N\} \cup S_{N-1}$
  construct $W_N := \{\xi_N = u(\mu_N)\} \cup W_{N-1}$
end while

1.8.1 Bounded Conditioning Number

First a classical problem with the previous approach is that the matrix A_N can be extremly ill conditioned. We can prove that

Proposition 4. *for coercive continuous bilinear forms*[14]*, when the basis function are orthonormalized, the conditioning number of* $A_N(\mu)$, $\forall\ \mu\ \in\ D^\mu$ *is bounded by the ratio between the continuity constant and the coercivity constant.*

$$\mathrm{cond}(A_N(\mu)) \leq \frac{\gamma(\mu)}{\alpha(\mu)} \tag{1.85}$$

This result, particularly important in practice, comes immediately from the application of the constant definition and the orthonormalization of the reduced basis.

1.8.2 Adaptive Generation

Now, we prefer to use rather an adaptive approach which uses the error estimator to determine the next point to use for the construction of the reduced basis space.

This approach is particularly attractive — $\Delta_N(\mu)$ is really cheap to compute and allows to sample very finely D^μ— as it ensures the decrease of $\max_{\mu \in D^\mu} \Delta_N(\mu)$. Note that orthonormalizing the basis functions ensures that the condition number is bounded and that we can reach a required accuracy ϵ. Moreover the adaptive procedure guarantee that no basis functions will be collinear and hence guarantees that the orthonormalization process won't break.

Various other adaptive sampling strategies of D^μ for different types of equations can be devised, they will be detailed in a publication in preparation see [59].

1.9 A Priori Convergence Properties

In preparation: [16]

This work, see chapter 8, is very recent — it will be submitted soon in October or November 2005. — It follows the one of T. Patera et al. on the a priori convergence of the reduced

[14] Similar results could be derived for other types of equations.

Algorithm 2 Adaptative generation of the reduced basis space

$N := 0$
$S_0 := \emptyset$
ϵ accuracy to be reached by the reduced basis approximation
while $\Delta_N^{\max} > \epsilon$ $\Delta_N^{\max}$ is the maximum error given by reduced basis approximation over a random sample of parameters **do**
  $N := N + 1$
  if $N := 1$ **then**
    pick log-randomly $\mu_N \in D^\mu$
     or alternatively
    pick a point μ_N *by hand* $\in D^\mu$
  end if
  construct $S_N := \{\mu_N\} \cup S_{N-1}$
  construct $W_N := \{\xi = u(\mu_N)\} \cup W_{N-1}$ {Gather statistics on W_N:}
  $\Delta_N^{\max} := \max_{\mu \in D^\mu} \Delta_N(\mu)$
  $\mu^{N+1} := \arg \Delta_N^{\max}$
end while

basis methods in 1D with 1 parameter, see [47, 46, 62]: they proved that there is exponential convergence, result which can be extended to multiple parameters by tensorization, but the convergence is in $e^{-N^{1/P}}$ with P being the dimension of the parameter space (or number of parameters) and N is the reduced basis space dimension. However we observe in practice that this result is far from optimal. This recent work, under an hypothesis of exponential decrease of the eigenvalues of a certain operator validated with numerical tests, recovers the exponential convergence in e^{-cN} for the reduced basis approximation and thus is independent of P, see lemma (4) page 29.

1.9.1 Proof

The proof works as follows:

We are interested in the solution $u(x, y)$ of a parametrized partial differential equation defined on the geometric domain $\Omega \subset \mathbb{R}^d, d = 1, 2, 3$ and a parameter domain $D \subset \mathbb{R}^P$ and the equation reads under abstract form (2.3).

We introduce the operators K and T which are convolutions with the kernels

$$\begin{aligned} K(\mu, \mu') &= \int_\Omega u(x, \mu) u(x, \mu') dx \\ T(x, x') &= \int_{D_r} u(x, \mu) u(x', \mu) d\mu. \end{aligned} \tag{1.86}$$

i.e. the operators

$$\begin{aligned} (K\ f)(\mu) &= \int_{D_r} K(\mu, \mu')\ f(\mu')\ d\mu' \\ (T\ u)(x) &= \int_\Omega T(x, x')\ f(x')\ dx' \end{aligned} \tag{1.87}$$

$u(x,\mu)$ then can be expanded as follows

$$u(x,\mu) = \sum_{k=1}^{\infty} \sqrt{\lambda_k}\phi_k(\mu)\psi_k(x). \tag{1.88}$$

where λ_k are the eigenvalues of K and T and ϕ_k, ψ_k are the eigenmodes of K and T respectively. We then make the following hypothesis on λ_k:

$$\lambda_k \leq ce^{-f(k)} \tag{1.89}$$

where f is at least linear such that $f(k) \geq \rho k, \rho > 0$. This hypothesis has been verified numerically on a few model problems in 2D.

First we prove that $U = \{u(.,\mu); \mu \in D\}$ is close to a linear space of small dimension Ψ_k spanned by the first functions $\psi_1, ..., \psi_n$ such that

$$\Psi_k = \text{span}\{\psi_1, \ldots, \psi_k\} \tag{1.90}$$

This is the result by the following lemma:

Lemma 2. *We define* $l_m = \sqrt{\sum_{k=m+1}^{\infty} \lambda_k}$. *We have:*

$$\forall\, u \in U \ \|u - \Pi_{L^2}^{\Psi_k} u\|_{L^2} \leq C_k^{1/P}. \tag{1.91}$$

where $\Pi_{L^2}^{\Psi_k} u$ *is the* L^2 *projection of* u *on* Ψ_k *i.e.*

$$\Pi_{L^2}^{\Psi_k} u = \text{arginf}_{\psi \in \Psi_k} \|u - \psi\|_{L^2}.$$

Once this result obtained and under hypothesis (1.89) we have

$$\forall\, u \in U \ \|u - \Pi_{L^2}^{\Psi_k} u\|_{L^2} \leq Ce^{-\alpha\, k} \tag{1.92}$$

with $\alpha > \ln(2)$, which means that $\rho > 2P\ln(2)$.

We construct then the reduced basis space such that the basis functions are orthogonal $V_N = \text{span}\{u_\ell = u(.,\mu_\ell), \mu_\ell \in D, \ell = 1 \ldots N\}$ and we prove the following lemma

Lemma 3. *For each* u_ℓ, $\ell \geq 1$, *there exists* $v_\ell \in \Psi_k$ *such that:*

$$\|u_\ell - v_\ell\| \leq C2^{\ell+1}e^{-\alpha\, k}. \tag{1.93}$$

Thanks to this last result and to the construction of the u_ℓ, we prove the final result

Lemma 4.

$$\exists \beta > 0 \mid \forall\, u \in U \ \inf_{u_N \in V_N} \|u - u_N\| \leq Ce^{-\beta\, N}. \tag{1.94}$$

1.9.2 Verifications

In order to corroborate the hypothesis (1.89), the eigenvalues of the operator K have been computed on a few academic problems for $P = 2$, $P = 3$ and two different geometries in dimension 2.

These results are very interesting and original, see section 8.4. Indeed in order to verify the exponential decay of the eigenvalues, the standard floating point arithmetic of modern computers is not sufficient — it is not accurate enough and does not allow to verify (1.89).

We had access to a set of generic scientific computing components, see section 2.1 of the next chapter, for the finite element, then reduced basis approximation and the verification of the exponential decay of the eigenvalues of K on a tensorized grid of D. All the computations have been done in quad-double precision, see [30], in order to have at least 60 significant digits.

It is important to note that almost all computing steps and data structures (besides the mesh data structure and the parameters of D) must use this numerical type of quad-double accuracy including the linear solvers, without which the results would not be more accurate than the least accurate computing component.

Besides the verification of hypothesis 1.89 on these few particular cases, these results illustrate the necessity to develop generic tools that would otherwise require heavy and very specific developments and which would be used only in this particular context. That would be a very good example of non-reuse of scientific computing components. These comments are illustrated further in the next chapter.

Chapter 2

Components for Scientific Computing

We have just seen, in the preceding chapter, the mathematical development of the reduced basis methods with error estimations for various types of equations and their three components: *(i)* a prediction of the output of interest by projection on a space of reduced dimension, *(ii)* a quantification of the error of approximation and *(iii)* an offline/online decomposition allowing during the online stage to obtain reliable results quasi-instantaneously. The design and the implementation of these methods present particularly interesting challenges.

First of all, they not only require to have an overall vision of the standard methods of approximation but also to control the ingredients necessary to the construction of the three components for the various classes of equations which we reviewed. These ingredients are often re-used or slightly modified to adapt to the problem of interest, such as for example the calculation of the constants of coercivity, continuity and inf-sup. One of the objectives of my work which will be presented in the coming chapter is the design and development of a computing environment whose components are generic and can be easily re-used in order to build new extensions quickly or to rapidly develop new applications.

Then, it is clear that the intrinsic properties of the reduced basis output bounds methods open new prospects as regards scientific computing. In particular, the integration of scientific computations in extremely simplified interfaces specifically conceived to solve a particular problem makes it possible to the non-specialist (computing methods) to use them with confidence (error estimation) in the specific context of the problem in real-time (fast prediction). We thus developed modern interfacing tools in the context of the application and in particular, we were interested in the generation of interactive documents.

Lastly, if the traditional methods to solve partial differential equations are not particularly suited to distributed computing and the client-server paradigm and often require the development of heavy data structures, the reduced basis output bounds methods on the other hand fit very well such architectures. However the specific nature of these methods require special attention: it is not only about treating an input/output relationship, but also to take into account the uncertainty quantification and a few other properties to allow the construction of an environment for real-time computing with certificate of fidelity for *(i)* repeated evaluation of the outputs of interest, *(ii)* the composition of these quantities to create new ones (adimensionalization, dimensionalization, algebraic dependencies, ...), *(iii)* sensitivity analysis of the

output of interest in instantaneously or quasi-instantaneously and finally *(iv)* the optimization and the control in real-time of these quantities. We have implemented such an environment which is still being developed and incorporates new mathematical improvements in the context of the reduced basis methods.

Figure 2.1 presents the global architecture and the main components of the computing environment [15] that have been put in place.

In this chapter, we indeed will explore these different aspects : *(i)* a mathematical kernel for the scientific computation in general and the solution of partial differential equations in particular, *(ii)* a language integrated into *C++* for numerical integration, the specification of the variational formulations and automatic differentiation and finally *(iii)* a preliminary version of the distributed environment with interactive interfaces. In a last section, a very general description of the technology which won the contest (along with others) organized by the French Ministry of research in 2003 regarding innovative technologies is briefly presented.

2.1 Mathematical Kernel

En préparation: [65]

This section presents some aspects of a library I wrote but whose design has not yet been published, see [65]. It has nevertheless allowed to obtain original results such as for example the verification of the exponential decay of the eigenvalues of convolution operators associated to the reduced basis approximation, see section 1.9 page 27, and it is the mathematical kernel of the language for partial differential equations in the next section.

The components to obtain these results have required the development of a completely generic framework for the various ingredients in partial differential equation resolutions in the sense that they are independent of the numerical type manipulated: in particular with respect to the standard floating point types which were insufficient in terms of accuracy for the computations mentioned above.

More precisely, the components are the following(non exhaustive) :

- a generic linear algebra library; a few exist that satisfy the needs of the following ingredients ;
- geometric entities — d simplices with $d = 1, 2, 3$ and product of simplices of order k — are not only used to represent the elements of a mesh but also to construct the polynomial sets, quadrature formulas and finite elements and thus must be generic ;
- polynomials and their representations; in particular I have used the construction framework presented in [32] ;
- integration methods; in particular I have used the framework presented in [32] ;

[15] I am actually the coordinator of the LifeV project, www.lifev.org, which is a methodology showcase for the European project, http://mox.polimi.it/it/progetti/haemodel/. In particular, LifeV implements and shall implement a number of state of the art applications in the context of the cardiovascular system simulation. The complexity of these applications requires the development of efficient scientific computing components for multiscale and multiphysics simulations.

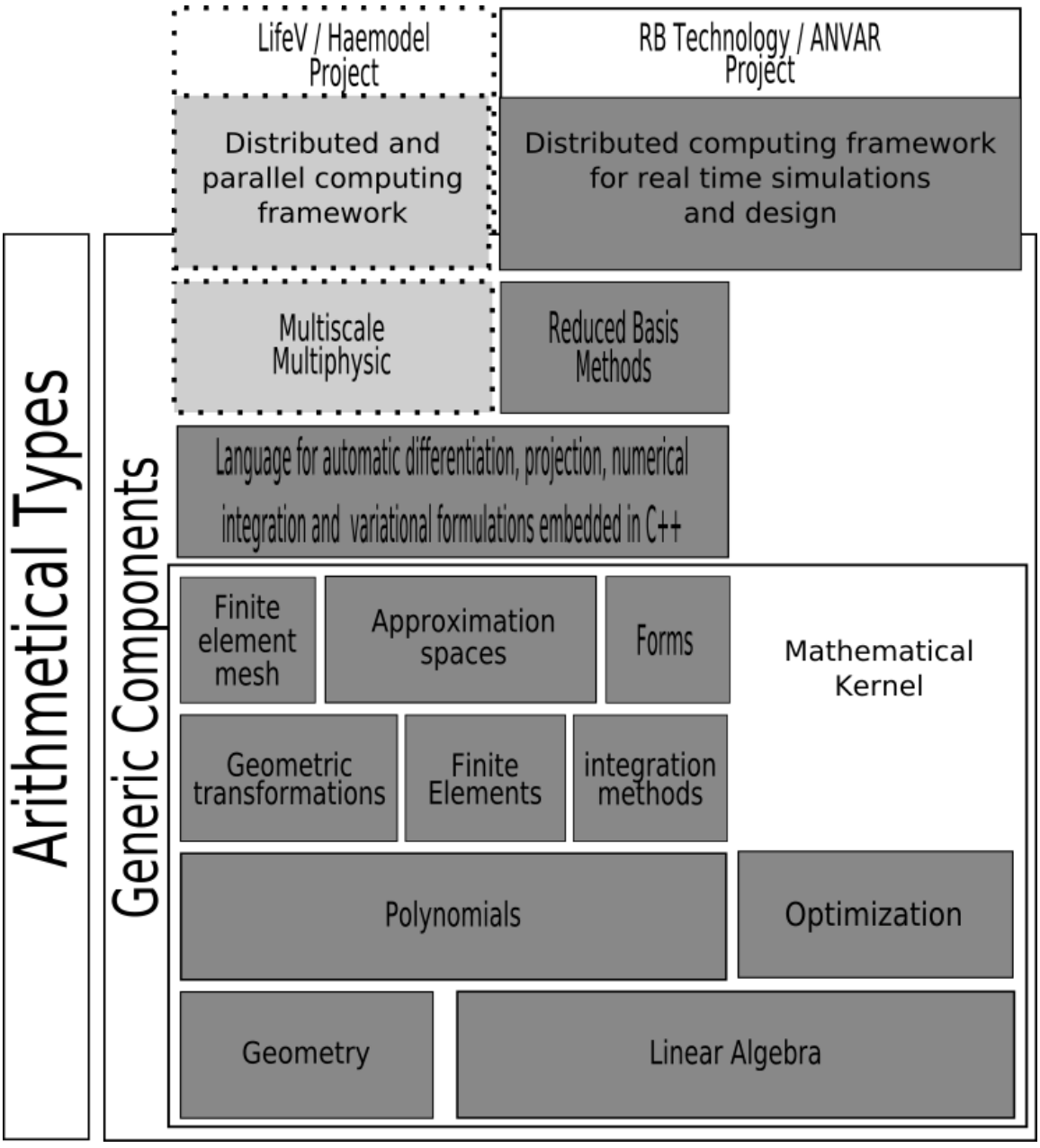


Figure 2.1: Computing kernel and *C++* integrated language for partial differential equations resolutions; reduced basis output bounds methods and distributed environment for real-time simulations; multiscale and multiphysics environment in the LifeV project.

- finite elements — for example Lagrange element for dimension d with $d = 1, 2.3$ and order k —, see [22] and [32] ;
- the geometric transformation between the reference convex and a convex allowing to do all the computations on the reference convex where the integration formulas and finite elements have been constructed ;
- approximation spaces parametrized by the underlying mesh and a finite element basis ;
- linear and bilinear forms and their representation, they are particularly useful in conjunction with the language presented in the next section.

We thus obtain a set of basic tools in *C++* representing their mathematical counterpart with no a priori on the arithmetical representation that will be used and without losing performances for standard numerical types. To reiterate, the design is completely generic and general and follows the mathematical construction closely. To achieve this, the essential techniques in *C++* are meta-programming, static polymorphism and static computation (static means here 'at compile time').
The supported numerical types that allows to instantiate the computation components are:

- the standard floating point types in *C++*
- `std::complex<>` complex type in *C++*
- `ADtype<.,.>` an automatic differentiation type, see the next section
- `dd_real` a double-double precision type from the QD library, see [30]
- `qd_real` a quad-double precision type from the QD library
- `mp_real` an arbitrary precision type from the ARPREC library, see [11]

Other numerical types can easily be added : some shall be integrated soon such as an interval arithmetic type to measure the impact of perturbations of uncertainty in computations. Of course, some components may be sensible only for certain numerical types, for example the different representations of a real (`double`, `long double`, `dd_real`, `qd_real` and `mp_real`).

2.2 A Language for the Resolution of PDE

submitted: [66]

We have just seen some components for partial differential equations solves. In order to benefit from these tools, we have developed a language integrated in *C++* allowing to define the problem to be solved under a form very close the mathematical expression. This language uses many keywords of FreeFEM++, see [29], but design and implementation are entirely different. The language is written in *C++* and is using directly within *C++*.

To illustrate further, listing 2.1 shows a small example for the possibilities of the language which have been taken from [66], where I compare the definition of a bilinear form from the mathematical point of view and the language point of view.

$$\begin{aligned} a : X_h \times X_h &\to \mathbb{R} \\ (u, v) &\to \int_\Omega \nabla u \cdot \nabla v + uv \end{aligned} \tag{2.1}$$

Listing 2.1: Variational Formulation in *C++*

```
// a mesh of Ω ⊂ R^d, d = 1,2,3
Mesh mesh;
// scalar finite element space P_K, K = 1,2,3,...
Space<Mesh,FEM_PK<d,K> > Xh(mesh);
// 2 elements of the Space Xh
Space<Mesh,FEM_PK<d,K> >::element_type u(Xh), v(Xh);
// A matrix in CSR format
csr_matrix_type M;
// bilinear form with M as its matrix representation
// with integration over all elements of the mesh
// and a method for exact integration of polynomials of
// degree ≤ K (IM_PK<d,K>)
BilinearForm<Xh> a(u,v,M);
a = integrate( elements(mesh),
               dot(gradt(u),grad(v))+idt(u)*id(v),
               IM_PK<N,K>() );
```

The example is rather trivial but already exhibits the possibilities of the language. The technique allowing this kind of high level programming is called *expression template*, see [77]. The truly innovative aspect of the publication is to be able to share the construction of the mathematical expressions between different applications — here, projection, numerical integration, variational formulation and automatic differentiation — and to provide different engines to evaluate the mathematical expressions depending on the context. The second main result of this article shows that it is indeed possible to build formulations and in particular mixed formulations whose evaluations during the construction of the bilinear and linear forms will be optimal without extra cost when evaluating integrals due to terms evaluated to 0 depending on the block being constructed — for example in the case of the Stokes equations, we have the velocity, pressure and mixed blocks. — This property of the language facilitates greatly the writing of variational formulations while ensuring that the language in conjunction with *C++* will generate efficient code.

Besides these two fundamental aspects of the language and the fact that the library is 1-2-3D, the advantages of an embedded language linked directly to the computing library are several folds :

- the inherent complexity of constructing a compiler/interpreter can be ignored;

- other libraries can easily be coupled with the library and the language which is not the case for specific language such as FreeFEM++ which would then have to develop their own library system;
- embedded languages inherit the capabilities of the host language, here *C++* : debugging, optimization, strong typing, generic programming. . . ;
- from a teaching point of view, we cumulate the learning of the host language (*C++*) and embedded language (variational formulation, numerical integration or automatic differentiation).

In the article, a benchmark of the performances to fill of matrices or vectors for diffusion equation (D), diffusion-reaction (DR) and diffusion-advection-reaction (DAR) is presented. Some interesting properties of the language are displayed:

- the performances in terms of the number of elements grows linearly (in log-log) which is the expected behavior
- the performance gap at a given number of elements between the different equations D and DR are very small and in certains cases also with DAR
- the extra cost due to the non-constant coefficients with respect to the equivalent constant one is very small, either in 2D or 3D. In particular, in the most debilitating case the ratio is always less than 2 and most often between 1.1 and 1.3. This is to be compared to the results of [56] where the ratios are comprised between 2 and 5 (5 in the DAR 3D case) and for relatively coarse meshes. The difference is that in the later the non-constant coefficients are treated using standard *C++* functions, whereas in the language, the non-constant coefficients are treated by the expression template technique.

These remarks illustrate the relevance of using techniques such as expression template or meta-programming for scientific computing. For more details, see section 10.4.1 page 214.

Moreover, the article proposed some test case which allow to evaluate the possibilities of the language : a variational inequality, see section 10.4.2 page 219, Stokes equations with and without stabilization , see section 10.4.3 page 221 and rigid particle in a shear flow, see section 10.4.4 page 222. Other application domains have been successfully tested: 2D and 3D linear elasticity, Navier-Stokes 2D and 3D. Listing 2.2 shows an example the specification for a Stokes problem:

Listing 2.2: Mixed Variational Formulation

```
// mixed finite element space (P2 velocity, P1 pressure)
// in 3D
typedef MixedFESpace<Mesh_t,
                     FEM_PK<3,2,vectorial>,
                     FEM_PK<3,1,scalar> > fespace_type;
fespace_type V_h;
V_h::element_type U,V;
// views for U and V
typedef fespace_type::fespace_1_type X_h;
```

```
typedef fespace_type::element_2_type M_h;
fespace_type::element_1_type u = U.element1();
fespace_type::element_2_type p = U.element2();
fespace_type::element_1_type v = V.element1();
fespace_type::element_2_type q = V.element2();
csr_matrix_type A;
MixedBilinearForm<fespace_type> a( V_h, V_h, A );
a = integrate( elements(mesh),
               dot(gradt(u), grad(v)) - idt(p)*div(v) +
               id(q)*divt(u) + 1e-6*idt(p)*id(q) )+
    // 10 identifies a dirichlet boundary
    on( 10, u, F, Px()*(1-Px())+Py()*(1-Py()) )+
    // and 20 too
    on( 20, u, F, 0 );
```

The mathematical kernel and the language provide thus a powerful development platform for scientific computing and partial differential equations solves, which is abstracting the underlying arithmetic until the instantiation of the problem to be solved. They form the basic blocks for the reduced basis methods implementation, see figure 2.1.

The results already obtained are very encouraging and there are still many possibilities to improve the performances and increase the number of application domains while reusing or extending to the maximum the components already in place.

2.3 Distributed System for Real-Time Simulations

Publication: [67]
In preparation: [60]

We have seen two upstream aspects of the implementation of the reduced basis output bounds methods. Now we turn to some downstream work. We have been developing an original and very cheap infrastructure allowing very simple interfaces for real-time computing requests through Internet. Regarding the interfaces, we were interested in electronic documents that could replace more standard archiving methods. In particular, we have been developing a set of tools — only based on open source software — that allows to write aesthetic documents where equations are *actionable* and allowing a user in a natural context to request and present results in real-time.

2.3.1 Real-Time Simulations Repository

The numerical methods proposed in the first chapter are rather unique compared to the standard ways to solve partial differential equations. The reduced basis output bounds methods — in particular the global approximation spaces, *a posteriori* error estimators, offline/online decomposition — are designed to make partial differential equations truly "useful": *(i)* real-time regarding operation counts ; *(ii)* "blackbox" regarding reliability and *(iii)* appropriate (customized) to the context regarding the inputs/outputs required by the methods.

But to effectively profit from this methodology, in particular the creation of an online code repository, it must reside in a special environment. This environment must allow a user *(i)* to

provide — in a native/natural context — the problem, the output of interest and the inputs values; and *(ii)* to receive quasi instantaneously the prediction and associated certificate of fidelity.

We have created a very simple distributed environment whose elements are servers modelling the equation

$$(s_N(\mu), \Delta_N(\mu)) = f(\mu), \quad \forall \mu \in \mathcal{D}^\mu \tag{2.2}$$

A general overview of the distributed environment for real-time simulation with a certificate of fidelity is presented at the section section 9.5.2 page 173.

2.3.2 Clients

Two client interfaces are briefly presented: SimTeX and SimLab.

SimTeX allows to generate interactive PDF documents with actionable equations which model equation (2.2) to define the "graphical interface", see the actionable equation 9.5.3.0 page 175 for a simple example. See also the section 9.5.3.0 page 174 for a more detailed description of SimTeX.

SimLab is a MatLab interface and allows either to use the simulation servers registered on the repository, or to create new ones and register them on the server. See section 9.5.3.0 page 175 for a more detailed description of SimLab.

Since then, new clients have been developed such as a Python client: Python can be used to manipulate and generate simulation servers like in MatLab.

2.3.3 API

Now we turn to the application programming interface (API) which allow to build new servers and clients very easily and quickly. Here is an example which registers in the repository the simulation `Troot` associated to the model `fin3d` (model for a three dimensional thermal fin) and this is achieved in very few lines of *C++* code, see listing 2.3.

Listing 2.3: Piece of *C++* code to build a real-time simulation server with a certificate of fidelity

```
#include <SApplicationServer.hpp>
#include <SModelOutput.hpp>

using namespace St;

// simulation server
class Troot: public SModelOutput
{
public:
  Troot( SModel* parent, const char* name )
    : SModelOutput( parent, name )
  {<snip> }
  virtual SOutput* run( SParameterSet const& pset )
   {
     SOutput* output = new SOutput;
```

```
      // do the computation here for the
      // parameter set, pset
      . . .
      output->output = <prediction value>;
      output->error = <associated error estimator value>;
      return output;
    }
};

int main(int argc, char** argv)
{
  SCommandLineArguments::init( argc, argv );
  SApplicationServer* server = new SApplicationServer();
  SModel* fin3d = new SModel( 0, "fin3d");
  SModelOutput* troot = new Troot( fin3d, "Troot");

  server->setMainSimget( fin3d );
  server->run();
}
```

From the point of view of the client, the interface is also very simple, see listing 2.4.

Listing 2.4: *C++* client for a real-time simulation server

```
#include <SApplication.hpp>
#include <SModelOutput.hpp>
using namespace St;


int main(int argc, char** argv)
{
  SCommandLineArguments::init( argc, argv );
  SApplication* client = new SApplication();

  SModelOutput_var fin3d_troot =
       client->resolve( "fin3d/Troot" );
  fin3d_Troot->setNewParameterSet( pset );
  fin3d_Troot->setRange( range );
  fin3d_troot->run( MultiParameter );

  coOutputArray_var results = fin3d_Troot->getOutputs();
  for( ULong  __i = 0;__i < __output_pl->length();++__i)
    {
      std::cerr << "output(␣" << __i << ")="
                << result( __i ).output << "\n";
      std::cerr << "error(␣" << __i << ")="
                << result( __i ).error << "\n";
    }
}
```

In this example, we look for the simulation server `Troot` associated to the model `fin3d`. Once found we pass it the parameters μ and retrieve the output of interest and error estimation.

The environment has been designed from the start to be efficient and to benefit from all opportunities to optimize and especially parallelize computations. In particular, *(i)* the servers are *multi-threaded* automatically at two levels: the CORBA server (multi-threaded) and the simulation server itself which can evaluate for many μ using threads; *(ii)* the registered servers to the repository are redundant : several copies are distributed on different computing servers (hardware) and which may not be on the same sub network ; and finally *(iii)* the clients can send computing requests in parallel in order to benefit from the strongly distributed/parallel architecture.

2.3.4 New Extensions of the API

A few particularly interesting extensions extensions have been added to the interface, see [60]. Of course the servers already in place don't support these new interfaces, it is therefore required to develop a versioning system for the supported interfaces in order to have them evolve while preserving the legacy servers — until they are ported to the new interface.

Online Control of Accuracy

A first extension is the support of the following features

- Online control of the accuracy: the user chooses the accuracy he desires for the predictions that will be done afterwards. Thanks to the developments done in 1.8 page 26, this requires *no online computations* at all , everything has been tabulated offline and in particular for all N the corresponding maximum error over D^μ.
- Online control of computing time: the user sends a request to the simulation server to make the next computations within a time constraint; this kind of application is crucial in hard real-time where every commands must be executed imperatively within an imparted time; the computation accuracy is the best possible one within the time constraint.

Automatic Differentiation and Optimization

We have seen also that these methods are going to be used in the context of many evaluations such as for example sensitivity analysis or optimization. The interface has been naturally extended to support automatic differentiation, with respect to the parameter set μ see for example section 2.2, of the online stage of reduced basis methods. The automatic differentiation is particularly suited to this context for two reasons: *(i)* the number of parameters in the methodology presented in the first chapter is typically small(less than 100) and *(ii)* the affine decomposition of the bilinear and linear forms allow the trivial application of automatic differentiation.

So the blackbox models now

$$(s_N^{(i)}(\mu), \Delta_N^{(i)}(\mu),) = f(\mu), \quad \forall \mu \in \mathcal{D}^\mu \text{ et } i = 0, 1, 2 \tag{2.3}$$

where $^{(i)}$ denotes the order of the differentiation. Thus the server associated to the blackbox with this slightly enhanced interface wait forever for new computing requests.

A new type of simulation servers has then been developed: they provide optimization services by composing forward simulation servers — typically different outputs of interest of the same model – in order to generate an application for very rapid online design — for the model associated with the outputs of interest.

2.4 Short Presentation of the Technology

The research activities presented in this and the preceding chapter regarding the various contributions of the research group of for A.T. Patera at MIT led to the development of the technology that was presented at a contest of the French Ministry of research in 2003 and received the price in the emergence category. It is briefly described in a " fact sheet". The document inserted in the following pages was carried out in collaboration with a consulting company and was distributed to 30 companies involved in engineering design and real-time computing for a marker study. The technology fits truly in objectives presented in the introduction of this synthesis, see page 3.

The technology evolved/moved since the development of the "fact sheet" and it now supports among others parabolic equations treatment, the developments as regards adaptive generation of the reduced basis space and also the recent developments concerning the magic points method to tackle non-affine decomposition.

NODAL CONSULTANTS

Paris - 209-211 rue de Bercy, F-75585 Paris Cedex 12
Tél +33 (0)1 40 02 75 57 - Fax +33 (0)1 40 02 75 44
info-paris@nodal.fr - http://www.nodal.fr

Lyon - 21 avenue G. Pompidou, F-69486 Lyon Cedex 03
Tél +33 (0)4 72 91 32 77 - Fax +33 (0)4 72 91 30 30
info-lyon@nodal.fr - http://www.nodal.fr

Real-time and reliable simulations

- Christophe Prud'homme, a Research Scientist at the Massachusetts Institute of Technology, then at the Numerical Analysis Laboratory Jacques-Louis Lions at the Université Paris 6 and currently Research Assistant in the Lausanne Federal Institute of Technology (EPFL), has developed a **real-time simulation** system for **mechanical engineering** problems with a **certificate of reliability** and an **accurate control** of either **maximum computation time** or of **maximum error.** This system is the product of a common project between MIT and Université Paris 6. It has been financed by various institutions, such as Nasa, the US Defense Advanced Research Projects Agency (DARPA), the US Air Force and the Singapore-MIT Alliance (Singapore government).
- The system's technical architecture makes it possible to produce **a vast number of repetitive calculations** on neighbouring structure simulation problems in a **very short time** (on the order of 1 ms per calculation), with different numerical values for the input parameters. The **savings** in calculation time for this type of problem is on the order of a **factor of 1,000** to **100,000** and more compared with traditional finite element numerical simulation tools.
- This extremely thigh calculation speed on closely-related problems opens up the possibility of radically **new functions** such as:
  - **reliable** and **accurate** (based upon finite element simulation) **instrumentation and control** of on-board (in aircraft, automobiles, pilot less machines) and industrial (ovens, rolling mills, chemical reactors) systems
  - **reliable** and **accurate optimisation** and sizing, during the initial phases of thermal, mechanical, fluid and acoustic design and development of industrial objects.
- A fundamental characteristic of Christophe Prud'homme's system is the **error test**. The results are accompanied by a certificate of reliability delimiting the error applied to the calculation, computed with respect to the output of the underlying Finite Element Method used. The engineer can therefore use the results in total confidence.
- Major benefit comes from pre determination of error bounds or computation time **before calculation**. Typically :
  - **real time** constraint implies time bounding
  - high **reliability** constraint implies accuracy and error control
- Christophe Prud'homme's real-time simulation system produces outstanding performances in calculation times and only requires limited online calculation resources (typically a 500 MHz Pentium PC) due to it **storing** previous calculation results and **error quantification** in a **simulation storage area**.
- Christophe Prud'homme's method is based on:
  - A **reduction of order** of the simulation problem by a "reduced base" **mathematical** method. The solution space size therefore moves typically from several tens of thousands to something of the order of 10. Once the problem is thus reduced, the

traditional fully reliable **finite element** simulation algorithms are activated **offline** and the generic results obtained are stored in the simulation storage area. The reduction of order is a complex mathematical operation necessitating top-level **scientific** skills. It relates to a specific type of problem, defined particularly by the **geometry** of the objects being simulated , the **physical characteristics** of the material and / or more generally the parameterisation of the underlying model equations.

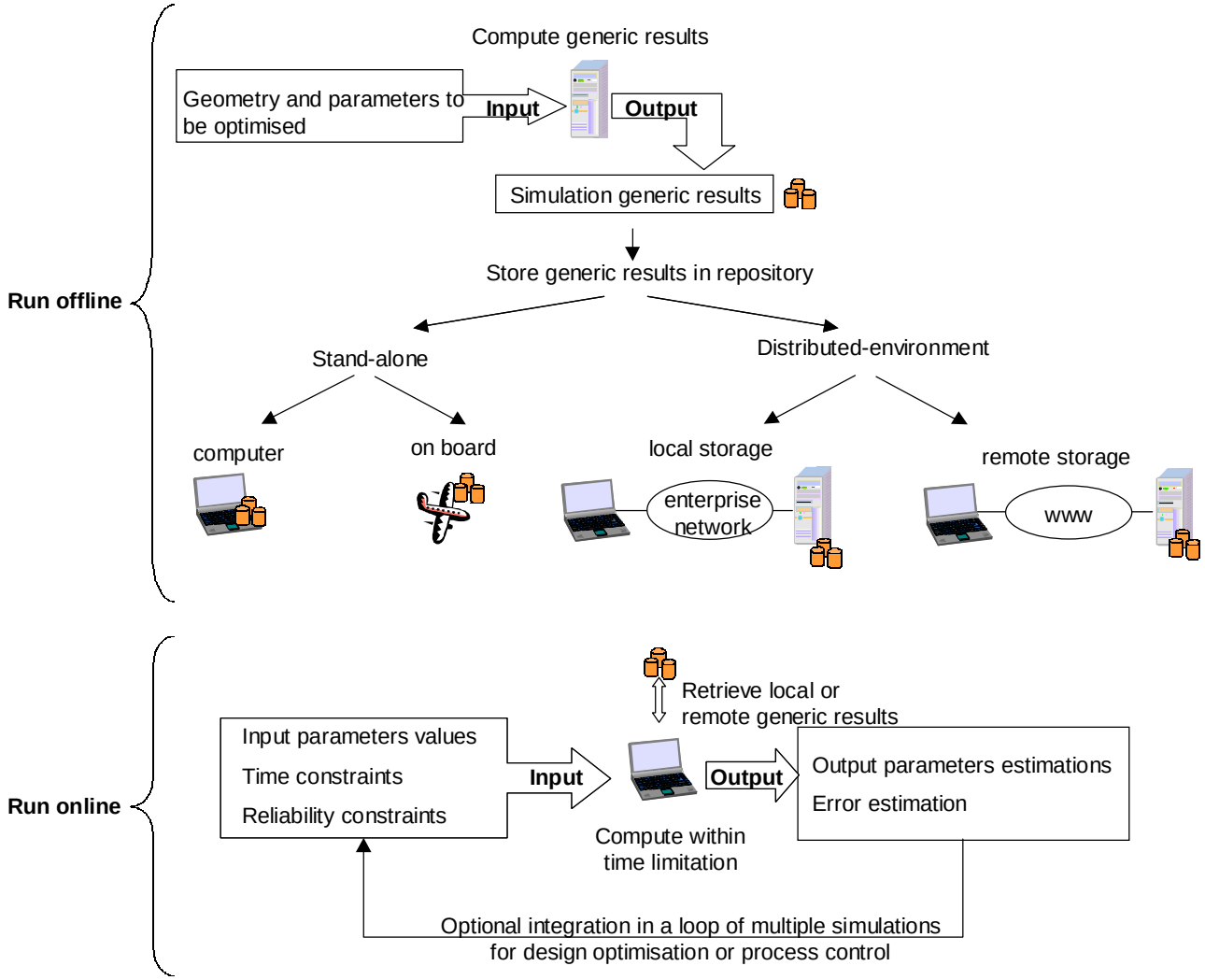
- **Computerized** automation of the reduction of order mathematical problem. In this way, a top-level mathematician is no longer necessary to adapt to a new geometry of the objects being simulated. The tool therefore processes a very wide range of problem classes, by offering the user the chance to define an object's geometry himself. This automation resembles the enabling factor of an **industrial** product, likely to be used by a **significant number** of engineers who are not specialized in the advanced techniques of numerical analysis.
- **Separation** between calculations requiring real-time processing (on-line), for each numerical parameter input vector, and those applying to the simulation storage area (off-line). The system deduces the output vector and a final quantitative error estimation in around **1 ms**, against several minutes of traditional tool use.
- The use of results from repeated simulation calculations by **traditional** optimisation and manufacturing automation tools, and instrumentation and control systems.

❑ The Christophe Prud'homme method can simulate the following **physical phenomena**:

- heat transfer
- elasticity (linear and non linear)
- fluid dynamics
- acoustics
- and more generally those represented by elliptic equations

❑ Integration into the design and optimisation chain goes through the following steps:

- Draw the geometry (eventually with CAD software) of the mechanical piece
- Define the independent boundary conditions that may vary from one simulation run to the next (pressure, heat flow, vibration amplitude & frequency, etc.)
- Select the independent parameters to be optimised :
  - physical characteristics of the material
  - independent geometrical parameters
  - nature of allowable geometrical transformations (stretch, skew, etc, …)
- Run the simulation generator "offline" once and run "online" simulations as needed, for each set of (boundary conditions x optimisation parameters) to be investigated
- Retrieve cost function parameters (temperature, heat flow, maximum deflection, etc.) for each simulation
- Eventually return parameters values for optimal configuration to CAD software



- The Christophe Prud'homme Software is **interoperable** with languages as Perl, Python, Java, C and C++ and platforms as Matlab (MathWorks), modeFrontier (Esteco). The Software can **stand alone** (plug-in style) or be in a **distributed environment**

Chapter 3

Research Projects and Perspectives

I very briefly present in this chapter some research projects and perspectives. The long-term prospects are to build a complete mathematical and data-processing framework for the reduced basis methods for all kinds of equations to which these methods apply. The computing framework is built in such a way that these components are easily reusable in other contexts.

3.1 Reduced Basis Methods

I work on four research orientations around the reduced bases: *(i)* the adaptive generation of the reduced bases, *(ii)* the use of methods of stabilization, in particular methods of the type penalization intérieure [17], in the context of the reduced bases, *(iii)* the development of the methods of the bases reduced in the context of the incompressible equations of Euler and finally *(iv)* the application of the methods to industrial problems and in particular for the design of components of planes.

The two points *(iii)* and *(iv)* were proposed in a European project STREP Excalade[16] whose principal partners will be Dassault, the DLR, the NLR, the University of Sheffield and a company Suisse SMR. This project will make it possible to clarify the needs for development and research around the methods of the bases reduced in an industrial context.

3.2 Mathematical Kernel and Language for the Resolution of PDE

Current and future work around the language includes:

- the application to the reduced basis methods and the systematic and automatic affine decomposition of the linear and bilinear forms. Already this language is used in the context the reduced basis methods and made it possible to obtain the results presented in section 1.9, however the generation of the affine decomposition — standard or using magic points — is not automated as in the technology presented in section 2.4.

[16] Flexible, Competitive and Agile Design of Aircraft Low-Noise-Emission in A Distributed Environment. Recorded near the European Commission under the number: FP6-031016

- the direction of a student on the implementation of the spectral methods in the context of the language.
- the use of the language in a context of methods of domain decomposition and in parallel — one can already implement the overlapping Schwarz method rather easily.
- the use of the language within the framework of multiphysics applications and multiscales for the project LifeV

Part II

Publications: Reduced Basis Methods

Chapter 4

Reliable Real-Time Solution of Parametrized Partial Differential Equations: Reduced-Basis Output Bound Methods

Authors: C. Prud'homme, D.V. Rovas, K. Veroy, L. Machiels, Y. Maday, A.T. Patera and G. Turinici.

4.1 Introduction

The optimization, control, and characterization of an engineering component or system requires the prediction of certain "quantities of interest," or performance metrics, which we shall denote *outputs* — for example deflections, maximum stresses, maximum temperatures, heat transfer rates, flowrates, or lift and drags. These outputs are typically expressed as functionals of field variables associated with a parametrized partial differential equation which describes the physical behavior of the component or system. The parameters, which we shall denote *inputs*, serve to identify a particular "configuration" of the component: these inputs may represent design or decision variables, such as geometry — for example, in optimization studies; control variables, such as actuator power — for example in real–time applications; or characterization variables, such as physical properties — for example in inverse problems. We thus arrive at an implicit *input–output* relationship, evaluation of which demands solution of the underlying partial differential equation.

Our goal is the development of computational methods that permit *rapid* and *reliable* evaluation of this partial-differential-equation-induced input-output relationship *in the limit of many queries* — that is, in the design, optimization, control, and characterization contexts. The "many query" limit has certainly received considerable attention: from "fast loads" or multiple right-hand side notions (e.g., [21, 25]) to matrix perturbation theories (e.g., [3, 85]) to continuation methods (e.g., [4, 71]). Our particular approach is based on the reduced–basis method, first introduced in the late 1970s for nonlinear structural analysis [5, 51], and subsequently developed more broadly in the 1980s and 1990s [13, 14, 26, 55, 58, 70]. The

reduced–basis method recognizes that the field variable is not, in fact, some arbitrary member of the infinite-dimensional solution space associated with the partial differential equation; rather, it resides, or "evolves," on a much lower–dimensional manifold induced by the parametric dependence.

The reduced–basis approach as earlier articulated is local in parameter space in both practice and theory. To wit, Lagrangian or Taylor approximation spaces for the low–dimensional manifold are typically defined relative to a particular parameter point; and the associated *a priori* convergence theory relies on asymptotic arguments in sufficiently small neighborhoods [26]. As a result, the computational improvements — relative to conventional (say) finite element approximation — are often quite modest [58]. Our work differs from these earlier efforts in several important ways: first, we develop (in some cases, provably) *global* approximation spaces; second, we introduce rigorous *a posteriori error estimators*; and third, we exploit *off–line/on–line* computational decompositions (see [13] for an earlier application of this strategy within the reduced–basis context). These three ingredients allow us — for the restricted but important class of "parameter-affine" problems — to reliably decouple the generation and projection stages of reduced–basis approximation, thereby effecting computational economies of several orders of magnitude.

In this expository review paper we focus on these new ingredients. In Section 2 we introduce an abstract problem formulation and several illustrative instantiations. In Section 3 we describe, for coercive symmetric problems and "compliant" outputs, the reduced–basis approximation; and in Section 4 we present the associated *a posteriori* error estimation procedures. In Section 5 we consider the extension of our approach to noncompliant outputs and nonsymmetric operators; eigenvalue problems; and, more briefly, noncoercive operators, parabolic equations, and non-affine problems. A description of the system architecture in which these numerical objects reside may be found in [79].

4.2 Problem Statement

4.2.1 Abstract Formulation

We consider a suitably regular domain $\Omega \subset \mathbb{R}^d$, $d = 1,\ 2$, or 3, and associated function space $X \subset H^1(\Omega)$, where $H^1(\Omega) = \{v \in L^2(\Omega),\ \nabla v \in (L^2(\Omega))^d\}$, and $L^2(\Omega)$ is the space of square integrable functions over Ω. The inner product and norm associated with X are given by $(\cdot\,,\,\cdot)_X$ and $\|\cdot\|_X = (\cdot\,,\,\cdot)^{1/2}$, respectively. We also define a parameter set $\mathcal{D} \in \mathbb{R}^P$, a particular point in which will be denoted μ. Note that Ω does *not* depend on the parameter.

We then introduce a bilinear form $a\colon\ X \times X \times \mathcal{D} \to \mathbb{R}$, and linear forms $f\colon\ X \to \mathbb{R}$, $\ell\colon\ X \to \mathbb{R}$. We shall assume that a is continuous, $a(w, v; \mu) \leq \gamma(\mu)\ \|w\|_X\ \|v\|_X \leq \gamma_0\ \|w\|_X\ \|v\|_X,\ \forall \mu \in \mathcal{D}$; furthermore, in Sections 4.3 and 4.4, we assume that a is coercive,

$$0 < \alpha_0 \leq\ \alpha(\mu) = \inf_{w \in X} \frac{a(w, w; \mu)}{\|w\|_X^2}, \quad \forall\, \mu \in \mathcal{D}, \tag{4.1}$$

and symmetric, $a(w, v; \mu) = a(v, w; \mu),\ \forall\, w, v \in X,\ \forall\, \mu \in \mathcal{D}$. We also require that our linear forms f and ℓ be bounded; in Sections 4.3 and 4.4 we additionally assume a "compliant" output, $f(v) = \ell(v),\ \forall\, v \in X$.

We shall also make certain assumptions on the parametric dependence of a, f, and ℓ. In particular, we shall suppose that, for some finite (preferably small) integer Q, a may be expressed as

$$a(w, v; \mu) \;=\; \sum_{q=1}^{Q} \sigma^q(\mu)\; a^q(w, v), \quad \forall\; w, v \in X,\; \forall\; \mu \in \mathcal{D}, \tag{4.2}$$

for some $\sigma^q\colon\; \mathcal{D} \to \mathbb{R}$ and $a^q\colon\; X \times X \to \mathbb{R}$, $q = 1, \ldots, Q$. This "separability," or "affine," assumption on the parameter dependence is crucial to computational efficiency; however, certain relaxations are possible — see Section 4.5.3.0. For simplicity of exposition, we assume that f and ℓ do not depend on μ; in actual practice, affine dependence is readily admitted.

Our abstract problem statement is then: for any $\mu \in \mathcal{D}$, find $s(\mu) \in \mathbb{R}$ given by

$$s(\mu) = \ell(u(\mu)), \tag{4.3}$$

where $u(\mu) \in X$ is the solution of

$$a(u(\mu), v; \mu) \;=\; f(v), \quad \forall\; v \in X. \tag{4.4}$$

In the language of the introduction, a is our partial differential equation (in weak form), μ is our parameter, $u(\mu)$ is our field variable, and $s(\mu)$ is our output. For simplicity of exposition, we may on occasion suppress the explicit dependence on μ.

4.2.2 Particular Instantiations

We indicate here a few instantiations of the abstract formulation; these will serve to illustrate the methods (for coercive, symmetric problems) of Sections 4.3 and 4.4.

A Thermal Fin

In this example we consider the two- and three-dimensional thermal fins shown in Figure 4.1; these examples may be (interactively) accessed on our web site[17]. The fins consist of a vertical central "post" of conductivity $\tilde{k}_0$ and four horizontal "subfins" of conductivity $\tilde{k}^i$, $i = 1, \ldots, 4$. The fins conduct heat from a prescribed uniform flux source $\tilde{q}''$ at the root $\tilde{\Gamma}_{\text{root}}$ through the post and large-surface-area subfins to the surrounding flowing air; the latter is characterized by a sink temperature $\tilde{u}_0$ and prescribed heat transfer coefficient $\tilde{h}$. The physical model is simple conduction: the temperature field in the fin, $\tilde{u}$, satisfies

$$\sum_{i=0}^{4} \int_{\tilde{\Omega}_i} \tilde{k}^i\; \tilde{\nabla}\tilde{u} \cdot \tilde{\nabla}\tilde{v} \;+\; \int_{\partial\tilde{\Omega}\setminus\tilde{\Gamma}_{\text{root}}} \tilde{h}\; (\tilde{u} - \tilde{u}_0)\; \tilde{v} \;=\; \int_{\tilde{\Gamma}_{\text{root}}} \tilde{q}''\; \tilde{v}, \quad \forall\; \tilde{v} \in \tilde{X} \equiv H^1(\tilde{\Omega}), \tag{4.5}$$

where $\tilde{\Omega}_i$ is that part of the domain with conductivity $\tilde{k}^i$, and $\partial\tilde{\Omega}$ denotes the boundary of $\tilde{\Omega}$.

We now (*i*) nondimensionalize the weak equations (4.5), and (*ii*) apply a continuous piecewise-affine transformation from $\tilde{\Omega}$ to a fixed (μ-independent) reference domain Ω [42].

[17] FIN2D: http://augustine.mit.edu/fin2d/fin2d.pdf and FIN3D: http://augustine.mit.edu/fin3d_1/fin3d_1.pdf

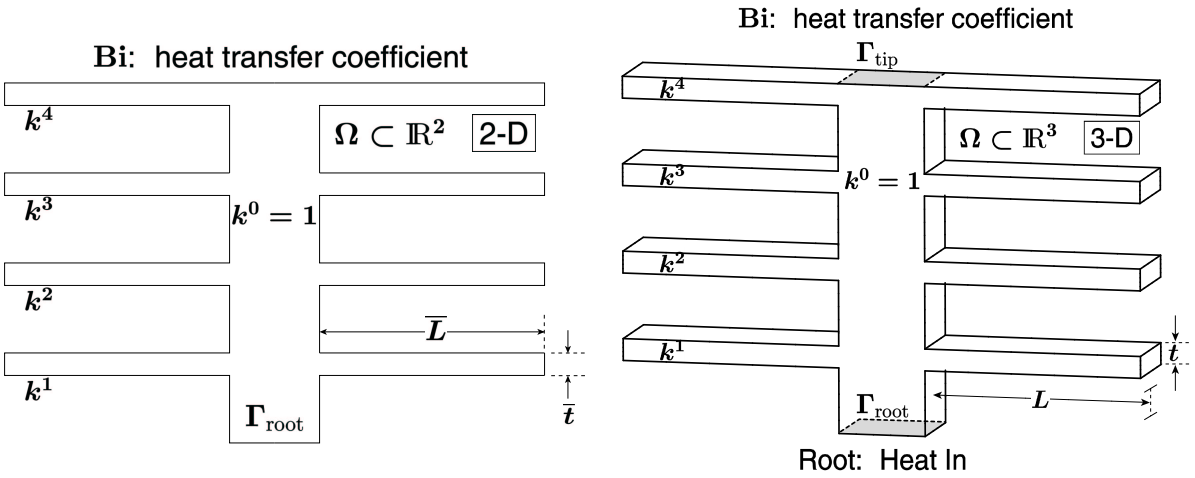


Figure 4.1: Two- and Three-Dimensional Thermal Fins.

The abstract problem statement (4.4) is then recovered for $\mu = \{k^1,\ k^2,\ k^3,\ k^4,\ \text{Bi},\ L,\ t\}$, $\mathcal{D} = [0.1, 10.0]^4 \times [0.01, 1.0] \times [2.0, 3.0] \times [0.1 \times 0.5]$, and $P = 7$; here $k^1, \ldots, k^4$ are the thermal conductivities of the "subfins" (see Figure 4.1) relative to the thermal conductivity of the fin base; Bi is a nondimensional form of the heat transfer coefficient; and, L, t are the length and thickness of each of the "subfins" relative to the length of the fin root $\tilde{\Gamma}_{\text{root}}$. It is readily verified that a is continuous, coercive, and symmetric; and that the "affine" assumption (4.2) obtains for $Q = 16$ (two-dimensional case) and $Q = 25$ (three-dimensional case). Note that the geometric variations are reflected, via the mapping, in the $\sigma^q(\mu)$.

For our output of interest, $s(\mu)$, we consider the average temperature of the root of the fin nondimensionalized relative to $\tilde{q}''$, $\tilde{k}^0$, and the length of the fin root. This output may be expressed as $s(\mu) = \ell(u(\mu))$, where $\ell(v) = \int_{\Gamma_{\text{root}}} v$. It is readily shown that this output functional is bounded and also "compliant": $\ell(v) = f(v),\ \forall v \in X$.

A Truss Structure

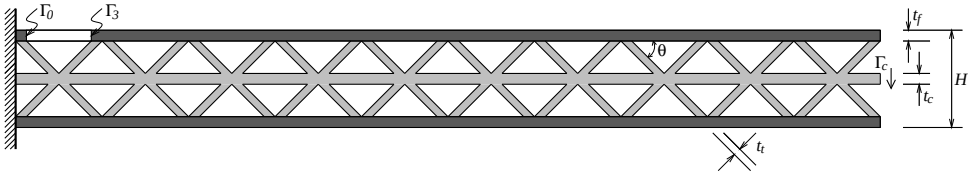


Figure 4.2: A Truss Structure

We consider a prismatic microtruss structure [24, 84] shown in Figure 4.2; this example may be (interactively) accessed on our web site[18]. The truss consists of a frame (upper and

[18] TRUSS: http://augustine.mit.edu/simple_truss/simple_truss.pdf

lower faces, in dark gray) and a core (trusses and middle sheet, in light gray). The structure transmits a force per unit depth $\tilde{F}$ uniformly distributed over the tip of the middle sheet $\tilde{\Gamma}_3$ through the truss system to the fixed left wall $\tilde{\Gamma}_0$. The physical model is simple plane–strain (two-dimensional) linear elasticity: the displacement field $u_i,\ i = 1, 2$, satisfies

$$\int_{\tilde{\Omega}} \frac{\partial \tilde{v}_i}{\partial \tilde{x}_j} \tilde{E}_{ijkl} \frac{\partial \tilde{u}_k}{\partial \tilde{x}_l} = -\left(\frac{\tilde{F}}{\tilde{t}_c}\right) \int_{\tilde{\Gamma}_3} \tilde{v}_2, \quad \forall\ v \in \tilde{X}, \tag{4.6}$$

where $\tilde{\Omega}$ is the truss domain, $\tilde{E}_{ijkl}$ is the elasticity tensor, and $\tilde{X}$ refers to the set of functions in $H^1(\tilde{\Omega})$ which vanish on $\tilde{\Gamma}_0$. We assume summation over repeated indices.

We now (*i*) nondimensionalize the weak equations (4.6), and (*ii*) apply a continuous piecewise-affine transformation from $\tilde{\Omega}$ to a fixed (μ-independent) reference domain Ω. The abstract problem statement (4.4) is then recovered for $\mu = \{t_f,\ t_t,\ H,\ \theta\}, \mathcal{D} = [0.08, 1.0] \times [0.2, 2.0] \times [4.0, 10.0] \times [30.0°, 60.0°]$, and $P = 4$; here t_f and t_t are the thicknesses of the frame and trusses (normalized relative to $\tilde{t}_c$), respectively; H is the total height of the microtruss (normalized relative to $\tilde{t}_c$); and θ is the angle between the trusses and the faces. The Poisson's ratio, $\nu = 0.3$, and the frame and core Young's moduli, $E_f = 75$ GPa and $E_c = 200$ GPa, respectively, are held fixed. It is readily verified that a is continuous, coercive, and symmetric; and that the "affine" assumption (4.2) obtains for $Q = 44$.

Our outputs of interest are (*i*) the average downward deflection (compliance) at the core tip, Γ_3, nondimensionalized by $\tilde{F}/\tilde{E}_f$; and (*ii*) the average normal stress across the critical (yield) section denoted Γ_1^s in Figure 4.2. These compliance and noncompliance outputs can be expressed as $s^1(\mu) = \ell^1(u(\mu))$ and $s^2(\mu) = \ell^2(u(\mu))$, respectively, where $\ell^1(v) = -\int_{\Gamma_3} v_2$, and

$$\ell^2(v) = \frac{1}{t_f} \int_{\Omega^s} \frac{\partial \chi_i}{\partial x_j} E_{ijkl} \frac{\partial u_k}{\partial x_l}$$

are bounded linear functionals; here χ_i is any suitably smooth function in $H^1(\Omega^s)$ such that $\chi_i \hat{n}_i = 1$ on Γ_1^s and $\chi_i \hat{n}_i = 0$ on Γ_2^s, where $\hat{n}$ is the unit normal. Note that $s^1(\mu)$ is a compliant output, whereas $s^2(\mu)$ is "noncompliant."

4.3 Reduced-Basis Approach

We recall that in this section, as well as in Section 4.4, we assume that a is continuous, coercive, symmetric, and affine in μ — see (4.2); and that $\ell(v) = f(v)$, which we denote "compliance."

4.3.1 Reduced–Basis Approximation

We first introduce a sample in parameter space, $S_N = \{\mu_1, \ldots, \mu_N\}$, where $\mu_i \in \mathcal{D},\ i = 1, \ldots, N$; see Section 4.3.2.0 for a brief discussion of point distribution. We then define our Lagrangian [58] reduced–basis approximation space as $W_N = \text{span}\ \{\zeta_n \equiv u(\mu_n),\ n = 1, \ldots, N\}$, where $u(\mu_n) \in X$ is the solution to (4.4) for $\mu = \mu_n$. In actual practice, $u(\mu_n)$ is replaced by an appropriate finite element approximation on a suitably fine truth mesh; we shall discuss the associated computational implications in Section 4.3.3. Our reduced–basis

approximation is then: for any $\mu \in \mathcal{D}$, find $s_N(\mu) = \ell(u_N(\mu))$, where $u_N(\mu) \in W_N$ is the solution of

$$a(u_N(\mu), v; \mu) \;=\; \ell(v), \quad \forall\, v \in W_N. \tag{4.7}$$

Non-Galerkin projections are briefly described in Section 4.5.3.0.

4.3.2 *A Priori* Convergence Theory

Optimality

We consider here the convergence rate of $u_N(\mu) \to u(\mu)$ and $s_N(\mu) \to s(\mu)$ as $N \to \infty$. To begin, it is standard to demonstrate optimality of $u_N(\mu)$ in the sense that

$$\|u(\mu) - u_N(\mu)\|_X \le \sqrt{\frac{\gamma(\mu)}{\alpha(\mu)}} \inf_{w_N \in W_N} \|u(\mu) - w_N\|_X \ . \tag{4.8}$$

(We note that, in the coercive case, stability of our ("conforming") discrete approximation is not an issue; the noncoercive case is decidedly more delicate (see Section 4.5.3.0).) Furthermore, for our compliance output,

$$s(\mu) = s_N(\mu) + \ell(u - u_N) = s_N(\mu) + a(u, u - u_N; \mu) = s_N(\mu) + a(u - u_N, u - u_N; \mu) \tag{4.9}$$

from symmetry and Galerkin orthogonality. It follows that $s(\mu) - s_N(\mu)$ converges as the square of the error in the best approximation and, from coercivity, that $s_N(\mu)$ is a lower bound for $s(\mu)$.

Best Approximation

It now remains to bound the dependence of the error in the best approximation as a function of N. At present, the theory is restricted to the case in which $P = 1$, $\mathcal{D} = [0, \mu_{\max}]$, and

$$a(w, v; \mu) = a_0(w, v) + \mu a_1(w, v), \tag{4.10}$$

where a_0 is continuous, coercive, and symmetric, and a_1 is continuous, positive semi-definite ($a_1(w, w) \ge 0$, $\forall\, w \in X$), and symmetric. This model problem (4.10) is rather broadly relevant, for example to variable orthotropic conductivity, variable rectilinear geometry, variable piecewise-constant conductivity, and variable Robin boundary conditions.

We now suppose that the μ_n, $n = 1, \ldots, N$, are logarithmically distributed in the sense that

$$\ln\left(\overline{\lambda}\,\mu_n + 1\right) = \frac{n-1}{N-1} \ln\left(\overline{\lambda}\,\mu_{\max} + 1\right), \quad n = 1, \ldots, N, \tag{4.11}$$

where $\overline{\lambda}$ is an upper bound for the maximum eigenvalue of a_1 relative to a_0. (Note $\overline{\lambda}$ is perforce bounded thanks to our assumption of continuity and coercivity; the possibility of a continuous spectrum does not, in practice, pose any problems.) We can then prove [46] that, for $N > N_{\text{crit}} \equiv e \ln(\overline{\lambda}\,\mu_{\max} + 1)$,

$$\inf_{w_N \in W_N} \|u(\mu) - w_N(\mu)\|_X \le (1 + \mu_{\max}\,\overline{\lambda})\, \|u(0)\|_X \, \exp\left\{\frac{-(N-1)}{(N_{\text{crit}} - 1)}\right\}, \quad \forall\, \mu \in \mathcal{D} \ . \tag{4.12}$$

We observe exponential convergence, uniformly (globally) for all μ in $\mathcal{D}$, with only very weak (logarithmic) dependence on the range of the parameter $(\mu_{\max})$. (Note the constants in (4.12) are for the particular case in which $(\cdot,\cdot)_X = a_0(\cdot,\cdot)$.)

The proof exploits a parameter–space (non-polynomial) interpolant as a surrogate for the Galerkin approximation. As a result, the bound is not always "sharp": we observe many cases in which the Galerkin projection is considerably better than the associated interpolant; optimality (4.8) chooses to "illuminate" only certain points μ_n, automatically selecting a best "sub–approximation" amongst all (combinatorially many) possibilities — we thus see why reduced–basis *state-space* approximation of $s(\mu)$ via $u(\mu)$ is preferred to simple *parameter-space* interpolation of $s(\mu)$ ("connecting the dots") via $(\mu_n, s(\mu_n))$ pairs. We note, however, that the logarithmic point distribution (4.11) implicated by our interpolant–based arguments is *not* simply an artifact of the proof: in numerous numerical tests, the logarithmic distribution performs considerably (and in many cases, provably) better than other more obvious candidates, in particular for large ranges of the parameter. Fortunately, the convergence rate is not *too* sensitive to point selection: the theory only requires a log "on the average" distribution [46]; and, in practice, $\overline{\lambda}$ need not be a sharp upper bound.

The result (4.12) is certainly tied to the particular form (4.10) and associated regularity of $u(\mu)$. However, we do observe similar exponential behavior for more general operators; and, most importantly, the exponential convergence rate degrades only very slowly with increasing parameter dimension, P. We present in Table 4.1 the error $|s(\mu) - s_N(\mu)|/s(\mu)$ as a function of N, at a particular representative point μ in $\mathcal{D}$, for the two-dimensional thermal fin problem of Section 4.2.2.0; we present similar data in Table 4.2 for the truss problem of Section 4.2.2.0. In both cases, since tensor-product grids are prohibitively profligate as P increases, the μ_n are chosen "log-randomly" over $\mathcal{D}$: we sample from a multivariate uniform probability density on $\log(\mu)$. We observe that, for both the thermal fin ($P = 7$) and truss ($P = 4$) problems, the error is remarkably small even for very small N; and that, in both cases, very rapid convergence obtains as $N \to \infty$. We do not yet have any theory for $P > 1$. But certainly the Galerkin optimality plays a central role, automatically selecting "appropriate" scattered-data subsets of S_N and associated "good" weights so as to mitigate the curse of dimensionality as P increases; and the log–random point distribution is also important — for example, for the truss problem of Table 4.2, a *non–logarithmic* uniform random point distribution for S_N yields errors which are larger by factors of 20 and 10 for $N = 30$ and 80, respectively.

4.3.3 Computational Procedure

The theoretical and empirical results of Sections 4.3.1 and 4.3.2 suggest that N may, indeed, be chosen very small. We now develop off–line/on–line computational procedures that exploit this dimension reduction.

We first express $u_N(\mu)$ as

$$u_N(\mu) = \sum_{j=1}^{N} u_{N\,j}(\mu)\,\zeta_j = (\underline{u}_N(\mu))^T\,\underline{\zeta}, \tag{4.13}$$

where $\underline{u}_N(\mu) \in \mathbb{R}^N$; we then choose for test functions $v = \zeta_i$, $i = 1, \ldots, N$. Inserting these

N	$\lvert s(\mu) - s_N(\mu)\rvert / s(\mu)$	$\Delta_N(\mu)/s(\mu)$	$\eta_N(\mu)$
10	1.29×10^{-2}	8.60×10^{-2}	2.85
20	1.29×10^{-3}	9.36×10^{-3}	2.76
30	5.37×10^{-4}	4.25×10^{-3}	2.68
40	8.00×10^{-5}	5.30×10^{-4}	2.86
50	3.97×10^{-5}	2.97×10^{-4}	2.72
60	1.34×10^{-5}	1.27×10^{-4}	2.54
70	8.10×10^{-6}	7.72×10^{-5}	2.53
80	2.56×10^{-6}	2.24×10^{-5}	2.59

Table 4.1: Error, error bound (Method I), and effectivity as a function of N, at a particular representative point $\mu \in \mathcal{D}$, for the two-dimensional thermal fin problem (compliant output).

N	$\lvert s(\mu) - s_N(\mu)\rvert / s(\mu)$	$\Delta_N(\mu)/s(\mu)$	$\eta_N(\mu)$
10	3.26×10^{-2}	6.47×10^{-2}	1.98
20	2.56×10^{-4}	4.74×10^{-4}	1.85
30	7.31×10^{-5}	1.38×10^{-4}	1.89
40	1.91×10^{-5}	3.59×10^{-5}	1.88
50	1.09×10^{-5}	2.08×10^{-5}	1.90
60	4.10×10^{-6}	8.19×10^{-6}	2.00
70	2.61×10^{-6}	5.22×10^{-6}	2.00
80	1.19×10^{-6}	2.39×10^{-6}	2.00

Table 4.2: Error, error bound (Method II), and effectivity as a function of N, at a particular representative point $\mu \in \mathcal{D}$, for the truss problem (compliant output).

representations into (4.7) yields the desired algebraic equations for $\underline{u}_N(\mu) \in \mathbb{R}^N$,

$$\underline{A}_N(\mu)\, \underline{u}_N(\mu) \;=\; \underline{F}_N, \tag{4.14}$$

in terms of which the output can then be evaluated as $s_N(\mu) = \underline{F}_N^T\, \underline{u}_N(\mu)$. Here $\underline{A}_N(\mu) \in \mathbb{R}^{N\times N}$ is the SPD matrix with entries $A_{N\,i,j}(\mu) \equiv a(\zeta_j, \zeta_i; \mu)$, $1 \le i, j \le N$, and $\underline{F}_N \in \mathbb{R}^N$ is the "load" (and "output") vector with entries $F_{N\,i} \equiv f(\zeta_i)$, $i = 1, \ldots, N$.

We now invoke (4.2) to write

$$A_{N\,i,j}(\mu) = a(\zeta_j, \zeta_i; \mu) = \sum_{q=1}^{Q} \sigma^q(\mu)\, a^q(\zeta_j, \zeta_i)\ , \tag{4.15}$$

or

$$\underline{A}_N(\mu) = \sum_{q=1}^{Q} \sigma^q(\mu)\, \underline{A}_N^q\ ,$$

where the $\underline{A}_N \in \mathbb{R}^{N\times N}$ are given by $A^q_{N\,i,j} = a^q(\zeta_j, \zeta_i)$, $i \le i, j \le N$, $1 \le q \le Q$. The off–line/on–line decomposition is now clear. In the *off–line* stage, we compute the $u(\mu_n)$ and form the $\underline{A}_N^q$ and $\underline{F}_N$: this requires N (expensive) "$\mathcal{A}$" finite element solutions and $O(QN^2)$ finite-element-vector inner products. In the *on–line* stage, for any given new μ, we first form $\underline{A}_N$ from (4.15), then solve (4.14) for $\underline{u}_N(\mu)$, and finally evaluate $s_N(\mu) = \underline{F}_N^T\, \underline{u}_N(\mu)$: this requires $O(QN^2) + O(\frac{2}{3}N^3)$ operations and $O(QN^2)$ storage.

Thus, as required, the incremental, or marginal, cost to evaluate $s_N(\mu)$ for any given new μ — as proposed in a design, optimization, or inverse-problem context — is very small: first, because N is very small, typically $O(10)$ — thanks to the good convergence properties of W_N; and second, because (4.14) can be very rapidly assembled and inverted — thanks to the off–line/on–line decomposition (see [13] for an earlier application of this strategy within the reduced–basis context). For the problems discussed in this paper, the resulting computational savings relative to standard (well-designed) finite-element approaches are significant — at least $O(10)$, typically $O(100)$, and often $O(1000)$ or more.

4.4 A Posteriori Error Estimation: Output Bounds

From Section 4.3 we know that, in theory, we can obtain $s_N(\mu)$ very inexpensively: the on–line computational effort scales as $O(\frac{2}{3}N^3) + O(QN^2)$; and N can, *in theory*, be chosen quite small. However, *in practice*, we do not know *how* small N can be chosen: this will depend on the desired accuracy, the selected output(s) of interest, and the particular problem in question; in some cases $N = 5$ may suffice, while in other cases, $N = 100$ may still be insufficient. In the face of this uncertainty, either too many or too few basis functions will be retained: the former results in computational inefficiency; the latter in unacceptable uncertainty — particularly egregious in the decision contexts in which reduced–basis methods typically serve. We thus need *a posteriori* error estimators for s_N. Surprisingly, *a posteriori* error estimation has received relatively little attention within the reduced–basis framework [51], even though reduced–basis methods are particularly in need of accuracy assessment: the spaces are *ad hoc*

and pre-asymptotic, thus admitting relatively little intuition, "rules of thumb," or standard approximation notions.

Recall that, in this section, we continue to assume that a is coercive and symmetric, and that ℓ is "compliant."

4.4.1 Method I

The approach described in this section is a particular instance of a general "variational" framework for *a posteriori* error estimation of outputs of interest. However, the reduced-basis instantiation described here differs significantly from earlier applications to finite element discretization error [43, 40] and iterative solution error [54] both in the choice of (energy) relaxation and in the associated computational artifice.

Formulation

We assume that we are given a positive function $g(\mu) : \mathcal{D} \rightarrow \mathbb{R}_+$, and a continuous, coercive, symmetric (μ-independent) bilinear form $\hat{a} : X \times X \rightarrow \mathbb{R}$, such that

$$\underline{\alpha}_0 \|v\|_X^2 \;\leq\; g(\mu)\; \hat{a}(v,v) \;\leq\; a(v,v;\mu), \quad \forall\, v \in X, \forall\, \mu \in \mathcal{D} \tag{4.16}$$

for some positive real constant $\underline{\alpha}_0$. We then find $\hat{e}(\mu) \in X$ such that

$$g(\mu)\; \hat{a}(\hat{e}(\mu), v) \;=\; R(v; u_N(\mu); \mu), \quad \forall v \in X, \tag{4.17}$$

where for a given $w \in X$, $R(v;w;\mu) = \ell(v) - a(w,v;\mu)$ is the weak form of the residual. Our lower and upper output estimators are then evaluated as

$$s_N^-(\mu) \;\equiv\; s_N(\mu), \text{ and } s_N^+(\mu) \equiv s_N(\mu) + \Delta_N(\mu), \tag{4.18}$$

respectively, where

$$\Delta_N(\mu) \;\equiv\; g(\mu)\; \hat{a}(\hat{e}(\mu), \hat{e}(\mu)) \tag{4.19}$$

is the estimator gap.

Properties

We shall prove in this section that $s_N^-(\mu) \leq s(\mu) \leq s_N^+(\mu)$, and hence that $|s(\mu) - s_N(\mu)| = s(\mu) - s_N(\mu) \leq \Delta_N(\mu)$. Our lower and upper output estimators are thus lower and upper output *bounds*; and our output estimator gap is thus an output *bound* gap — a rigorous bound for the error in the output of interest. It is also critical that $\Delta_N(\mu)$ be a relatively *sharp* bound for the true error: a poor (overly large) bound will encourage us to refine an approximation which is, in fact, already adequate — with a corresponding (unnecessary) increase in off–line and on–line computational effort. We shall prove in this section that $\Delta_N(\mu) \leq \frac{\gamma_0}{\underline{\alpha}_0}(s(\mu) - s_N(\mu))$, where γ_0 and $\underline{\alpha}_0$ are the N-*independent* a-continuity and $g(\mu)\hat{a}$-coercivity constants defined earlier. Our two results of this section can thus be summarized as

$$1 \;\leq\; \eta_N(\mu) \;\leq\; \text{Const}, \quad \forall\, N, \tag{4.20}$$

where

$$\eta_N(\mu) \;=\; \frac{\Delta_N(\mu)}{s(\mu) - s_N(\mu)} \tag{4.21}$$

is the *effectivity*, and Const is a constant independent of N. We shall denote the left (bounding property) and right (sharpness property) inequalities of (4.20) as the lower effectivity and upper effectivity inequalities, respectively.

We first prove the lower effectivity inequality (bounding property): $s_N^-(\mu) \le s(\mu) \le s_N^+(\mu),\ \forall \mu \in \mathcal{D}$, for $s_N^-(\mu)$ and $s_N^+(\mu)$ defined in (4.18). The lower bound property follows directly from Section 4.3.2.0. To prove the upper bound property, we first observe that $R(v; u_N; \mu) = a(u(\mu) - u_N(\mu), v; \mu) = a(e(\mu), v; \mu)$, where $e(\mu) \equiv u(\mu) - u_N(\mu)$; we may thus rewrite (4.17) as $g(\mu)\hat{a}(\hat{e}(\mu), v) = a(e(\mu), v; \mu)\ \forall\ v \in X$. We thus obtain

$$\begin{aligned} g(\mu)\hat{a}(\hat{e},\hat{e}) &= g(\mu)\hat{a}(\hat{e}-e,\hat{e}-e) + 2g(\mu)\hat{a}(\hat{e},e) - g(\mu)\hat{a}(e,e) \\ &= g(\mu)\hat{a}(\hat{e}-e,\hat{e}-e) + (a(e,e;\mu) - g(\mu)\hat{a}(e,e)) + a(e,e;\mu) \\ &\ge a(e,e;\mu) \end{aligned} \tag{4.22}$$

since $g(\mu)\ \hat{a}(\hat{e}(\mu) - e(\mu), \hat{e}(\mu) - e(\mu)) \ge 0$ and $a(e(\mu), e(\mu); \mu) - g(\mu)\ \hat{a}(e(\mu), e(\mu)) \ge 0$ from (4.16). Invoking (4.9) and (4.22), we then obtain $s(\mu) - s_N(\mu) = a(e(\mu), e(\mu); \mu) \le g(\mu)\ \hat{a}(\hat{e}(\mu), \hat{e}(\mu))$; and thus $s(\mu) \le s_N(\mu) + g(\mu)\ \hat{a}(\hat{e}(\mu), \hat{e}(\mu)) \equiv s_N^+(\mu)$, as desired.

We next prove the upper effectivity inequality (sharpness property):

$$\eta_N(\mu) = \frac{\Delta_N(\mu)}{s(\mu) - s_N(\mu)} \le \frac{\gamma_0}{\underline{\alpha}_0}, \quad \forall\ N.$$

To begin, we appeal to a-continuity and $g(\mu)\hat{a}$-coercivity to obtain

$$a(\hat{e}(\mu), \hat{e}(\mu); \mu) \;\le\; \frac{\gamma_0\ g(\mu)}{\underline{\alpha}_0}\hat{a}(\hat{e}(\mu), \hat{e}(\mu)). \tag{4.23}$$

But from the modified error equation (4.17) we know that $g(\mu)\hat{a}(\hat{e}(\mu), \hat{e}(\mu)) = R(\hat{e}(\mu); \mu) = a(e(\mu), \hat{e}(\mu); \mu)$. Invoking the Cauchy-Schwartz inequality, we obtain

$$g(\mu)\hat{a}(\hat{e},\hat{e}) \;=\; a(e,\hat{e};\mu) \;\le\; (a(\hat{e},\hat{e};\mu))^{1/2}(a(e,e;\mu))^{1/2} \;\le\; \left(\frac{\gamma_0}{\underline{\alpha}_0}\right)^{1/2} (g(\mu)\ \hat{a}(\hat{e},\hat{e}))^{1/2}\ (a(e,e;\mu))^{1/2};$$

the desired result then directly follows from (4.19) and (4.9).

We now provide empirical evidence for (4.20). In particular, we present in Table 4.1 the bound gap and effectivities for the thermal fin example. Clearly $\eta_N(\mu)$ is always greater than unity for any N, and bounded — indeed, quite close to unity — as $N \to \infty$.

Computational Procedure

Finally, we turn to the computational artifice by which we can efficiently compute $\Delta_N(\mu)$ in the on–line stage of our procedure. We again exploit the affine parameter dependence, but now in a less transparent fashion. To begin, we rewrite the "modified" error equation, (4.17), as

$$\hat{a}(\hat{e}(\mu), v) \;=\; \frac{1}{g(\mu)}\left(\ell(v) - \sum_{q=1}^{Q}\sum_{j=1}^{N} \sigma^q(\mu) u_{N\,j}(\mu) a^q(\zeta_j, v)\right), \quad \forall\ v \in X,$$

where we have appealed to our reduced–basis approximation (4.13) and the affine decomposition (4.2). It is immediately clear from linear superposition that we can express $\hat{e}(\mu)$ as

$$\hat{e}(\mu) = \frac{1}{g(\mu)} \left(\hat{z}_0 + \sum_{q=1}^{Q} \sum_{j=1}^{N} \sigma^q(\mu) u_{N\,j}(\mu) \hat{z}_j^q \right), \tag{4.24}$$

where $\hat{z}_0 \in X$ satisfies $\hat{a}(\hat{z}_0, v) = \ell(v),\ \forall\ v \in X$, and $\hat{z}_j^q \in X, j = 1, \ldots, N,\ q = 1, \ldots, Q$, satisfies $\hat{a}(\hat{z}_j^q, v) = -a^q(\zeta_j, v),\ \forall\ v \in X$. Inserting (4.24) into our expression for the upper bound, $s_N^+(\mu) = s_N(\mu) + g(\mu)\hat{a}(\hat{e}(\mu), \hat{e}(\mu))$, we obtain

$$\begin{aligned} s_N^+(\mu) = s_N(\mu) + \frac{1}{g(\mu)} \Bigg(& c_0 + 2 \sum_{q=1}^{Q} \sum_{j=1}^{N} \sigma^q(\mu) u_{N\,j}(\mu) \Lambda_j^q + \\ & \sum_{q=1}^{Q} \sum_{q'=1}^{Q} \sum_{j=1}^{N} \sum_{j'=1}^{N} \sigma^q(\mu) \sigma^{q'}(\mu) u_{N\,j}(\mu) u_{N\,j'}(\mu) \Gamma_{jj'}^{qq'} \Bigg) \end{aligned} \tag{4.25}$$

where $c_0 = \hat{a}(\hat{z}_0, \hat{z}_0)$, $\Lambda_j^q = \hat{a}(\hat{z}_0, \hat{z}_j^q)$, and $\Gamma_{jj\prime}^{qq\prime} = \hat{a}(\hat{z}_j^q, \hat{z}_{j\prime}^{q\prime})$.

The off–line/on–line decomposition should now be clear. In the *off–line* stage we compute $\hat{z}_0$ and $\hat{z}_j^q$, $j = 1, \ldots, N$, $q = 1, \ldots, Q$, and then form c_0, Λ_j^q, and $\Gamma_{jj'}^{qq'}$: this requires $QN + 1$ (expensive) "$\hat{a}$" finite element solutions, and $O(Q^2N^2)$ finite-element-vector inner products. In the *on–line* stage, for any given new μ, we evaluate s_N^+ as expressed in (4.25): this requires $O(Q^2N^2)$ operations and $O(Q^2N^2)$ storage (for c_0, Λ_j^q, and $\Gamma_{jj'}^{qq'}$). As for the computation of $s_N(\mu)$, the marginal cost for the computation of $s_N^{\pm}(\mu)$ for any given new μ is quite small — in particular, it is *independent* of the dimension of the truth finite element approximation space X.

There are a variety of ways in which the off–line/on–line decomposition and output error bounds can be exploited. A particularly attractive mode incorporates the error bounds into an on–line adaptive process, in which we successively approximate $s_N(\mu)$ on a sequence of approximation spaces $W_{N_j'} \subset W_N, N_j' = N_0 2^j$ — for example, $W_{N_j'}$ may contain the N_j' sample points of S_N closest to the new μ of interest — until $\Delta_{N_j'}$ is less than a specified error tolerance. This procedure both minimizes the on–line computational effort and reduces conditioning problems — while simultaneously ensuring accuracy and certainty.

The essential advantage of the approach described in this section is the guarantee of rigorous bounds. There are, however, certain disadvantages. The first set of disadvantages relates to the choice of $g(\mu)$ and $\hat{a}$. In many cases, simple inspection suffices: for example, in our thermal fin problem of Section 4.2.2.0, $g(\mu) = \min_{q=1,\ldots,Q} \sigma^q(\mu)$ and $\hat{a}(w, v) = \sum_{q=1}^{Q} a^q(w, v)$ yields the very good effectivities summarized in Table 4.1. In other cases, however, there is no self-evident (or readily computed [44]) good choice: for example, for the truss problem of Section 4.2.2.0, the existence of almost–pure rotations renders $g(\mu)$ very small relative to $\gamma(\mu)$, with corresponding detriment to $\eta_N(\mu)$. The second set of disadvantages relates to the computational expense — the $O(Q)$ off–line and the $O(Q^2)$ on–line scaling induced by (4.24) and (4.25), respectively. Both of these disadvantages are eliminated in the "Method II" to be discussed in the next section; however "Method II" only provides *asymptotic* bounds as $N \to \infty$. The choice thus depends on the relative importance of absolute certainty and computational efficiency.

4.4.2 Method II

As already indicated, Method I has certain limitations; we discuss here a Method II which addresses these limitations — albeit at the loss of complete certainty.

Formulation

To begin, we set $M > N$, and introduce a parameter sample $S_M = \{\mu_1, \ldots, \mu_M\}$ and associated reduced–basis approximation space $W_M = \text{span}\,\{\zeta_m \equiv u(\mu_m),\ m = 1, \ldots, M\}$; for both theoretical and practical reasons we require $S_N \subset S_M$ and therefore $W_N \subset W_M$. The procedure is very simple: we first find $u_M(\mu) \in W_M$ such that $a(u_M(\mu), v; \mu) = f(v), \forall\ v \in W_M$; we then evaluate $s_M(\mu) = \ell(u_M(\mu))$; and, finally, we compute our upper and lower output estimators as

$$s^-_{N,M}(\mu) = s_N(\mu), \qquad s^+_{N,M}(\mu) = s_N(\mu) + \Delta_{N,M}(\mu), \tag{4.26}$$

where $\Delta_{N,M}(\mu)$, the estimator bound gap, is given by

$$\Delta_{N,M}(\mu) = \frac{1}{\tau}\left(s_M(\mu) - s_N(\mu)\right) \tag{4.27}$$

for some $\tau \in (0, 1)$. The effectivity of the approximation is defined as

$$\eta_{N,M}(\mu) = \frac{\Delta_{N,M}(\mu)}{s(\mu) - s_N(\mu)}. \tag{4.28}$$

For our purposes here, we shall consider $M = 2N$.

Properties

As for Method I, we would like to prove the effectivity inequality $1 \le \eta_{N,2N}(\mu) \le \text{Const}$, $\forall\ N$. However, we will only be able to demonstrate an asymptotic form of this inequality. Furthermore, the latter shall require — and we shall make — the hypothesis that

$$\varepsilon_{N,2N}(\mu) \equiv \frac{s(\mu) - s_{2N}(\mu)}{s(\mu) - s_N(\mu)} \to 0, \quad \text{as } N \to \infty. \tag{4.29}$$

We note that the assumption (4.29) is certainly plausible: if our *a priori* bound of (4.12) in fact reflects asymptotic behavior, then $s(\mu) - s_N(\mu) \sim c_1 e^{-c_2 N}$, $s(\mu) - s_{2N}(\mu) \sim c_1 e^{-2c_2 N}$, and hence $\varepsilon_{N,2N}(\mu) \sim e^{-c_2 N}$, as desired.

We first prove the lower effectivity inequality (bounding property): $s^-_{N,2N}(\mu) \le s(\mu) \le s^+_{N,2N}(\mu)$, as $N \to \infty$. To demonstrate the lower bound we again appeal to (4.9) and the coercivity of a; indeed, this result (still) obtains for *all* N. To demonstrate the upper bound, we write

$$\begin{aligned} s^+_{N,2N}(\mu) &= s + \left(\frac{1}{\tau} - 1\right)\left(s(\mu) - s_N(\mu)\right) - \frac{1}{\tau}(s(\mu) - s_{2N}(\mu)) && (4.30)\\ &= s + \left(\frac{1}{\tau}[1 - \varepsilon_{N,2N}(\mu)] - 1\right)\left(s(\mu) - s_N(\mu)\right). && (4.31) \end{aligned}$$

We now recall that $s(\mu) - s_N(\mu) \geq 0$, and that $0 < \tau < 1$ — that is, $1/\tau > 1$; it then follows from (4.31) and our hypothesis (4.29) that there exists a finite N^* such that

$$s^+_{N,2N}(\mu) - s(\mu) \geq 0, \quad \forall\ N > N^*. \tag{4.32}$$

This concludes the proof: we obtain *asymptotic* bounds.

We now prove the upper effectivity inequality (sharpness property). From the definitions of $\eta_{N,2N}(\mu)$, $\Delta_{N,2N}(\mu)$ and $\varepsilon_{N,2N}(\mu)$, we directly obtain

$$\eta_{N,2N}(\mu) = \frac{1}{\tau}\frac{s_{2N}(\mu) - s_N(\mu)}{s(\mu) - s_N(\mu)} = \frac{1}{\tau}\frac{(s_{2N}(\mu) - s(\mu)) - (s_N(\mu) - s(\mu))}{(s(\mu) - s_N(\mu))} \tag{4.33}$$

$$= \frac{1}{\tau}(1 - \varepsilon_{N,2N}(\mu)). \tag{4.34}$$

It is readily shown that $\eta_{N,2N}(\mu)$ is bounded from above by $1/\tau$ for all N: we know from (4.9) that $\varepsilon_{N,2N}(\mu)$ is strictly non-negative. It can also readily be shown that $\eta_{N,2N}(\mu)$ is non-negative: since $W_N \subset W_{2N}$, it follows from (4.8) (for $(\cdot,\cdot)_X = a(\cdot,\cdot;\mu)$) and (4.9) that $s(\mu) \geq s_{2N}(\mu) \geq s_N(\mu)$, and hence $\varepsilon_{N,2N}(\mu) \leq 1$. We thus conclude that $0 \leq \eta_{N,2N}(\mu) \leq 1/\tau$ for all N. Furthermore, from our hypothesis on $\varepsilon_{N,2N}(\mu)$, (4.29), we know that $\eta_{N,2N}(\mu)$ will *tend* to $1/\tau$ as N increases.

The essential approximation enabler is exponential convergence: we obtain bounds even for rather small N and relatively large τ. We thus achieve both "near" certainty *and* good effectivities. We demonstrate this claim in Table 4.2, in which we present the bound gap and effectivity for our truss example of Section 4.2.2.0; the results tabulated correspond to the choice $\tau = 1/2$. We clearly obtain bounds for all N; and we observe that $\eta_{N,2N}(\mu)$ does, indeed, rather quickly approach $1/\tau$.

Computational Procedure

Since the error bounds are based entirely on evaluation of the output, we can directly adapt the off–line/on–line procedure of Section 4.3.3. Note that the calculation of the output approximation $s_N(\mu)$ and the output bounds are now integrated: $\underline{A}_N(\mu)$ and $\underline{F}_N(\mu)$ (yielding $s_N(\mu)$) are a sub-matrix and sub-vector of $\underline{A}_{2N}(\mu)$ and $\underline{F}_{2N}(\mu)$ (yielding $s_{2N}(\mu)$, $\Delta_{N,2N}(\mu)$, and $s^{\pm}_{N,2N}(\mu)$), respectively. In the *off–line* stage, we compute the $u(\mu_n)$ and form the $\underline{A}^q_{2N}$ and $\underline{F}_{2N}$: this requires $2N$ (expensive) "a" finite element solutions, and $O(4QN^2)$ finite-element-vector inner products. In the *on–line* stage, for any given new μ, we first form $\underline{A}_N(\mu)$, $\underline{F}_N$ and $\underline{A}_{2N}(\mu)$, $\underline{F}_{2N}$, then solve for $\underline{u}_N(\mu)$ and $\underline{u}_{2N}(\mu)$, and finally evaluate $s^{\pm}_{N,2N}(\mu)$: this requires $O(4QN^2) + O(\frac{16}{3}N^3)$ operations and $O(4QN^2)$ storage. The on–line effort for this Method II predictor/error estimator procedure (based on $s_N(\mu)$ and $s_{2N}(\mu)$) will thus require eightfold more operations than the "predictor-only" procedure of Section 4.3.

Method II is in some sense very naïve: we simply replace the true output $s(\mu)$ with a finer–approximation surrogate $s_{2N}(\mu)$. (There are more obscure ways to describe the method — in terms of a reduced–basis approximation for the error — however there is little to be gained from these alternative interpretations.) The essential computation enabler is again exponential convergence, which permits us to choose $M = 2N$ — hence controlling the additional computational effort attributable to error estimation — while simultaneously ensuring that

$\varepsilon_{N,2N}(\mu)$ tends rapidly to zero. Exponential convergence also ensures that the cost to compute both $s_N(\mu)$ and $s_{2N}(\mu)$ is "negligible." In actual practice, since $s_{2N}(\mu)$ is available, we can of course take $s_{2N}(\mu)$, rather than $s_N(\mu)$, as our output prediction; this greatly improves not only accuracy, but also certainty — $\Delta_{N,2N}(\mu)$ is almost surely a bound for $s(\mu) - s_{2N}(\mu)$, albeit an exponentially conservative bound as N tends to infinity.

4.5 Extensions

4.5.1 Noncompliant Outputs and Nonsymmetric Operators

In Sections 4.3 and 4.4 we formulate the reduced-basis method and associated error estimation procedure for the case of compliant outputs, $\ell(v) = f(v),\ \forall v \in X$. We briefly summarize here the formulation and theory for more general linear bounded output functionals; moreover, the assumption of symmetry (but not yet coercivity) is relaxed, permitting treatment of a wider class of problems — a representative example is the convection-diffusion equation, in which the presence of the convective term renders the operator nonsymmetric. We first present the reduced-basis approximation, now involving a dual or adjoint problem; we then formulate the associated *a posteriori* error estimators; and we conclude with a few illustrative results.

As a preliminary, we first generalize the abstract formulation of Section 4.2.1. As before, we define the "primal" problem as in (4.4), however we of course no longer require symmetry. But we also introduce an associated adjoint or "dual" problem: for any $\mu \in X$, find $\psi(\mu) \in X$ such that

$$a(v, \psi(\mu); \mu) = -\ell(v), \quad \forall\ v \in X; \tag{4.35}$$

recall that $\ell(v)$ is our output functional.

Reduced-Basis Approximation

To develop the reduced-basis space, we first choose — randomly or log-randomly as described in Section 4.3.2 — a sample set in parameter space, $S_{N/2} = \{\mu_1, \ldots, \mu_{N/2}\}$, where $\mu_i \in \mathcal{D},\ i = 1, \ldots, N/2$ (N even); we next define an "integrated" Lagrangian reduced-basis approximation space, $W_N = \text{span}\ \{(u(\mu_n), \psi(\mu_n)),\ n = 1, \ldots, N/2\}$.

For any $\mu \in \mathcal{D}$, our reduced basis approximation is then obtained by standard Galerkin projection onto W_N (though for highly nonsymmetric operators minimum residual and Petrov-Galerkin projections are attractive — stabler — alternatives). To wit, for the primal problem, we find $u_N(\mu) \in W_N$ such that $a(u_N(\mu), v; \mu) = f(v),\ \forall\ v \in W_N$; and for the adjoint problem, we define (though, unless otherwise indicated, do *not* compute) $\psi_N(\mu) \in W_N$ such that $a(v, \psi_N(\mu); \mu) = -\ell(v),\ \forall v \in W_N$. The reduced-basis output approximation is then calculated from $s_N(\mu) = \ell(u_N(\mu))$.

Turning now to the *a priori* theory, it follows from standard arguments that $u_N(\mu)$ and $\psi_N(\mu)$ are "optimal" in the sense that

$$\begin{aligned} \|u(\mu) - u_N(\mu)\|_X &\leq \left(1 + \frac{\gamma(\mu)}{\alpha(\mu)}\right) \inf_{w_N \in W_N} \|u(\mu) - w_N\|_X, \\ \|\psi(\mu) - \psi_N(\mu)\|_X &\leq \left(1 + \frac{\gamma(\mu)}{\alpha(\mu)}\right) \inf_{w_N \in W_N} \|\psi(\mu) - w_N\|_X. \end{aligned}$$

The best approximation analysis is then similar to that presented in Section 4.3.2. As regards our output, we now have

$$\begin{aligned} |s(\mu) - s_N(\mu)| &= |\ell(u(\mu)) - \ell(u_N(\mu))| \\ &= |a(u - u_N, \psi; \mu)| \\ &= |a(u - u_N, \psi - \psi_N; \mu)| \\ &\leq \gamma_0 \|u - u_N\|_X \|\psi - \psi_N\|_X \end{aligned} \tag{4.36}$$

from Galerkin orthogonality, the definition of the primal and the adjoint problems, and the Cauchy-Schwartz inequality. We now understand why we include the $\psi(\mu_n)$ in W_N: to ensure that $\|\psi(\mu) - \psi_N(\mu)\|_X$ is small. We thus recover the "square" effect in the convergence rate of the output, albeit (and unlike the symmetric case) at the expense of some additional computational effort — the inclusion of the $\psi(\mu_n)$ in W_N; typically, even for the very rapidly convergent reduced-basis approximation, the "fixed error-minimum cost" criterion favors the adjoint enrichment.

For simplicity of exposition (and to a certain extent, implementation), we present here the "integrated" primal-dual approximation space. However, there are significant computational and conditioning advantages associated with a "non-integrated" approach, in which we introduce *separate* primal $(u(\mu_n))$ and dual $(\psi(\mu_n))$ approximation spaces for $u(\mu)$ and $\psi(\mu)$, respectively. Note in the "non-integrated" case we are obliged to compute $\psi_N(\mu)$, since to preserve the output error "square effect" we must modify our predictor with a residual correction, $f(\psi_N(\mu)) - a(u_N(\mu), \psi_N(\mu); \mu)$ [44]. Both the "integrated" and "non-integrated" approaches admit an off–line/on–line decomposition similar to that described in Section 4.3.3 for the compliant, symmetric problem; as before, the on–line complexity and storage are independent of the dimension of the very fine ("truth") finite element approximation.

Method I *A Posteriori* Error Estimators

We extend here the method developed in Section 4.4.1.0 to the more general case of noncompliant and nonsymmetric problems. We begin with the formulation.

We first find $\hat{e}^{\rm pr}(\mu) \in X$ such that

$$g(\mu)\, \hat{a}(\hat{e}^{\rm pr}(\mu), v) = R^{\rm pr}(v; u_N(\mu); \mu),\ \forall\ v \in X,$$

where $R^{\rm pr}(v; w; \mu) \equiv f(v) - a(w, v; \mu),\ \forall\ v \in X$; and $\hat{e}^{\rm du}(\mu) \in X$ such that

$$g(\mu)\hat{a}(\hat{e}^{\rm du}(\mu), v) = R^{\rm du}(v; \psi_N(\mu); \mu),\ \forall\ v \in X,$$

where $R^{\rm du}(v; w; \mu) \equiv -\ell(v) - a(v, w; \mu), \forall\ v \in X$. We then define

$$\bar{s}_N(\mu) = s_N(\mu) - \frac{g(\mu)}{2}\hat{a}(\hat{e}^{pr}(\mu), \hat{e}^{du}(\mu)),\ \text{and} \tag{4.37}$$

$$\Delta_N(\mu) = \frac{g(\mu)}{2}\left[\hat{a}(\hat{e}^{\rm pr}(\mu), \hat{e}^{\rm pr}(\mu))\right]^{\frac{1}{2}} \left[\hat{a}(\hat{e}^{\rm du}(\mu), \hat{e}^{\rm du}(\mu))\right]^{\frac{1}{2}}. \tag{4.38}$$

Finally, we evaluate our lower and upper estimators as $s_N^{\pm}(\mu) = \bar{s}_N(\mu) \pm \Delta_N(\mu)$. Note that, as before, $g(\mu)$ and $\hat{a}$ still satisfy (4.16); and that, furthermore, (4.16) will only involve the

symmetric part of a. We define the effectivity as

$$\eta_N(\mu) \;=\; \frac{\Delta_N(\mu)}{|s(\mu) - s_N(\mu)|}; \tag{4.39}$$

note that $s(\mu) - s_N(\mu)$ now has no definite sign.

We now prove that our error estimators are bounds (the lower effectivity inequality): $s_N^-(\mu) \le s(\mu) \le s_N^+(\mu),\ \forall\ N$. To begin, we define $\hat{e}^{\pm}(\mu) = \hat{e}^{\rm pr}(\mu) \mp \frac{1}{\kappa}\hat{e}^{\rm du}(\mu)$, and note that, from the coercivity of $\hat{a}$,

$$\kappa g(\mu)\hat{a}(e^{\rm pr} - \frac{1}{2}\hat{e}^{\pm}, e^{\rm pr} - \frac{1}{2}\hat{e}^{\pm}) = \kappa g(\mu)\hat{a}(e^{\rm pr}, e^{\rm pr}) + \frac{\kappa g(\mu)}{4}\hat{a}(\hat{e}^{\pm}, \hat{e}^{\pm}) - \kappa g(\mu)\hat{a}(\hat{e}^{\pm}, e^{\rm pr}) \ge 0, \tag{4.40}$$

where $e^{\rm pr}(\mu) = u(\mu) - u_N(\mu)$, $e^{\rm du}(\mu) = \psi(\mu) - \psi_N(\mu)$, and κ is a positive real number. From the definition of $\hat{e}^{\pm}(\mu)$ and $\hat{e}^{\rm pr}(\mu)$, $\hat{e}^{\rm du}(\mu)$, we can express the "cross-term" as

$$\begin{aligned} g(\mu)\hat{a}(\hat{e}^{\pm}, e^{\rm pr}) \;&=\; R^{\rm pr}(e^{\rm pr}; u_N; \mu) \mp \frac{1}{\kappa}R^{\rm du}(e^{\rm pr}; \psi_N; \mu) = a(e^{\rm pr}, e^{\rm pr}; \mu) \mp \frac{1}{\kappa}a(e^{\rm pr}, e^{\rm du}; \mu) \\ &=\; a(e^{\rm pr}, e^{\rm pr}; \mu) \pm \frac{1}{\kappa}(s(\mu) - s_N(\mu)), \end{aligned} \tag{4.41}$$

since $R^{\rm pr}(e^{\rm pr}; u^N; \mu) = a(u, e^{\rm pr}; \mu) - a(u_N, e^{\rm pr}; \mu) = a(e^{\rm pr}, e^{\rm pr}; \mu)$, $R^{\rm du}(e^{\rm pr}; \psi^N; \mu) = a(e^{\rm pr}, \psi; \mu) - a(e^{\rm pr}, \psi_N; \mu) = a(e^{\rm pr}, e^{\rm du}; \mu)$, and $\ell(\mu) - \ell(u_N) = -a(u - u_N, \psi; \mu) = -a(u - u_N, \psi - \psi_N; \mu)$ (by Galerkin orthogonality) $= -a(e^{\rm pr}, e^{\rm du}; \mu)$. We then substitute (4.41) into (4.40) to obtain

$$\pm(s(\mu) - s_N(\mu)) \le -\kappa\left(a(e^{\rm pr}, e^{\rm pr}; \mu) - g(\mu)\hat{a}(e^{\rm pr}, e^{\rm pr})\right) + \frac{\kappa g(\mu)}{4}\hat{a}(\hat{e}^{\pm}, \hat{e}^{\pm}) \le \frac{\kappa g(\mu)}{4}\hat{a}(\hat{e}^{\pm}, \hat{e}^{\pm}),$$

since $\kappa > 0$ and $a(e^{\rm pr}(\mu), e^{\rm pr}(\mu); \mu) - g(\mu)\hat{a}(e^{\rm pr}(\mu), e^{\rm pr}(\mu)) \ge 0$ from (4.16).

Expanding $\hat{e}^{\pm}(\mu) = \hat{e}^{\rm pr}(\mu) \mp \frac{1}{\kappa}\hat{e}^{\rm du}(\mu)$ then gives

$$\pm(s(\mu) - s_N(\mu)) \le \frac{g(\mu)}{4}\left[\kappa\hat{a}(\hat{e}^{\rm pr}, \hat{e}^{\rm pr}) + \frac{1}{\kappa}\hat{a}(\hat{e}^{\rm du}, \hat{e}^{\rm du}) \mp 2\hat{a}(\hat{e}^{\rm pr}, \hat{e}^{\rm du})\right],$$

or

$$\pm\left(s(\mu) - (s_N(\mu) - \frac{g(\mu)}{2}\hat{a}(\hat{e}^{\rm pr}, \hat{e}^{\rm du}))\right) \le \frac{\kappa g(\mu)}{4}\hat{a}(\hat{e}^{\rm pr}, \hat{e}^{\rm pr}) + \frac{g(\mu)}{4\kappa}\hat{a}(\hat{e}^{\rm du}, \hat{e}^{\rm du}). \tag{4.42}$$

We now choose $\kappa(\mu)$ as

$$\kappa(\mu) = \left(\frac{\hat{a}(\hat{e}^{\rm du}(\mu), \hat{e}^{\rm du}(\mu))}{\hat{a}(\hat{e}^{\rm pr}(\mu), \hat{e}^{\rm pr}(\mu))}\right)^{\frac{1}{2}}$$

so as to minimize the right-hand side (4.42); we then obtain

$$|s(\mu) - \bar{s}_N(\mu)| \le \Delta_N(\mu), \tag{4.43}$$

and hence $s_N^-(\mu) \le s(\mu) \le s_N^+(\mu)$.

We now turn to the upper effectivity inequality (sharpness property). If the primal and dual errors are a-orthogonal, or become increasingly orthogonal as N increases, then the effectivity

will not, in fact, be bounded as $N \to \infty$. However, if we make the (plausible) hypothesis that $|s(\mu) - s_N(\mu)| \geq \underline{C}\|e^{\rm pr}(\mu)\|_X \|e^{\rm du}(\mu)\|_X$, then it is simple to demonstrate that

$$\eta_N(\mu) \leq \frac{\gamma_0^2}{2\underline{C}\ \underline{\alpha}_0}. \tag{4.44}$$

In particular, it is an easy matter to demonstrate that $g^{1/2}(\mu)\left(\hat{a}\left(\hat{e}^{\rm pr}(\mu), \hat{e}^{\rm pr}(\mu)\right)\right)^{1/2} \leq \frac{\gamma_0}{\underline{\alpha}_0^{1/2}}\|e^{\rm pr}(\mu)\|_X$ (note we lose a factor of $\gamma_0^{1/2}$ relative to the symmetric case); similarly, $g^{1/2}(\mu)\left(\hat{a}\left(\hat{e}^{\rm du}(\mu), \hat{e}^{\rm du}(\mu)\right)\right)^{1/2} \leq \frac{\gamma_0}{\underline{\alpha}_0^{1/2}}\|e^{\rm du}(\mu)\|_X$. The desired result then directly follows from the definition of $\Delta_N(\mu)$ and our hypothesis on $|s(\mu) - s_N(\mu)|$.

Finally, turning to computational issues, we note that the off–line/on–line decomposition described in Section 4.4.1 for compliant symmetric problems directly extends to the noncompliant, nonsymmetric case — except that we must compute the norm of both the primal and dual "modified errors," with a concomitant doubling of computational effort.

Method II *A Posteriori* Error Estimators

We discuss here the extension of Method II of Section 4.4.2 to noncompliant outputs and nonsymmetric operators.

To begin, we set $M > N$, M even, and introduce a parameter sample $S_{M/2} = \{\mu_1, \ldots, \mu_{M/2}\}$ and associated "integrated" reduced-basis approximation space $W_M = \text{span}\{u(\mu_m), \psi(\mu_m), m = 1, \ldots, M/2\}$. We first find $u_M(\mu) \in W_M$ such that $a(u_M(\mu), v; \mu) = f(v),\ \forall v \in W_M$; we then evaluate $s_M(\mu) = \ell(u_M(\mu))$; and finally, we compute our upper and lower output estimators as

$$s_{N,M}^{\pm}(\mu) = s_N(\mu) + \frac{1}{2\tau}(s_M - s_N) \pm \frac{1}{2}\Delta_{N,M}(\mu), \tag{4.45}$$

$$\Delta_{N,M}(\mu) = \frac{1}{\tau}|s_M(\mu) - s_N(\mu)|, \tag{4.46}$$

for $\tau \in (0, 1)$. The effectivity of the approximation is defined as

$$\eta_{N,M}(\mu) = \frac{\Delta_{N,M}(\mu)}{|s(\mu) - s_N(\mu)|}. \tag{4.47}$$

We shall again only consider $M = 2N$.

As in Section 4.4.2, we would like to prove that $1 \leq \eta_{N,2N}(\mu) \leq \text{Const}$ for sufficiently large N; and, as in Section 4.4.2, we must again make the hypothesis (4.29). We first consider the lower effectivity inequality (bounding property), and prove that

$$s_{N,2N}^{-}(\mu) \leq s(\mu) \leq s_{N,2N}^{+}(\mu), \text{ as } N \to \infty. \tag{4.48}$$

N	$\lvert s(\mu) - s_N(\mu)\rvert / s(\mu)$	$\Delta_{N,2N}(\mu)/s(\mu)$	$\eta_{N,2N}(\mu)$
20	2.35×10^{-2}	4.67×10^{-2}	1.99
40	1.74×10^{-4}	3.19×10^{-4}	1.83
60	5.59×10^{-5}	1.06×10^{-4}	1.90
80	1.44×10^{-5}	2.73×10^{-5}	1.89
100	7.45×10^{-6}	1.40×10^{-5}	1.88

Table 4.3: Error, error bound (Method II), and effectivity as a function of N, at a particular representative point $\mu \in \mathcal{D}$, for the truss problem (noncompliant output).

In particular, simple algebraic manipulations yield

$$s^-_{N,2N}(\mu) = s(\mu) - \frac{1}{1-\varepsilon_{N,2N}} |s_N(\mu) - s_{2N}(\mu)| \times \tag{4.49}$$

$$\begin{cases} 1 & s_{2N}(\mu) \geq s_N(\mu) \\ \frac{1}{\tau}(1-\varepsilon_{N,2N}) - 1 & s_{2N}(\mu) < s_N(\mu) \end{cases}, \tag{4.50}$$

$$s^+_{N,2N}(\mu) = s(\mu) + \frac{1}{1-\varepsilon_{N,2N}} |s_N(\mu) - s_{2N}(\mu)| \times \tag{4.51}$$

$$\begin{cases} \frac{1}{\tau}(1-\varepsilon_{N,2N}) - 1 & s_{2N}(\mu) \geq s_N(\mu) \\ 1 & s_{2N}(\mu) < s_N(\mu). \end{cases} \tag{4.52}$$

The desired result then directly follows from our hypothesis on $\varepsilon_{N,2N}$, (4.29), and the range of τ.

The proof for the upper effectivity inequality (sharpness property) parallels the derivation of Section 4.4.2.0. In particular, we write

$$\eta_{N,2N}(\mu) = \frac{\frac{1}{\tau}|s_{2N} - s_N|}{|s - s_N|} = \frac{\frac{1}{\tau}|s_{2N} - s + s - s_N|}{|s - s_N|} \tag{4.53}$$

$$= \frac{1}{\tau}|1 - \varepsilon_{N,2N}|; \tag{4.54}$$

from our hypothesis (4.29) we may thus conclude that $\eta_{N,2N}(\mu) \rightarrow \frac{1}{\tau}$ as $N \rightarrow \infty$. Note in the noncompliant, nonsymmetric case we can make no stronger statement.

We demonstrate our effectivity claims in Table 4.3, in which we present the error, bound gap, and effectivity for the noncompliant output ($s^2(\mu)$, average stress) of the truss example of Section 4.2.2.0; the results tabulated correspond to the choice $\tau = 1/2$. We clearly obtain bounds for all N; and the effectivity rather quickly approaches $1/\tau$ (for $N \geq 120$, $\eta_{N,2N}$ remains fixed at $1/\tau = 2.0$).

4.5.2 Eigenvalue Problems

We next consider the extension of our approach to symmetric positive definite eigenvalue problems. The eigenvalues of appropriately defined partial-differential-equation eigenproblems

convey critical information about a physical system; in linear elasticity, the critical buckling load; in dynamic analysis of structures, the resonant modes; in conduction heat transfer, the equilibrium timescales. Solution of large-scale eigenvalue problems is computationally intensive: the reduced-basis method is thus very attractive.

The abstract statement of our eigenvalue problem is: find $(u_i(\mu), \lambda_i(\mu)) \in X \times \mathbb{R},\ i = 1, \ldots,$ such that

$$a(u_i(\mu), v; \mu) = \lambda_i(\mu) m(u_i(\mu), v; \mu),\ \forall v \in X,\ \text{and}\ m(u_i(\mu), u_i(\mu)) = 1. \tag{4.55}$$

Here a is the continuous, coercive, symmetric form introduced earlier, and m is (say) the L^2 inner product over Ω. The assumptions on a and m imply the eigenvalues $\lambda_i(\mu)$ will be real and positive. We order the eigenvalues (and corresponding eigenfunctions u_i) such that $0 < \lambda_1(\mu) < \lambda_2(\mu) \leq \ldots$; we shall assume that $\lambda_1(\mu)$ and $\lambda_2(\mu)$ are distinct. We suppose that our output of interest is the minimum eigenvalue,

$$s(\mu) = \lambda_1(\mu); \tag{4.56}$$

other outputs may also be considered.

Following [37], we present here a reduced-basis predictor and a Method I error estimator for symmetric positive-definite eigenvalue problems; we also briefly describe the simpler Method II approach.

Reduced-Basis Approximation

We sample — randomly or log-randomly — our design space $\mathcal{D}$ to create the parameter sample $S_{N/2} = \{\mu_1, \ldots, \mu_{N/2}\}$; we then introduce the reduced-basis space $W_N = \text{span}\{u_1(\mu_1), u_2(\mu_1), \ldots, u_1(\mu_{N/2}), u_2(\mu_{N/2})\}$, where we recall that $u_1(\mu)$ and $u_2(\mu)$ are the eigenfunctions associated with the first (smallest) and second eigenvalues $\lambda_1(\mu)$ and $\lambda_2(\mu)$, respectively. Note that W_N has good approximation properties both for the first and second lowest eigenfunctions, and hence eigenvalues; this is required by the Method I error estimator to be presented below. Our reduced-order approximation is then: find $(u_{N\,i}(\mu), \lambda_{Ni}(\mu)) \in W_N \times \mathbb{R},\ i = 1, \ldots, N,$ such that

$$a(u_{N\,i}(\mu), v; \mu) = \lambda_{N\,i}(\mu) m(u_{N\,i}(\mu), v; \mu),\ \forall v \in W_N,\ \text{and}\ m(u_{N\,i}(\mu), u_{N\,i}(\mu)) = 1; \tag{4.57}$$

the output approximation is then $s_N(\mu) = \lambda_{N\,1}(\mu)$.

The formulation admits an on–line/off–line decomposition [37] very similar to the approach described for equilibrium problems in Section 4.3.

Method I *A Posteriori* Error Estimators

As before, we assume that we are given a positive function $g(\mu) : \mathcal{D} \to \mathbb{R}_+$ and a continuous, coercive, symmetric bilinear form $\hat{a}(w, v) : X \times X \to \mathbb{R}$, that satisfy the inequality (4.16). We then find $\hat{e}(\mu) \in X$ such that

$$g(\mu)\hat{a}(\hat{e}(\mu), v) = \left[\lambda_{N\,1} m(u_{N\,1}(\mu), v; \mu) - a(u_{N\,1}(\mu), v; \mu)\right],\ \forall v \in X, \tag{4.58}$$

N	$\lvert\lambda_1(\mu) - \lambda_{N\,1}(\mu)\rvert/\lambda_1(\mu)$	$\Delta_N(\mu)/\lambda_1(\mu)$	$\eta_N(\mu)$
10	1.19×10^{-2}	6.66×10^{-2}	5.63
20	1.08×10^{-3}	7.19×10^{-3}	6.65
30	6.20×10^{-4}	3.19×10^{-3}	5.17
40	1.72×10^{-4}	1.55×10^{-3}	9.44
50	3.47×10^{-5}	4.06×10^{-4}	11.74

Table 4.4: Error, error bound (Method I), and effectivities as a function of N, at a particular representative point $\mu \in \mathcal{D}$, for the thermal fin eigenproblen.

in which the right-hand side is the eigenproblem equivalent of the residual. We then evaluate our estimators as

$$s_N^+(\mu) = \lambda_{N\,1}(\mu), \qquad s_N^-(\mu) = \lambda_{N\,1}(\mu) - \Delta_N(\mu),$$

$$\Delta_N(\mu) = \frac{g(\mu)}{\tau\delta(\mu)}\hat{a}(\hat{e}(\mu), \hat{e}(\mu)),$$

where $\delta(\mu) = 1 - \frac{\lambda_{N\,1}(\mu)}{\lambda_{N\,2}(\mu)}$ and $\tau \in (0, 1)$. The effectivity is defined as $\eta_N(\mu) = \frac{\Delta_N(\mu)}{\lambda_{N\,1}(\mu) - \lambda_1(\mu))}$.

We now consider the lower and upper effectivity inequalities. As regards the lower effectivity inequality (bounding property), we of course obtain $s_N^+(\mu) \geq \lambda_1(\mu), \ \forall\, N$. The difficult result is the lower bound: it can be proven [37] that there exists an $N^*(S_{N/2}, \mu)$ such that $s_N^-(\mu) \leq \lambda_1(\mu), \ \forall N > N^*$. In practice, $N^* = 1$, due to the good (theoretically motivated) choice for $\delta(\mu)$; there is thus very little uncertainty in our (asymptotic) bounds. We also prove in [37] a result related to the upper effectivity inequality (sharpness property); in, practice, very good effectivities are obtained. To demonstrate these claims we consider the eigenvalue problem associated with (the homogenous version) of our two-dimensional thermal fin example of Section 4.2.2.0. We present in Table 4.5.2.0 the error, error bound, and effectivity as a function of N at a particular point $\mu \in \mathcal{D}$. We observe rapid convergence, bounds for all N, and good effectivities.

Finally, we note that our output estimator admits an off–line/on–line decomposition similar to that for equilibrium problems; the additional terms in (4.58) are readily treated through our affine expansion/linear superposition procedure.

Method II *A Posteriori* Error Estimators

For Method II, we no longer require an estimate for the second eigenvalue. We may thus define $S_N = \{\mu_1, \ldots, \mu_N\}$, $W_N = \text{span}\{u_1(\mu_i), i = 1, \ldots, N\}$, and (for $M = 2N$) $S_{2N} = \{\mu_1, \ldots, \mu_{2N}\} \supset S_N$, $W_{2N} = \text{span}\{u_1(\mu_i), i = 1, \ldots, 2N\} \supset W_N$. The reduced basis approximation now takes the form (4.55), yielding $s_N(\mu) = \lambda_{N\,1}(\mu)$ and (for $N \to 2N$) $s_{2N}(\mu) = \lambda_{2N\,1}(\mu)$. Our estimators are then given by

$$\begin{aligned} s_{N,2N}^+(\mu) &= \lambda_{N\,1}(\mu), \quad s_{N,2N}^- = \lambda_{N\,1}(\mu) - \Delta_{N,2N}(\mu), \\ \Delta_{N,2N}(\mu) &= \frac{1}{\tau}(s_N(\mu) - s_{2N}(\mu)) \end{aligned} \tag{4.59}$$

for $\tau \in (0,1)$. The effectivity $\eta_{N,2N}(\mu)$ is defined as for Method I.

For the lower effectivity inequality (bounding property), we of course retain $s^+_{N,2N}(\mu) > \lambda_1(\mu)$, $\forall N$. We also readily derive $s^-_{N,2N}(\mu) = \lambda_1 - (\lambda_{N\,1} - \lambda_1)(\frac{1}{\tau}(1-\varepsilon_{N,2N})-1)$; under our hypothesis (4.29), we thus obtain asymptotic bounds as $N \to \infty$. For the upper effectivity inequality (sharpness property), we directly obtain $\eta_{N,2N} = \frac{1}{\tau}(1-\varepsilon_{N,2N})$. By variational arguments it is readily shown that $0 \le \varepsilon_{N,2N} \le 1$: we thus conclude that $0 \le \eta_{N,2N} \le \frac{1}{\tau}$, $\forall N$. Additionally, under hypothesis (4.29), we deduce that $\eta_{N,2N} \to \frac{1}{\tau}$ as $N \to \infty$.

4.5.3 Further Generalizations

In this section we briefly describe several additional extensions of the methodology. In each case we focus on the essential new ingredient; further details (in most cases) may be found in the referenced literature.

Noncoercive Linear Operators

The archetypical noncoercive linear equation is the Helmholtz, or reduced–wave, equation; many (e.g., inverse scattering) applications of this equation arise, for example, in acoustics and electromagnetics. The essential new mathematical ingredient is the loss of coercivity of a. In particular, well–posedness is now ensured only by the inf–sup condition: there exists positive β_0, $\beta(\mu)$, such that

$$0 \;<\; \beta_0 \;\le\; \beta(\mu) \;=\; \inf_{w\in X}\sup_{v\in X}\frac{a(w,v;\mu)}{\|w\|_X\,\|v\|_X}, \quad \forall\ \mu\in\mathcal{D}. \tag{4.60}$$

Two numerical difficulties arise due to this "weaker" stability condition.

The first difficulty is preservation of the inf–sup stability condition for finite dimensional approximation spaces. To wit, although in the coercive case restriction to the space W_N actually increases stability, in the noncoercive case restriction to the space W_N can easily decrease stability: the relevant supremizers may not be adequately represented. Loss of stability can, in turn, lead to poor approximations — the inf–sup parameter enters in the denominator of the *a priori* convergence result. The second numerical difficulty is estimation of the inf–sup parameter, which for noncoercive problems plays the role of $g(\mu)$ in Method I *a posteriori* error estimation techniques. In particular, $\beta(\mu)$ can not typically be deduced analytically, and thus must be evaluated (via an eigenvalue formulation) as part of the reduced–basis approximation. Our resolution of both these difficulties involves two elements [44]: first, we consider projections other than standard Galerkin; and second, we consider "enriched" approximation spaces.

In one approach [44], we pursue a minimum–residual projection: the (low-dimensional) infimizing space contains both the solution $u(\mu)$ and also the inf–sup infimizer at the μ_n sample points; and the (high-dimensional) supremizing space is taken to be X. Stability is ensured and rigorous (sharp) error bounds are obtained — though technically the bounds are only asymptotic due to the approximation of the inf–sup parameter; and, despite the presence of X, the on–line complexity remains independent of the dimension of X — as in Section 4.3.3, we exploit affine parameter dependence and linear superposition to precompute the necessary inversions. In a second suite of much simpler and more general approaches (see [44] for one

example in the symmetric case), we exploit minimum-residual or Petrov-Galerkin projections with infimizer–supremizer enriched, but still very low–dimensional, infimizing and supremizing spaces. Plausible but not yet completely rigorous arguments, and empirical evidence, suggest that stability is ensured and rigorous asymptotic (and sharp) error bounds are obtained.

In [44] we focus entirely on Method I *a posteriori* error estimator procedures; but Method II techniques are also appropriate. In particular, Method II approaches do not require accurate estimation of the inf-sup parameter; we thus need be concerned only with stability in designing our reduced–basis spaces.

Parabolic Partial Differential Equations

The next extension considered is the treatment of parabolic partial differential equations of the form $m(u_t, v; \mu) = a(u, v; \mu)$; typical examples are time-dependent problems such as unsteady heat conduction — the "heat" or "diffusion" equation. The essential new ingredient is the presence of the time variable, t.

The reduced-basis approximation and error estimator procedures are similar to those for noncompliant nonsymmetric problems, except that we now include the time variable as an additional parameter. Thus, as in certain other time-domain model-order-reduction methods [6, 73], the basis functions are "snapshots" of the solution at selected time instants; however, in our case, we construct an *ensemble* of such series corresponding to different points in the non-time parameter domain $\mathcal{D}$. For rapid convergence of the output approximation, the solutions to an adjoint problem — which evolves *backward* in time — must also be included in the reduced-basis space.

For the temporal discretization method, many possible choices are available. The most appropriate method — although not the only choice — is the discontinuous Galerkin method [39]. The variational origin of the discontinuous Galerkin approach leads naturally to rigorous output bounds for Method I *a posteriori error* estimators; the Method II approach is also directly applicable. Under our affine assumption, off–line/on–line decompositions can be readily crafted; the complexity of the on–line stage (calculation of the output predictor and associated bound gap) is, as before, independent of the dimension of X.

Locally Non–Affine Parameter Dependence

An important restriction of our methods is the assumption of affine parameter dependence. Although many property, boundary condition, load, and even geometry variations can indeed be expressed in the required form (4.2) for reasonably small Q, there are many problems — for example, general boundary shape variations — which do not admit such a representation. One simple approach to the treatment of this more difficult class of non-affine problems is (*i*) in the off–line stage, store the $\zeta_n \equiv u(\mu_n)$, and (*ii*) in the on–line stage, directly evaluate the reduced–basis stiffness matrix as $a(\zeta_j, \zeta_i, \mu)$. Unfortunately, the operation count (respectively, storage) for the on–line stage will now scale as $O(N^2 \dim(X))$ (respectively, $O(N \dim(X))$, where $\dim(X)$ is the dimension of the truth (very fine) finite element approximation space: the resulting method may no longer be competitive with advanced iterative techniques; and, in any event, "real–time" response may be compromised.

We prefer an approach which is slightly less general but potentially much more efficient. In particular, we note that in many cases — for example, boundary geometry modification — the non–affine parametric dependence can be restricted to a small subdomain of Ω, Ω_{II}. We can then express our bilinear form a as an affine/non–affine sum,

$$a(w, v; \mu) = a_I(w, v; \mu) + a_{II}(w, v; \mu). \tag{4.61}$$

Here a_I, defined over Ω_I — the majority of the domain — is affinely dependent on μ; and a_{II}, defined over Ω_{II} — a small portion of the domain — is not affinely dependent on μ. It immediately follows that the reduced–basis stiffness matrix can be expressed as the sum of two stiffness matrices corresponding to contributions from a_I and a_{II} respectively; that the stiffness matrix associated with a_I admits the usual on–line/off–line decomposition described in Section 4.3.3; and that the stiffness matrix associated with a_{II} requires storage (and inner product evaluation) *only of* $\zeta_i|_{\Omega_{II}}$ (ζ_i restricted to Ω_{II}). The non–affine contribution to the on–line computational complexity thus scales only as $O(N^2 \dim(X|_{\Omega_{II}}))$, where $\dim(X|_{\Omega_{II}})$ refers (in practice) to the number of finite-element nodes located within Ω_{II} — often extremely small. We thus recover a method that is (almost) independent of $\dim(X)$, though clearly the on–line code will be more complicated than in the purely affine case.

In the above we focus on approximation. As regards *a posteriori* error estimation, the non–affine dependence of a (even locally) precludes the precomputation and linear superposition strategy required by Method I (unless domain decomposition concepts are exploited [38]); however, Method II directly extends to the locally non-affine case.

Acknowledgements We would like to thank Mr. Thomas Leurent (formerly) of MIT for his many contributions to the work described in this paper; thanks also to Shidrati Ali of the Singapore-MIT Alliance and Yuri Solodukhov of MIT for very helpful discussions. We would also like to acknowledge our longstanding collaborations with Professor Jaime Peraire of MIT and Professor Einar Rønquist of the Norwegian University of Science and Technology. This work was supported by the Singapore-MIT Alliance, by DARPA and ONR under Grant F49620-01-1-0458, by DARPA and AFOSR under Grant N00014-01-1-0523 (Subcontract 340-6218-3), and by NASA under Grant NAG-1-1978.

Chapter 5

A Posteriori Error Bounds for Reduced-Basis Approximation of Parametrized Noncoercive and Nonlinear Elliptic Partial Differential Equations

Authors: K. Veroy, C. Prud'homme, D.V. Rovas, A.T. Patera.

5.1 1 Introduction

The optimization, control, and characterization of an engineering component or system requires the prediction of certain "quantities of interest," or performance metrics, which we shall denote *outputs* — for example deflections, heat transfer rates, or drags. These outputs are typically expressed as functionals of field variables associated with a parametrized partial differential equation which describes the physical behavior of the component or system. The parameters, which we shall denote *inputs*, serve to identify a particular "configuration" of the component. We thus arrive at an implicit *input-output* relationship, evaluation of which demands solution of the underlying partial differential equation.

Our goal is the development of computational methods that permit *rapid* and *reliable* evaluation of this partial-differential-equation-induced input-output relationship *in the limit of many queries* — that is, in the design, optimization, control, and characterization contexts. Our particular approach is based on the reduced-basis method, first introduced in the late 1970s for nonlinear structural analysis [5, 51], and subsequently developed more broadly in the 1980s and 1990s [13, 14, 26, 55, 58, 70]. The reduced-basis method recognizes that the field variable is not, in fact, some arbitrary member of the infinite-dimensional solution space associated with the partial differential equation; rather, it resides, or "evolves," on a much lower-dimensional manifold induced by the parametric dependence.

The reduced-basis approach as earlier articulated is local in parameter space in both practice and theory [26]. As a result, the computational improvements — relative to conventional

(say) finite element approximation — are often quite modest [58]. Our work [37, 44, 46, 62, 82, 61] differs from these earlier efforts in several important ways: first, we develop *global* approximation spaces; second, we introduce rigorous *a posteriori error estimators*; and third, we exploit *off-line/on-line* computational decompositions (see [13] for an earlier application of this strategy.) These three ingredients allow us — for the restricted but important class of "parameter-affine" problems — to reliably decouple the generation and projection stages of reduced-basis approximation, thereby effecting computational economies of several orders of magnitude.

In this paper we develop new *a posteriori* error estimation procedures for noncoercive linear, and certain nonlinear, problems that — unlike our earlier "asymptotic" techniques [44, 62] — yield rigorous error statements for *all* N. We consider three particular examples: the Helmholtz (reduced-wave) equation (Section 2); a cubically nonlinear Poisson equation (Section 3); and Burgers equation (Section 4) — a model for incompressible Navier-Stokes. The Helmholtz (and Burgers) example introduce our new lower bound constructions for the requisite inf-sup (singular value) stability factor; the cubic nonlinearity exercises symmetry factorization procedures necessary for treatment of high-order Galerkin summations in the (say) residual dual-norm calculation; and the Burgers equation illustrates our accommodation of potentially multiple solution branches in our *a posteriori* error statement. Numerical results are presented that demonstrate the rigor, sharpness, and efficiency of our proposed error bounds, and the application of these bounds to adaptive (optimal) approximation.

5.2 2 Noncoercive Linear Problems: Helmholtz Equation

5.2.1 2.1 Preliminaries

We consider a suitably regular domain $\Omega \subset \mathbb{R}^d$, $1 \leq d \leq 3$, with boundary $\partial\Omega$. We then introduce a Hilbert space Y with associated inner product, $(\cdot\,,\cdot)_Y$, and induced norm, $\|\cdot\|_Y$. We shall assume that $H^1_0(\Omega) \subset Y \subset H^1(\Omega)$, where $H^1(\Omega) \equiv \{v \in L^2(\Omega),\ \nabla v \in (L^2(\Omega))^d\}$, $H^1_0 \equiv \{v \in H^1(\Omega)\,|\,v|_{\partial\Omega} = 0\}$, and $L^2(\Omega)$ is the space of square-integrable functions over Ω. We shall further assume that

$$\begin{aligned} (\cdot\,,\cdot)_Y &= (\cdot\,,\cdot)_{H^1(\Omega)}\ , \\ \|\cdot\|_Y &= \|\cdot\|_{H^1(\Omega)}\ , \end{aligned} \tag{5.1}$$

where

$$\begin{aligned} (w,v)_{H^1(\Omega)} &\equiv \int_\Omega \nabla w\cdot\nabla v + wv, \quad \forall\, w,v \in H^1(\Omega)\ , \\ \|v\|_{H^1(\Omega)} &\equiv \int_\Omega |\nabla v|^2 + v^2, \quad \forall\, v \in H^1(\Omega)\ . \end{aligned} \tag{5.2}$$

More general inner products and norms can (and should) be considered, as discussed in Section 2.4.2.

We shall denote by Y' the dual space of Y. For a $g \in Y'$, the dual norm is given by

$$\|g\|_{Y'} = \sup_{v\in Y} \frac{g(v)}{\|v\|_Y}\ . \tag{5.3}$$

If we introduce the "representation" operator $\mathcal{Y}: Y' \rightarrow Y$ such that, for any $g \in Y'$,

$$(\mathcal{Y}g, v)_Y = g(v) \ , \tag{5.4}$$

then

$$\|g\|_{Y'} = \|\mathcal{Y}g\|_Y \ ; \tag{5.5}$$

this is simply a statement of the Riesz representation theorem.

We now introduce our parametrized bilinear form. We first define a parameter set $\mathcal{D}^\mu \subset \mathbb{R}^P$, a typical point in which — our input P-tuple — shall be denoted μ; we can then define, for any $\mu \in \mathcal{D}^\mu$, our bilinear form $a(\,\cdot\,,\,\cdot\,;\mu): Y \times Y \rightarrow \mathbb{R}$. We shall assume that a satisfies a continuity and inf-sup condition for all $\mu \in \mathcal{D}$, as we now state more precisely.

It shall prove convenient to state our hypotheses in terms of a "supremizing" operator $T^\mu: Y \rightarrow Y$. In particular, for any given $\mu \in \mathcal{D}^\mu$, and any $w \in Y$,

$$(T^\mu w, v)_Y = a(w, v; \mu), \quad \forall\, v \in Y \ ; \tag{5.6}$$

it is readily shown that

$$T^\mu w = \arg\sup_{v \in Y} \frac{a(w, v; \mu)}{\|v\|_Y} \ . \tag{5.7}$$

Furthermore, if we define the inf-sup (singular value) and continuity constants as

$$\beta(\mu) \equiv \inf_{w \in Y} \sup_{v \in Y} \frac{a(w, v; \mu)}{\|w\|_Y \, \|v\|_Y} \tag{5.8}$$

and

$$\gamma(\mu) \equiv \sup_{w \in Y} \sup_{v \in Y} \frac{a(w, v; \mu)}{\|w\|_Y \, \|v\|_Y} \ , \tag{5.9}$$

then,

$$\beta(\mu) = \inf_{w \in Y} \sigma(w; \mu) \ , \tag{5.10}$$

$$\gamma(\mu) = \sup_{w \in Y} \sigma(w; \mu) \ , \tag{5.11}$$

where

$$\sigma(w; \mu) \equiv \frac{\|T^\mu w\|_Y}{\|w\|_Y} \ . \tag{5.12}$$

Our assumptions are then: for some positive constant ε_s, $\varepsilon_s \leq \beta(\mu) \leq \gamma(\mu) < \infty$, $\forall\, \mu \in \mathcal{D}^\mu$.

We next define the bilinear form $b(\,\cdot\,,\,\cdot\,;\mu): Y \times Y \rightarrow \mathbb{R}$ as

$$b(w, v; \mu) = (T^\mu w, T^\mu v)_Y, \quad \forall\, w, v \in Y \ . \tag{5.13}$$

We then introduce the eigenproblem: Given $\mu \in \mathcal{D}^\mu$, find $\chi^i(\mu) \in Y$, $\lambda^i(\mu) \in \mathbb{R}$, $i = 1, \ldots, \infty$, such that

$$b(\chi^i(\mu), v; \mu) = \lambda^i(\mu)(\chi^i(\mu), v)_Y, \ \forall\, v \in Y \ , \tag{5.14}$$

$$\|\chi^i(\mu)\|_Y = 1 \ . \tag{5.15}$$

We shall, for convenience, assume that the spectrum is discrete (in actual practice we require only that the first few modes belong to the discrete component). In that case, we may assume that

$$b(\chi^i(\mu), \chi^j(\mu); \mu) = \lambda^i(\mu)(\chi^i(\mu), \chi^j(\mu))_Y = \lambda^i(\mu)\delta_{ij} \ , \tag{5.16}$$

where δ_{ij} is the Kronecker-delta symbol; that $0 < \lambda^1(\mu) \leq \lambda^2(\mu) \leq \cdots$; and that $Y = \text{span}\ \{\chi^i(\mu),\ i = 1, \ldots, \infty\}$. Note that, from (5.10)-(5.14), $\beta(\mu) = \sqrt{\lambda^1(\mu)}$; furthermore, $\gamma(\mu)$ is an upper bound for the spectrum.

We shall make the further assumption that a is "affine in the parameter" in the sense that, for some finite Q,

$$a(w, v; \mu) = \sum_{q=1}^{Q} \Theta_q(\mu)\, a_q(w, v) \ , \tag{5.17}$$

where $\Theta\colon \mathcal{D}^\mu \to \mathbb{R}^Q$ are differentiable *parameter-dependent* coefficient functions, and the $a_q\colon Y \times Y \to \mathbb{R}$, $1 \leq q \leq Q$, are *parameter-independent* bilinear forms. We define, for future reference,

$$D_{qp} = \max_{\mu \in \mathcal{D}^\mu} \left| \frac{\partial \Theta_q}{\partial \mu_p}\,(\mu) \right| \ , \tag{5.18}$$

for $1 \leq q \leq Q$, $1 \leq p \leq P$. Furthermore, we assume that the a_q are continuous in the sense that there exist positive finite constants Γ_q, $1 \leq q \leq Q$, such that

$$|a_q(w, v)| \leq \Gamma_q\, |w|_q\, |v|_q \ ; \tag{5.19}$$

here $|\cdot|_q\colon H^1(\Omega) \to \mathbb{R}$ are seminorms that satisfy

$$\left(\sum_{q=1}^{Q} |v|_q^2 \right)^{1/2} \leq C_Y^{1/2} \|v\|_Y, \quad \forall\, v \in Y \ , \tag{5.20}$$

where C_Y is a finite constant.

Finally, it directly follows from (5.6) and (5.17) that, for any $w \in Y$, $T^\mu w \in Y$ may be expressed as

$$T^\mu w = \sum_{q=1}^{Q} \Theta_q(\mu)\, T_q w \ , \tag{5.21}$$

where, for any $w \in Y$, $T_q w$, $1 \leq q \leq Q$, is given by

$$(T_q w, v)_Y = a_q(w, v), \quad \forall\, v \in Y \ . \tag{5.22}$$

Note that the operators $T_q\colon Y \to Y$ are independent of the parameter μ.

5.2.2 2.2 Problem Formulation

2.2.1 Weak Statement

We introduce an output functional $\ell \in Y'$ and "data" functional $f \in Y'$. Our weak statement of the partial differential equation is then: Given $\mu \in \mathcal{D}^\mu$, find

$$s = \ell(u(\mu)) \ , \tag{5.23}$$

where $u(\mu) \in Y$ satisfies

$$a(u(\mu), v; \mu) = f(v), \quad \forall\, v \in Y \ . \tag{5.24}$$

In the language of the introduction, $s(\mu)$ is our output, and $u(\mu)$ is our field variable.

In actual practice, we shall replace (5.23)–(5.24) with a truth approximation: Given $\mu \in \mathcal{D}^\mu$, find

$$s^{\mathcal{N}}(\mu) = \ell(u^{\mathcal{N}}(\mu)) \ ,$$

where $u^{\mathcal{N}}(\mu) \in Y^{\mathcal{N}} \subset Y$ satisfies

$$a(u^{\mathcal{N}}(\mu), v; \mu) = f(v), \quad \forall\, v \in Y^{\mathcal{N}} \ , \tag{5.25}$$

and $Y^{\mathcal{N}}$ is a finite element approximation subspace. We assume that $\mathcal{N}$ is chosen sufficiently large that $s^{\mathcal{N}}(\mu)$ and $u^{\mathcal{N}}(\mu)$ may be effectively equated with $s(\mu)$ and $u(\mu)$, respectively. We shall thus distinguish between $Y^{\mathcal{N}}$ and Y only in our discussion of computational complexity. (Note that issues associated with a possible continuous component to the spectrum of (5.14) may be addressed by considering Y as the limit of $Y^{\mathcal{N}}$, $\mathcal{N} \to \infty$.)

2.2.2 Reduced-Basis Approximation

The focus of the current paper is *a posteriori* error estimation. We shall thus take our reduced-basis approximation as *given*. In particular, we assume that we are provided with a reduced-basis approximation to $u(\mu)$, $u_N(\mu) \in W_N$, where

$$W_N = \text{span}\ \{\zeta_n \equiv u(\mu^n),\ 1 \le n \le N\} \ , \tag{5.26}$$

$S_N = \{\mu^1 \in \mathcal{D}^\mu, \ldots, \mu^N \in \mathcal{D}^\mu\}$, and $u(\mu^n)$ satisfies (5.24) (in practice, (5.25)) for $\mu = \mu^n$. It follows that $u_N(\mu)$ may be expressed as

$$u_N(\mu) = \sum_{n=1}^{N} u_{Nn}(\mu)\, \zeta_n \ . \tag{5.27}$$

The reduced-basis approximation to the output $s(\mu)$, $s_N(\mu)$, is given by $s_N(\mu) = \ell(u_N(\mu))$.

For the purposes of this paper, we shall consider only standard Galerkin projections: $a(u_N(\mu), v; \mu) = f(v)$, $\forall\, v \in W_N$. However, the *discrete* inf-sup parameter associated with the latter may not be "good," with corresponding detriment to the accuracy of $u_N(\mu)$ and hence $s_N(\mu)$. More sophisticated minimum-residual [44, 72] and in particular Petrov-Galerkin [45, 72] approaches restore (guaranteed) stability, albeit at some additional complexity and cost.

2.2.3 Error Estimation: Objective

We now wish to develop *a posteriori* error bounds $\Delta_N(\mu)$ and $\Delta_N^s(\mu)$ such that

$$\|e(\mu)\|_{H^1(\Omega)} \le \Delta_N(\mu) \ , \tag{5.28}$$

and

$$|s(\mu) - s_N(\mu)| = |\ell(e(\mu))| \le \Delta_N^s(\mu) \ , \tag{5.29}$$

where $e(\mu) \equiv u(\mu) - u_N(\mu)$. For the purposes of this paper, we shall focus on the $H^1(\Omega)$ bound, $\Delta_N(\mu)$, in terms of which $\Delta_N^s(\mu)$ can be expressed as $\|\mathcal{Y}\ell\|_Y \, \Delta_N(\mu)$; the latter may be significantly improved by the introduction of adjoint techniques [28, 62].

It shall prove convenient to introduce the notion of effectivity, defined (here) as

$$\eta_N(\mu) \equiv \frac{\Delta_N(\mu)}{\|e(\mu)\|_{H^1(\Omega)}} \ . \tag{5.30}$$

Our certainty requirement (5.28) may be stated as $\eta_N(\mu) \geq 1, \ \forall \, \mu \in \mathcal{D}^\mu$. However, for efficiency, we must also require $\eta_N(\mu) \leq C_\eta$, where $C_\eta \geq 1$ is a constant independent of N and μ; preferably, C_η is close to unity, thus ensuring that we choose the *smallest* N — and hence most economical — reduced-basis approximation consistent with the specified error tolerance.

5.2.3 2.3 A Posteriori Error Estimation

2.3.1 Error Bound

We assume that we are given a $\hat{\beta}(\mu)$ such that, for the given inner product $(\cdot, \cdot)_Y \equiv (\cdot, \cdot)_{H^1(\Omega)}$ (which in our previous papers [82, 61] would be denoted a "bound conditioner"),

$$\beta(\mu) \geq \hat{\beta}(\mu) \geq (1 - \tau) \, \varepsilon_s, \quad \forall \, \mu \in \mathcal{D}^\mu \ , \tag{5.31}$$

where $\tau \in]0, 1[$. We then define our error bound as

$$\Delta_N(\mu) \equiv \frac{\|\mathcal{Y} r(\,\cdot\,; \mu)\|_Y}{\hat{\beta}(\mu)} \ , \tag{5.32}$$

where

$$r(v; \mu) = f(v) - a(u_N(\mu), v; \mu), \quad \forall \, v \in Y \ , \tag{5.33}$$

is the residual associated with $u_N(\mu)$. Note it follows from (5.24) that (5.33) may be restated as

$$a(e(\mu), v; \mu) = r(v; \mu), \quad \forall \, v \in Y \ , \tag{5.34}$$

where we recall that $e(\mu) \equiv u(\mu) - u_N(\mu)$.

We can then state

Proposition 5. *For the error bound $\Delta_N(\mu)$ of (5.32), the effectivity satisfies*

$$1 \leq \eta_N(\mu) \leq \frac{\gamma(\mu)}{(1 - \tau) \, \varepsilon_s}, \quad \forall \, \mu \in \mathcal{D} \ , \tag{5.35}$$

for all $N \in \mathbb{N}$.

Proof It follows from (5.4), (5.6), and (5.34) that

$$\|\mathcal{Y} r(\,\cdot\,; \mu)\|_Y = \|T^\mu e(\mu)\|_Y \ . \tag{5.36}$$

Furthermore, from (5.12) we know that

$$\|e(\mu)\|_Y = \frac{\|T^\mu e(\mu)\|_Y}{\sigma(e(\mu);\mu)} \ , \tag{5.37}$$

and hence from (5.1), (5.30), (5.32), (5.36), and (5.37)

$$\eta^N(\mu) = \frac{\sigma(e(\mu);\mu)}{\hat{\beta}(\mu)} \ . \tag{5.38}$$

The result then directly follows from (5.10), (5.11), (5.31), and (5.38). □

We note that our proof (or bound) does not exploit any special properties of $e(\mu)$ (or $u_N(\mu)$).

It remains to develop our lower bound construction, $\hat{\beta}(\mu)$, and to demonstrate that both $\hat{\beta}(\mu)$ and $\|\mathcal{Y}r(\,\cdot\,;\mu)\|_Y$ may be computed efficiently (that is, in complexity *independent* of $\mathcal{N}$).

2.3.2 Inf-Sup Lower Bound Construction

Many of the most obvious eigenvalue approximation concepts are not relevant here, since we require a lower, not upper, bound. We thus develop a construction particularly suited to our context.

We assume that we are given a set of J parameter points, $\mathcal{L}_J \equiv \{\overline{\mu}^1 \in \mathcal{D}^\mu, \ldots, \overline{\mu}^J \in \mathcal{D}^\mu\}$, and associated set of polygonal regions $\mathcal{R}^{\overline{\mu}^j,\tau}$, $1 \le j \le J$, where

$$\mathcal{R}^{\overline{\mu},\tau} \equiv \{\mu \in \mathcal{D}^\mu \,|\mathcal{B}_q^{\overline{\mu}}(\mu) \le \frac{\tau}{C_Y}\,\beta(\overline{\mu}),\ 1 \le q \le Q\} \ , \tag{5.39}$$

and

$$\mathcal{B}_q^{\overline{\mu}}(\mu) = \Gamma_q \sum_{p=1}^{P} D_{qp}\,|\mu_p - \overline{\mu}_p| \ ; \tag{5.40}$$

we further assume that

$$\bigcup_{j=1}^{J} \mathcal{R}^{\overline{\mu}^j,\tau} = \mathcal{D}^\mu \ . \tag{5.41}$$

We then define $\mathcal{J}: \mathcal{D}^\mu \to \{1,\ldots,J\}$ such that, for a given μ, $\mathcal{R}^{\overline{\mu}^{\mathcal{J}(\mu)},\tau}$ is that region (or a selected region) which contains μ.

Our lower bound is then: Given $\mu \in \mathcal{D}^\mu$,

$$\hat{\beta}(\mu) = \beta(\overline{\mu}^{\mathcal{J}(\mu)}) - C_Y\,\mathcal{B}_{\max}^{\overline{\mu}^{\mathcal{J}(\mu)}}(\mu) \ , \tag{5.42}$$

where

$$\mathcal{B}_{\max}^{\overline{\mu}}(\mu) = \max_{q\in\{1,\ldots,Q\}} \mathcal{B}_q^{\overline{\mu}}(\mu) \tag{5.43}$$

for $\mathcal{B}_q^{\overline{\mu}}(\mu)$ defined in (5.40).

We can now state

Proposition 6. *The construction $\hat{\beta}(\mu)$ of (5.42) satisfies the inequality (5.31).*

Proof We first note that, for given $\overline{\mu}$ and $\mu \in \mathcal{D}^\mu$, and any $w \in Y$,

$$T^\mu w = T^{\overline{\mu}} w + \sum_{q=1}^{Q} (\Theta_q(\mu) - \Theta_q(\overline{\mu}))\, T_q w \ ,$$

where we have appealed to (5.21). It thus follows from (5.10), (5.12), and the triangle inequality that

$$\beta^2(\mu) \geq \inf_{w \in Y} \left\{ \left[\sigma(w;\overline{\mu}) - \frac{\left\| \sum_{q=1}^{Q} (\Theta_q(\mu) - \Theta_q(\overline{\mu}))\, T_q w \right\|_Y}{\|w\|_Y} \right]^2 \right\} . \tag{5.44}$$

It is then a simple matter to show, from (5.22), (5.18), (5.19), the Cauchy-Schwarz inequality, and (5.20), that

$$\left\| \sum_{q=1}^{Q} (\Theta_q(\mu) - \Theta_q(\overline{\mu}))\, T_q w \right\|_Y \leq C_Y\, \mathcal{B}^{\overline{\mu}}_{\max}(\mu)\, \|w\|_Y \ ; \tag{5.45}$$

it thus follows from (5.10), (5.11), and (5.44), (5.45) that

$$\beta^2(\mu) \geq \inf_{t \in [\beta(\overline{\mu}), \gamma(\overline{\mu})]} \left\{ \left[t - (C_Y\, \mathcal{B}^{\overline{\mu}}_{\max}(\mu)) \right]^2 \right\} \ . \tag{5.46}$$

Recall that (5.46) is valid for any $\overline{\mu}$ and $\mu \in \mathcal{D}^\mu$.

We now choose, for any given μ, $\overline{\mu} = \overline{\mu}^{\mathcal{J}(\mu)}$. We thus obtain

$$\beta^2(\mu) \geq \inf_{t \in [\beta(\overline{\mu}^{\mathcal{J}(\mu)}), \gamma(\overline{\mu}^{\mathcal{J}(\mu)})]} \left\{ \left[t - (C_Y\, \mathcal{B}^{\overline{\mu}^{\mathcal{J}(\mu)}}_{\max}(\mu)) \right]^2 \right\} \ . \tag{5.47}$$

However, from (5.39) and (5.43) we know that

$$C_Y\, \mathcal{B}^{\overline{\mu}^{\mathcal{J}(\mu)}}_{\max}(\mu) \leq \tau\, \beta(\overline{\mu}^{\mathcal{J}(\mu)}) \ ; \tag{5.48}$$

the infimizer of (5.47) is thus $\beta(\overline{\mu}^{\mathcal{J}(\mu)})$, yielding

$$\beta(\mu) \geq \beta(\overline{\mu}^{\mathcal{J}(\mu)}) - C_Y\, \mathcal{B}^{\overline{\mu}^{\mathcal{J}(\mu)}}_{\max}(\mu) \ . \tag{5.49}$$

The desired result, (5.31), immediately follows from (5.49), (5.42), (5.48), and $\beta(\mu) \geq \varepsilon_s$. □

It should be clear that our bound of $|\Theta(\mu) - \Theta(\overline{\mu})|$ is rather crude; a more careful treatment of this term, leading to correspondingly larger regions $\mathcal{R}^{\overline{\mu}^j,\tau}$, and hence smaller J, is described elsewhere.[50]

It may appear paradoxical to combine a linear approximation — in particular, with $O(1)$ *error* — to $\beta(\mu)$ with an exponentially convergent reduced-basis approximation — with *very small error* — to $u(\mu)$. In fact, it is not inconsistent: $|\beta(\mu) - \hat{\beta}(\mu)|/\beta(\mu) \sim O(1)$ is manifested as a 100% "error in the error" — and is acceptable; in contrast, $|s(\mu) - s_N(\mu)|/s(\mu) \sim O(1)$ is manifested as a 100% error in the solution — and is thus clearly unacceptable. In essense, the equation for the error $e(\mu)$, (5.34), permits relaxations — and hence rigorous yet inexpensive bounds — that can not be directly applied to the original equation for $u(\mu)$, (5.24).

2.3.3 Offline/Online Computational Procedure

Summary. The central computational aspect of our reduced-basis approach is an offline/online computational decomposition which separates the requisite calculations into two distinct stages. The complexity of the *offline* — or preprocessing — stage will depend on $\mathcal{N}$ (large), N, Q, and J; however, the complexity of the *online* stage — in which, given a new value of μ, we evaluate $s_N(\mu)$ and $\Delta_N^s(\mu)$ — will depend only on N, Q, and J. The absence of $\mathcal{N}$ dependence in the online stage translates, in many cases, into real-time response.

It is simple to show[62] that, for our Galerkin approximation, the *online* cost to evaluate $u_{Nn}(\mu)$, $1 \le n \le N$, and $s_N(\mu)$ is $O(QN^2) + O(N^3)$ and $O(N)$, respectively. We develop here similar estimates for $\Delta_N(\mu)$ (and hence $\Delta_N^s(\mu)$). In particular, we shall show that, in the *online* stage, $\|\mathcal{Y}r(\,\cdot\,;\mu)\|_Y$ and $\hat{\beta}(\mu)$ may be calculated in only $O(Q^2N^2)$ and $O(P \log J)$ operations, respectively. We also briefly address the associated *offline* complexity.

Calculation of the Dual Norm of the Residual. We first invoke (5.17) and (5.27) to write the residual (5.33) as

$$r(v;\mu) = f(v) - \sum_{q=1}^{Q}\sum_{n=1}^{N} \Theta_q(\mu) u_{Nn}(\mu) a_q(\zeta_n, v) \ . \tag{5.50}$$

It then follows from linear superposition that

$$\mathcal{Y}r(\,\cdot\,;\mu) = \hat{z}_{00} - \sum_{q=1}^{Q}\sum_{n=1}^{N} \Theta_q(\mu) u_{Nn}(\mu) \hat{z}_{qn} \tag{5.51}$$

where $(\hat{z}_{00}, v)_Y = f(v)$, $\forall\, v \in Y$, and

$$(\hat{z}_{qn}, v)_Y = a_q(\zeta_n, v), \quad \forall\, v \in Y \ , \tag{5.52}$$

for $1 \le q \le Q$, $1 \le n \le N$. We now insert the expression for $\mathcal{Y}r(\,\cdot\,;\mu)$ of (5.51) into the definition of the $\|\,\cdot\,\|_Y$ norm to obtain

$$\begin{aligned}
\|\mathcal{Y}r(\,\cdot\,;\mu)\|_Y^2 \ &= \ (\hat{z}_{00}, \hat{z}_{00})_Y \\
&- 2 \sum_{q,n=1}^{Q,N} \Theta_q(\mu) u_{Nn}(\mu) (\hat{z}_{qn}, \hat{z}_{00})_Y \\
&+ \sum_{q,n=1}^{Q,N} \sum_{q',n'=1}^{Q',N'} \Theta_q(\mu) \Theta_{q'}(\mu) u_{Nn}(\mu) u_{Nn'}(\mu) (\hat{z}_{qn}, \hat{z}_{q'n'})_Y .
\end{aligned}$$

The offline/online decompostion is now clear.

In the *offline* stage, we evaluate and store $(\hat{z}_{00}, \hat{z}_{00})_Y$, $(\hat{z}_{qn}, \hat{z}_{00})_Y$, and $(\hat{z}_{qn}, \hat{z}_{q'n'})_Y$, $1 \le q, q' \le Q$, $1 \le n, n' \le N$: the cost is $O(\mathcal{N}Q^2N^2)$ operations. Then, in the *online* stage, to compute $\|\mathcal{Y}r(\,\cdot\,;\mu)\|_Y$ — given any new value of μ — we need only perform summations over Q and N: the cost is $O(Q^2N^2)$ operations. (In many cases, domain decomposition may be exploited to further reduce the Q dependence — from quadratic to linear [74].) As required, the complexity of the online stage is independent of $\mathcal{N}$.

Calculation of the Inf-Sup Lower Bound. The offline/online decomposition for $\hat{\beta}(\mu)$ follows directly from the construction. In the *offline* stage, we must solve J "inf-sup" eigenproblems — calculate $\sqrt{\lambda^1(\mu)}$ from (5.14) for $\mu = \overline{\mu}^j$, $1 \le j \le J$: the cost is $O(\mathcal{N}^t J)$ for some exponent $t \ge 1$. In the *online* stage, we need only determine $\mathcal{J}(\mu)$ and then evaluate (5.42): the cost for the former is $O(P \log J)$ for a suitable indexing of the $\mathcal{R}^{\overline{\mu}^j,\tau}$, $1 \le j \le J$; the cost for the latter is $O(PQ)$. We explore the dependence of J on $\mathcal{D}^\mu$, ε_s, τ, and P in Sections 2.4 and 2.5 in the context of particular examples.

We implicitly assume in our estimates above that we know, *a priori*, how to choose $\mathcal{L}_J$ such that (5.41) will be satisfied. In practice, that will not be the case, and hence additional inf-sup eigenproblems (5.14) will need to be solved. However, for purposes of constructing the sample $\mathcal{L}_J$, we may exploit a *reduced-basis approximation* to (5.14) [72]; this inexpensive yet accurate surrogate greatly reduces the design costs.

5.2.4 2.4 Improvements

2.4.1 Model Helmholtz Problem: $P = 1$

To motivate our improvements, we consider a very simple Helmholtz problem: $\Omega \subset \mathbb{R}^d$, $Y = H_0^1(\Omega)$, $P = 1$, and

$$a(w, v; \mu) = \int_\Omega \nabla w \cdot \nabla v - \underbrace{\mu}_{\omega^2} \int_\Omega wv \ . \tag{5.53}$$

This model represents harmonic forcing of a "pinned" membrane at frequency $\omega = \sqrt{\mu}$. We may also identify $Q = 2$, $\Theta_1(\mu) = 1$, $\Theta_2 = \mu$, $a_1(w, v) = \int_\Omega \nabla w \cdot \nabla v$, $a_2(w, v) = -\int_\Omega wv$; furthermore, for $|w|_1^2 = \int_\Omega |\nabla w|^2$, $|w|_2^2 = \int_\Omega w^2$, we readily calculate $\Gamma_1 = 1$, $\Gamma_2 = 1$, and $C_Y = 1$. Recall that the latter are for the case in which $(\cdot, \cdot)_Y = (\cdot, \cdot)_{H^1(\Omega)}$, $\|\cdot\|_Y = \|\cdot\|_{H^1(\Omega)}$.

It is clear that (5.53) exhibits resonances at $\sigma_1, \sigma_2, \ldots$, where $0 < \sigma_1 \le \sigma_2 < \ldots$ are the real positive eigenvalues of $a_1(\,\cdot\,,\,\cdot\,)$ relative to $-a_2(\,\cdot\,,\,\cdot\,)$ (for example, for $\Omega =]0, 1[$, $\sigma_k = k^2\pi^2$). It can be further shown that the spectrum of (5.14) is indeed discrete, and that $\beta(\mu)$ is piecewise linear — a "sawtooth." More quantitatively, we obtain

$$\beta(\mu) = \frac{|\sigma_k - \mu|}{\sigma_k + 1} \tag{5.54}$$

for μ in a neighborhood of σ_k.

For purposes of illustration, we shall consider $\mathcal{D}^\mu = [\sigma^*(1 - \Lambda), \sigma^*(1 - \varepsilon_s)]$ for some small $\Lambda > \varepsilon_s$ and for some $\sigma^* \equiv \sigma_{j^*} \gg 1$. It is then readily shown from (5.39), (5.40), and (5.54) that J — the number of regions $\mathcal{R}^{\overline{\mu}^j,\tau}$ required to satisfy our "coverage" constraint, (5.41) — is given by

$$J(\sigma^*, \Lambda, \varepsilon_s, \tau) \sim \frac{\sigma^* \ln(\frac{\Lambda}{\varepsilon_s})}{\tau} \tag{5.55}$$

for sufficiently large σ^*. Recall that $\tau \in]0, 1[$.

2.4.2 Bound Conditioner

The first evident problem with (5.55) is the dependence on σ^* — in particular, since σ^* large will often be the case of interest. This can be remedied by better choice of our bound conditioner. To wit, we now define $(\cdot\,,\cdot\,)_Y = (\cdot\,,\cdot\,)_{\tilde{\mu}}$, $\|\cdot\|_Y = \|\cdot\|_{\tilde{\mu}}$, where

$$(\cdot\,,\cdot\,)_{\tilde{\mu}} \equiv \int_\Omega \nabla w \cdot \nabla v + \tilde{\mu} \int_\Omega wv \ , \tag{5.56}$$

and $\|\cdot\|^2_{\tilde{\mu}} \equiv (\cdot\,,\cdot\,)_{\tilde{\mu}}$; we require that $\tilde{\mu} \geq 1$. In what follows subscript $\tilde{\mu}$ refers to quantities (previously introduced but now) defined relative to the $(\cdot\,,\cdot\,)_Y = (\cdot\,,\cdot\,)_{\tilde{\mu}}$ and $\|\cdot\|_Y = \|\cdot\|_{\tilde{\mu}}$ inner product and norm, respectively.

We first note that $\|e(\mu)\|_{H^1(\Omega)} \leq \|e(\mu)\|_{\tilde{\mu}}$ (since $\tilde{\mu} \geq 1$). We next note that, from the definition of the dual norm, (5.4)-(5.5),

$$\|\mathcal{Y}r(\,\cdot\,;\mu)\|_{\tilde{\mu}_2} \leq \|\mathcal{Y}r(\,\cdot\,;\mu)\|_{\tilde{\mu}_1} \tag{5.57}$$

for $\tilde{\mu}_2 \geq \tilde{\mu}_1$. Finally, it can be shown that (5.54) must now be replaced with

$$\beta_{\tilde{\mu}}(\mu) = \frac{|\sigma_k - \mu|}{\sigma_k + \tilde{\mu}} \ . \tag{5.58}$$

Assuming (for our purposes here) that $\beta_{\tilde{\mu}}(\mu)/\hat{\beta}_{\tilde{\mu}}(\mu)$ does not depend on $\tilde{\mu}$, it follows from (5.32), (5.57), and (5.58) that $\Delta_{N,\tilde{\mu}=\sigma^*} \leq 2\Delta_{N,\tilde{\mu}=1}$. Therefore, if we bound $\|e(\mu)\|_{H^1(\Omega)}$ by $\Delta_{N,\tilde{\mu}=\sigma^*}$ — which from Proposition 1 is a bound for $\|e(\mu)\|_{\tilde{\mu}=\sigma^*}$ and hence $\|e(\mu)\|_{H^1(\Omega)}$ — rather than by $\Delta_{N,\tilde{\mu}=1}$ (as before), the effectivity should increase by no more than roughly a factor of two.

However, there will be a significant decrease in the requisite J. In particular, we can now take $|w|_1^2 = \int_\Omega |\nabla w|^2$, $|w|_2^2 = \hat{\mu} \int_\Omega w^2$, in terms of which $\Gamma_1 = 1$, $\Gamma_2 = 1/\hat{\mu}$, and $C_Y = 1$. It then follows from (5.39), (5.40) and (5.58) that

$$J_{\hat{\mu}=\sigma^*}(\sigma^*, \Lambda, \varepsilon_s, \tau) \sim \frac{2\ln(\frac{\Lambda}{\varepsilon_s})}{\tau} \ ; \tag{5.59}$$

we have successfully eliminated the σ^* dependence, at little detriment to the effectivity. The reason is simple: $\chi^1(\mu)$ is high-wavenumber, and thus we may add a significant L^2 contribution to our bound conditioner without adversely affecting the inf-sup parameter; this additional L^2 term does, however, significantly improve our continuity constants — on which our lower bound construction is critically dependent. These arguments apply to Helmholtz problems generally; however, for larger ranges of frequency, we will need different bound conditioners for different subdomains of $\mathcal{D}^\mu$ — if we wish to retain the $\mathcal{D}^\mu$-independence of J.

2.4.3 Deflation

The second debilitating aspect of (5.55) is the $-\ln(\varepsilon_s)$ dependence. This, too, can be eliminated, albeit at slightly increased effort. The strategy is deflation: remove the most dangerous components of the error — those near $\chi^1(\mu), \chi^2(\mu), \ldots$, — thereby increasing the *effective*

inf-sup parameter. The latter should improve our bounds and effectivity; but, more importantly, it will remove the ε_s dependence from (5.55) — our regions will be generally larger, and will not shrink to zero as we approach resonances (or, at most, except very near resonances).

To begin, we define the "trial" and "test" spaces $U_M(\mu), V_M(\mu)$, with $U_M(\mu) = \text{span}\{\phi_i(\mu), 1 \le i \le M\}$, $V_M(\mu) = \text{span}\{\xi_i(\mu),\ 1 \le i \le M\}$. We next introduce the correction $\delta_M^D(\mu) \in U_M(\mu)$ through the Petrov-Galerkin projection,

$$a(\delta_M^D(\mu), v; \mu) = r(v; \mu), \quad \forall\, v \in V_M(\mu)\ . \tag{5.60}$$

We can then define the "deflated" reduced-basis approximation as

$$u_{N,M}^D(\mu) = u_N(\mu) + \delta_M^D(\mu)\ , \tag{5.61}$$

with corresponding output $s_{N,M}^D(\mu) = \ell(u_{N,M}^D(\mu))$. The associated residual is now given by

$$r^D(v; \mu) \equiv f(v) - a(u_{N,M}^D(\mu), v; \mu), \quad \forall\, v \in Y\ ; \tag{5.62}$$

and thus

$$a(e^D(\mu), v; \mu) = r^D(v; \mu), \quad \forall\, v \in Y\ , \tag{5.63}$$

where the deflated error is defined as $e^D(\mu) \equiv u(\mu) - u_{N,M}^D(\mu)$.

For the purposes of this paper, we shall consider a particular set of spaces $U_M(\mu), V_M(\mu)$: for a given $\overline{\mu}$ (which shall ultimately depend on μ), we set

$$U_M(\mu) = \text{span}\ \{\chi^1(\overline{\mu}), \ldots, \chi^M(\overline{\mu})\}\ , \tag{5.64}$$

$$V_M(\mu) = \text{span}\ \{T^\mu\chi^1(\overline{\mu}), \ldots, T^\mu\chi^M(\overline{\mu})\}\ . \tag{5.65}$$

We then note that, from (5.61)–(5.63), (5.6), and (5.13),

$$b(e^D(\mu), v; \mu) = r(T^\mu v; \mu) - a(\delta_M^D(\mu), T^\mu v), \quad \forall\, v \in Y\ . \tag{5.66}$$

It then follows from (5.60), (5.64), and (5.65) that

$$b(e^D(\mu), v; \mu) = 0, \quad \forall\, v \in U_M(\mu)\ , \tag{5.67}$$

and therefore $e^D(\mu) \in Z_M(\mu)$ where $Z_M(\mu) = \{v \in Y \mid b(v, \chi^m(\overline{\mu}); \mu) = 0,\ 1 \le m \le M\}$. We conclude from (5.12), (5.13), and (5.67) that

$$\|e^D(\mu)\|_Y \le \frac{\|T^\mu e^D(\mu)\|_Y}{\beta_M(\mu)}\ ,$$

where

$$\beta_M(\mu) \equiv \inf_{w \in Z_M(\mu)} \sigma(w; \mu)\ . \tag{5.68}$$

It follows from (5.4), (5.63), and (5.6) that

$$\|\mathcal{Y} r^D(\,\cdot\,; \mu)\|_Y = \|T^\mu e^D(\mu)\|_Y$$

and therefore $\|e^D(\mu)\|_Y \leq \Delta_{N,M}(\mu)$ where

$$\Delta_{N,M}(\mu) \equiv \frac{\|\mathcal{Y} r^D(\,\cdot\,;\mu)\|_Y}{\hat{\beta}_M(\mu)} \ ,$$

and $\hat{\beta}_M(\mu)$ is a lower bound for $\beta_M(\mu)$.

It remains to construct $\hat{\beta}_M(\mu)$. Revisiting the arguments in the proof of Proposition 2, we note that

$$\beta_M^2(\mu) \geq \inf_{w \in Z_M(\mu)} \left\{ \left[\sigma(w;\overline{\mu}) - \frac{\left\| \sum_{q=1}^{Q} (\Theta_q(\mu) - \Theta_q(\overline{\mu}))\, T_q w \right\|_Y}{\|w\|_Y} \right]^2 \right\} . \tag{5.69}$$

It then follows from (5.68), (5.11), (5.69), and (5.45), that for any $\overline{\mu}$ and $\mu \in \mathcal{D}^\mu$,

$$\beta_M^2(\mu) \geq \inf_{t \in [\beta_M(\overline{\mu}), \gamma(\overline{\mu})]} \left\{ \left[t - (C_Y\, \mathcal{B}^{\overline{\mu}}_{\max}(\mu)) \right]^2 \right\} \ .$$

It thus follows that, if $\mu \in \mathcal{R}_M^{\overline{\mu},\tau}$ where

$$\mathcal{R}_M^{\overline{\mu},\tau} \equiv \{ \mu \in \mathcal{D}^\mu \mid \mathcal{B}_q^{\overline{\mu}}(\mu) \leq \frac{\tau}{C_Y} \sqrt{\lambda^{M+1}(\overline{\mu})}, \quad 1 \leq q \leq Q \} \ , \tag{5.70}$$

then

$$\hat{\beta}_M(\mu) \equiv \sqrt{\lambda^{M+1}(\overline{\mu})} - C_Y\, \beta^{\overline{\mu}}_{\max}(\mu) \tag{5.71}$$

satisfies $\beta_M(\mu) \geq \hat{\beta}_M(\mu) \geq (1-\tau)\sqrt{\lambda^{M+1}(\overline{\mu})}$; here we have invoked (5.16) to deduce that $\beta(\overline{\mu}) = \sqrt{\lambda^{M+1}(\overline{\mu})}$. This result is much improved over (5.31) since $\sqrt{\lambda^{M+1}(\overline{\mu})}$ $(M \geq 1)$ will, generically, remain much larger than ε_s — indeed, $O(1)$ — for all μ.

In terms of our model problem of Section 2.4.1, we completely eliminate the ε_s dependence of (5.55) (or (5.59)) — J is now $O(1)$. For this model problem a simple deflation $M = 1$ suffices. However, for more complicated problems — that exhibit mode-crossing — $M = 2$ or even $M = 3$ (near where, say, three modes intersect) higher-order deflations will prove beneficial; we shall observe such a case in Section 2.5.

It remains to address two issues concerning $\delta_M^D(\mu)$. First, we must show that $\|\delta_M^D(\mu)\|_{H^1(\Omega)} \leq$ Const $\|e(\mu)\|_{H^1(\Omega)}$. This is readily demonstrated — for a Const typically considerably better that ε_s^{-1} — thanks to the choice of perforce stable Petrov-Galerkin spaces, (5.64), (5.65). Second, we must show that $\delta_M^D(\mu)$ can be calculated efficiently online. This is indeed the case. The matrix associated with the left-hand side of (5.60) can be expressed, for the spaces (5.64), (5.65), as

$$\begin{aligned} a(\chi^i(\overline{\mu}), T^\mu \chi^j(\overline{\mu}); \mu) &= \sum_{q=1}^{Q} \Theta_q(\mu) a_q(\chi^i(\overline{\mu}), T^\mu \chi^j(\overline{\mu})) \\ &= \sum_{q=1}^{Q} \sum_{q'=1}^{Q} \Theta_q(\mu)\, \Theta_{q'}(\mu)\, a_q(\chi^i(\overline{\mu}), T_{q'} \chi^j(\overline{\mu})) \ , \end{aligned} \tag{5.72}$$

which may thus be formed in $O(Q^2M^2)$ operations; here we have invoked (5.17) and (5.21). Similar arguments indicate that the right-hand side of (5.60) may be formed in $O(Q^2MN)$ operations. We conclude that the additional online complexity is negligible (e.g., compared to calculation of $\|\mathcal{Y}r(\cdot;\mu)\|_Y$).[†]

5.2.5 2.5 Numerical Results

2.5.1 Model Helmholtz Problem: $P = 2$

We consider a model problem which illustrates a potential application of our methodology: real-time, reliable solution of partial differential equations in the service of non-destructive evaluation (and adaptive mission design) "in the field."

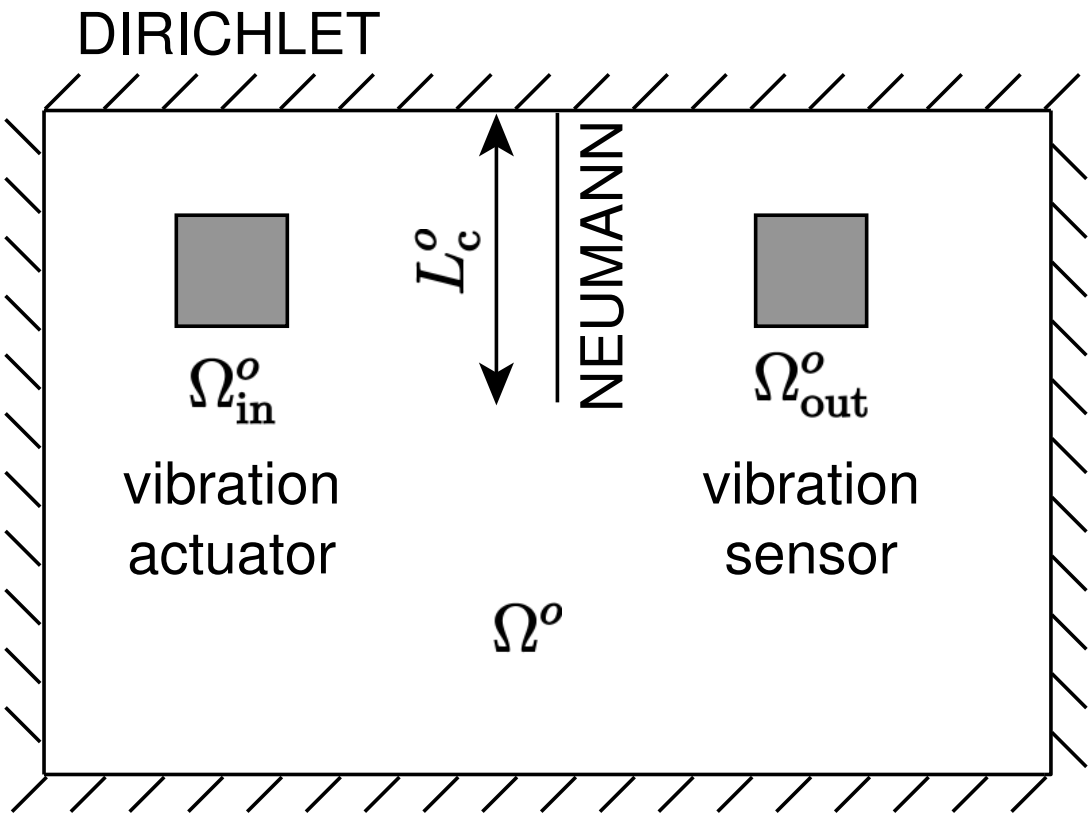


Figure 5.1: Membrane with crack of length L_c^o under harmonic excitation at frequency ω.

Our domain $\Omega^o(L_c^o)$ is a rectangular membrane $[0, \frac{3}{2}] \times [0, 1]$ with a "crack" Γ^o_{crack} of length L_c^o, as shown in Figure 1. We assume that the boundary of the memrane is "pinned" except on

[†] The offline expense will be increased somewhat, not so much due to the $\lambda^2(\overline{\mu}^j)$, $\chi^2(\overline{\mu}^j)$ (say for $M = 1$) — in particular, since J will now be much smaller — but rather due to the $JQ^2(M^2 + MN)\mathcal{N}$ operations required for the inner products associated with the deflation correction (5.60). Relatedly, the online storage will also increase, by $JQ^2(M^2 + MN)$. This effect can be very significantly reduced if we replace $\chi^1(\overline{\mu}), \chi^2(\overline{\mu}), \ldots$ in $U_M(\mu), V_M(\mu)$ with a reduced-basis approximation to these quantities [72]; however, there will be a concomitant slight increase in online cost.

the "stress-free" crack. The membrane is forced over a patch $\Omega^o_{\rm in}$ at frequency ω; the response is measured over the patch $\Omega^o_{\rm out}$. Our problem statement is then: Given $L^o_{\rm c}$ and ω^2, find

$$s(\omega^2, L^o_{\rm c}) = \frac{1}{|\Omega^o_{\rm out}|}\int_{\Omega^o_{\rm out}} u^o(\omega^2, L^o_{\rm c}) \tag{5.73}$$

where $u^o(\omega^2, L^o_{\rm c}) \in X^o(L^o_{\rm c})$ satisfies

$$\begin{aligned}\int_{\Omega^o(L^o_{\rm c})} \nabla u^o(\omega^2, L^o_{\rm c})\cdot\nabla v - \omega^2 u^o(\omega^2, L^o_{\rm c})\, v \\ = \frac{1}{|\Omega^o_{\rm in}|}\int_{\Omega^o_{\rm in}} v, \quad \forall\, v \in X^o(L^o_{\rm c})\ ;\end{aligned} \tag{5.74}$$

here $X^o(L^o_{\rm c}) = \{H^1(\Omega^o(L^o_{\rm c}))\,|\,v|_{\partial\Omega^o(L^o_{\rm c})\setminus\Gamma^o_{\rm crack}} = 0\}$. Clearly, an elastic plate (and more realistic outputs) would be a much more relevant model; our methodology directly applies to this case as well.

We now map $\Omega^o(L^o_{\rm c})$ to a reference domain independent of $L^o_{\rm c}$, $\Omega \equiv \Omega^o(L^o_{\rm c} = \frac{1}{2})$, through piecewise-affine subdomain co-ordinate tranformations. The resulting equations can then be cast in the desired form (5.23), (5.24), and (5.17) for $P = 2$, $\mu = (\omega^2, L^o_{\rm c})$, $\mathcal{D}^\mu = \{\mu \in [25, 50]\times[.3, .7]\,|\,\beta(\mu) \geq \varepsilon_s \cong 0.005\}$, and $Q = 8$. (Note the relatively large value of Q, relative to P, originates in the (accuracy) requirement that $\Omega^o_{\rm in}$ and $\Omega^o_{\rm out}$ map to invariant images in Ω, the reference domain.) We do not give here the detailed expressions for either the $\Theta_q(\mu)$ or $a_q(\,\cdot\,,\,\cdot\,)$, $1 \leq q \leq Q$.

We briefly comment on the structure of this model problem. For a *given* $L^o_{\rm c}$, the bilinear form is, apart from several scaling factors, identical to (5.53) of Section 2.4.1. We conclude that, for any given $(\omega^2, L^o_{\rm c})$, the spectrum of (5.14) is discrete; and furthermore that, for given $\overline{L}^o_{\rm c}$, $\beta(\omega^2, \overline{L}^o_{\rm c})$ is a piecewise-linear function of ω^2. However, if we now permit $L^o_{\rm c}$ to also vary, the behavior of $\beta(\omega^2, L^o_{\rm c})$ is no longer trivial. In particular, unlike in Section 2.4, we can no longer characterize $\beta(\omega^2, L^o_{\rm c})$ in terms of a few (more generally, denumerable) "resonance" eigenvalues — our lower bound constructions are now required. We show in Figure 2 a contour plot of $\beta(\omega^2, L^o_{\rm c})$ over $\mathcal{D}^\mu$; the dark lines indicate "excluded regions" that contain resonances — lines (more generally, $P-1$ dimensional manifolds) along which $\beta(\mu)$ vanishes. (The resonances will be eliminated if we incorporate damping or radiation into our model; however, the inf-sup parameter may still remain small.)

2.5.2 The Inf-Sup Lower Bound

For this first set of tests we do not consider a complete "paving" of $\mathcal{D}^\mu$; rather $\underline{\mathcal{L}}_J \equiv \mathcal{L}^{\rm TEST}_J$ comprises 25 points $\overline{\mu}^1, \ldots, \overline{\mu}^J$ equidistributed along a segment $\overline{(25, 0.4)(50, 0.6)}$ in $\mathcal{D}^\mu$. We shall consider the following cases:

I	$\|\cdot\|_Y = \|\cdot\|_{H^1(\Omega)}$,	no deflation;
II	$\|\cdot\|_Y = \|\cdot\|_{\tilde{\mu}=\omega^2_{\rm min}=25}$,	no deflation;
III	$\|\cdot\|_Y = \|\cdot\|_{\tilde{\mu}=\omega^2_{\rm min}=25}$,	deflation, $M = 1$;
IV	$\|\cdot\|_Y = \|\cdot\|_{\tilde{\mu}=\omega^2_{\rm min}=25}$,	deflation, $M = 2$;
V	$\|\cdot\|_Y = \|\cdot\|_{\tilde{\mu}=\omega^2_{\rm min}=25}$,	deflation, $M = 3$.

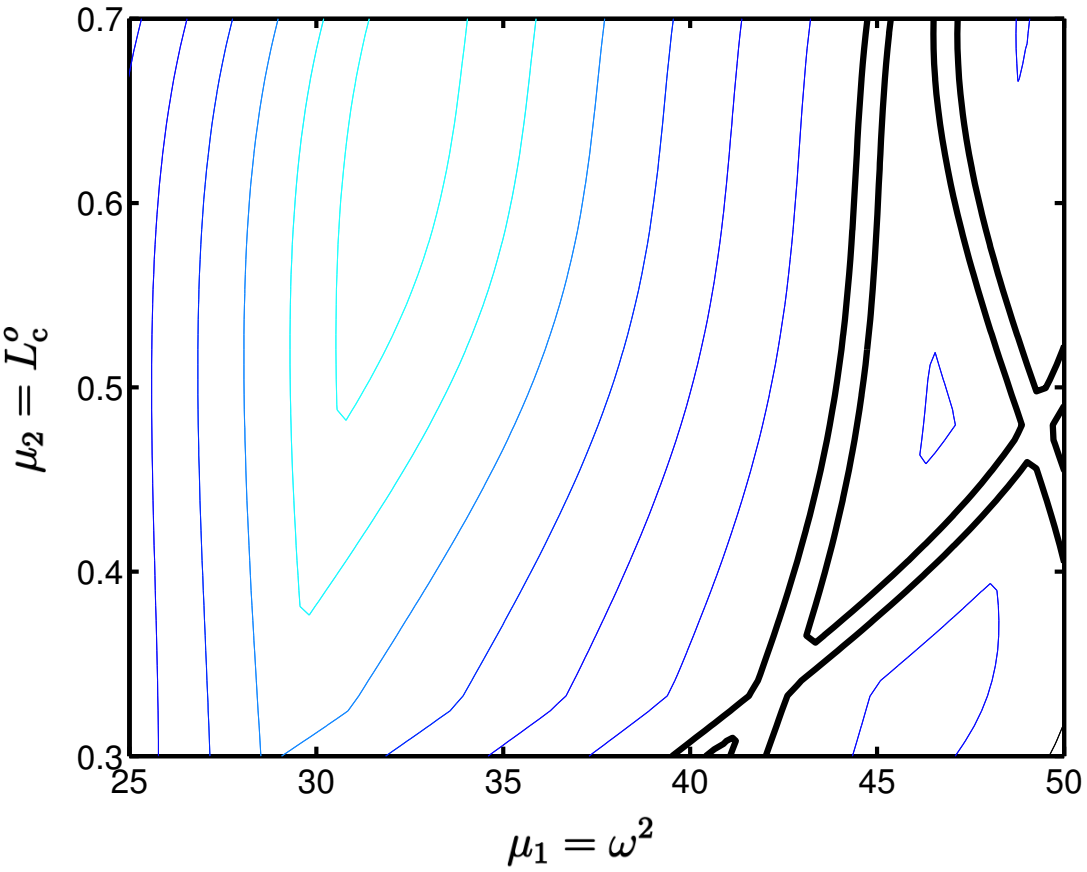


Figure 5.2: Contours of $\beta(\omega^2, L_c^o)$ over $\mathcal{D}^\mu$ for the (cracked membrane) model Helmholtz problem.

For the definition of $\|\cdot\|_{\tilde{\mu}}$ and of deflation see Sections 2.4.2 and 2.4.3, respectively. In all cases, we choose $\tau = 3/4$.

We plot in Figure 3 the polygons $\mathcal{R}^{\overline{\mu}^j,\tau}$ for cases I, II, III, and V. Note we shift vertically the regions for cases I (highest), II, and III for purposes of easy comparison. (In actual fact, for given j, the center of $\mathcal{R}^{\overline{\mu}^j,\tau}$ for all cases (I, II, II, and V) is $\mu = \overline{\mu}^j$; in Figure 3, only $\mathcal{R}^{\overline{\mu}^j,\tau}$ for case V is honestly (vertically) located.) First, we observe that the "correct" bound conditioner (I $\rightarrow$ II) considerably increases the size of the regions; furthermore, this effect will be even more dramatic for higher frequency ranges. Second, we observe that some deflation (II $\rightarrow$ III) further improves the situation; and *sufficient* deflation (III $\rightarrow$ V) greatly improves the situation, in particular as we approach resonance. Note that although IV performs better than III, only with V do we have sufficient deflation in the sense that all dangerous modes are neutralized — it is clear from Figure 3 that *three* modes are "active" near the end of our segment $\overline{(25, 0.4)(50, 0.6)}$. Increasing M beyond 3 has little effect as all modes now appear "far away."

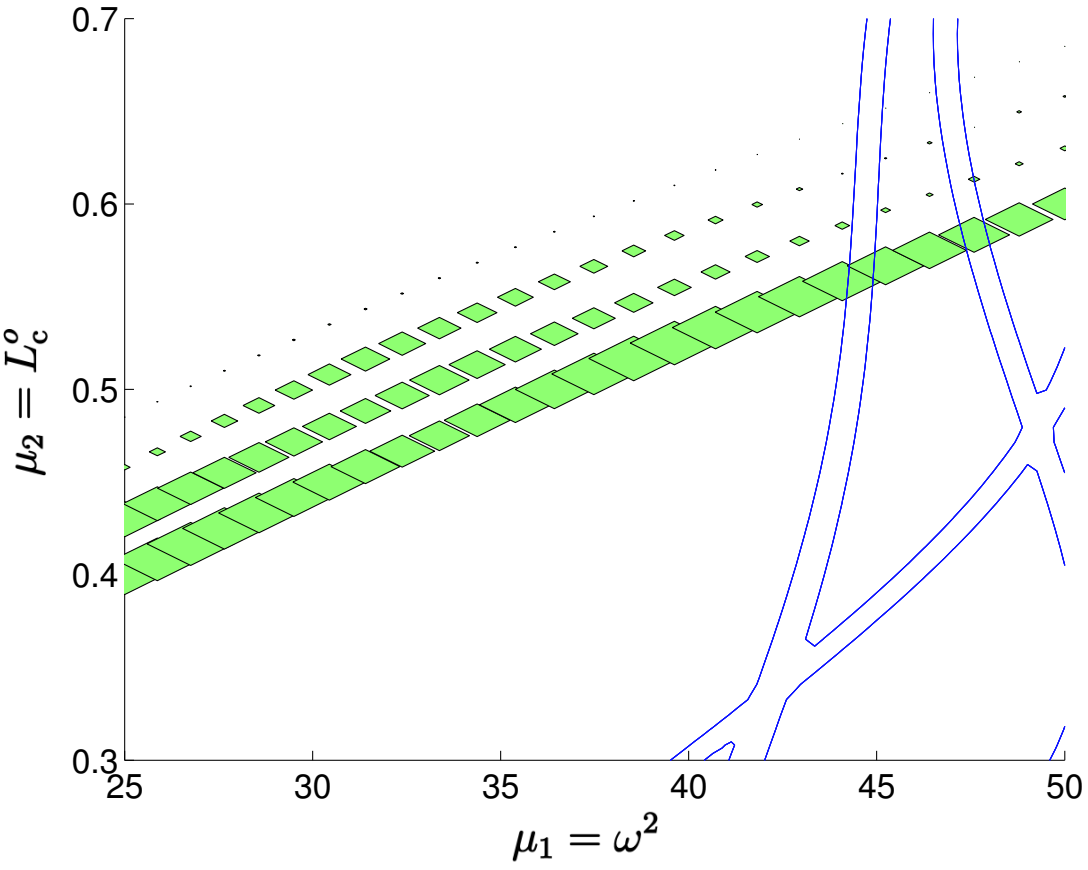


Figure 5.3: Polygons $\mathcal{R}^{\overline{\mu}^j,\tau}$ for cases I, II, III, and V.

In short, a combination of "tuned" bound conditioners and sufficient deflation greatly increases the size of our regions — in particular at high frequencies and near resonance, respectively — such that we can expect J to be roughly independent of $\mathcal{D}^\mu$ and ε^s. We show in Figure 4, for case V, a (more or less) complete paving of $\mathcal{D}^\mu$ by regions $\mathcal{R}^{\overline{\mu}^j,\tau}$,

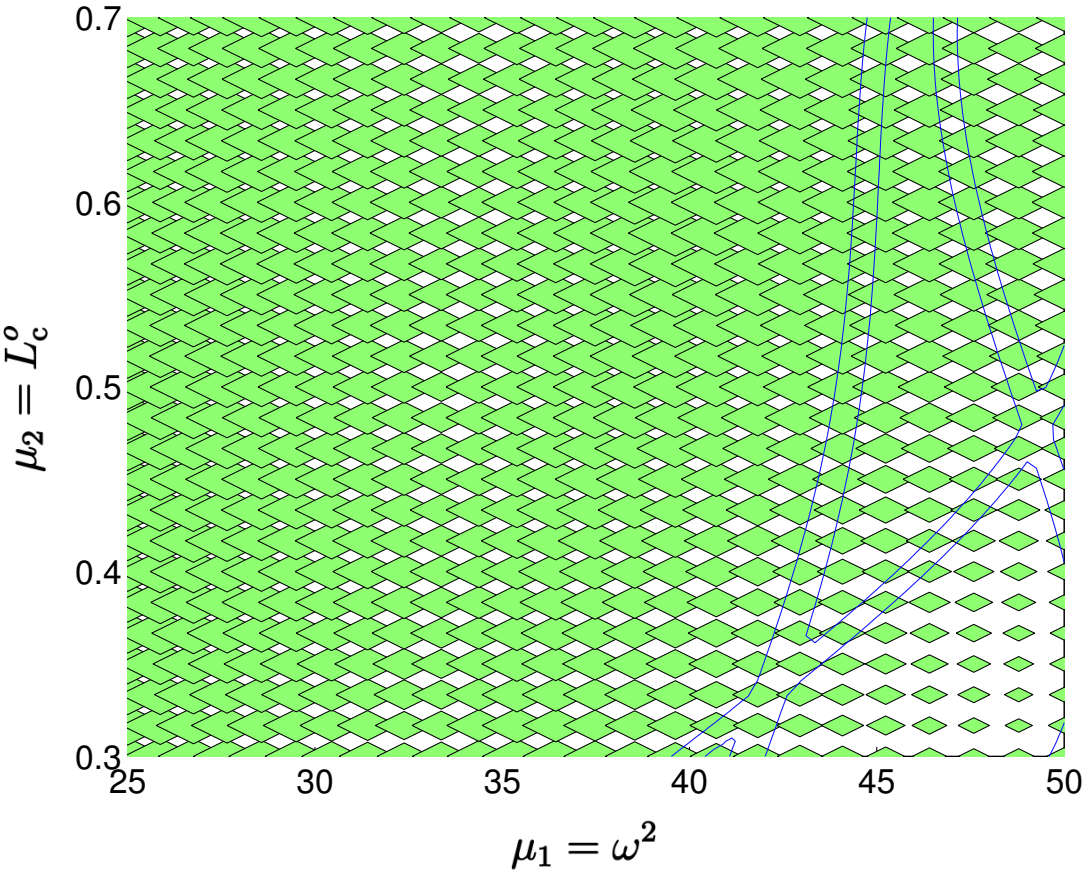


Figure 5.4: A (more or less) complete "paving" of $\mathcal{D}^\mu$ for case V.

$1 \le j \le J = 625$. Note we anticipate that J can be further reduced by roughly 5–10 based on less conservative polygons which exploit the monotonicity of the $\Theta_q(\mu)$ [50].

The result of Figure 4 is disappointing in one aspect. In actual fact, $\beta(\mu)$ varies significantly (locally) only in the *one* direction perpendicular to the $P-1$ dimensional "resonance" manifolds. Our construction, even with deflation, remains isotropic, and thus $J \sim (\text{Const} > 1)^P$. In contrast, an anisotropic construction would be relatively insensitive to P. Unfortunately, at present, we see no way to capture this anisotropy efficiently while maintaining the requisite lower bound property.

2.5.3 Error Bounds and Effectivity

We consider here a point $\overline{\mu}^{\text{TEST}}$ which lies within a region $\mathcal{R}^{\overline{\mu}^j,\tau}$ for all cases I, II, III, IV, and V. This point, $\overline{\mu}^{\text{TEST}} = (42.95, 0.55)$, is quite close to a resonance.

We present in Tables 5.1 and 5.2 $\|e^{(D)}(\mu^{\text{TEST}})\|_{H^1(\Omega)}/\ \|u(\mu^{\text{TEST}})\|_{H^1(\Omega)}$ and $\eta_N(\mu^{\text{TEST}})$, respectively, as a function of N for Cases I, II, and V. Note S_N, our reduced-basis approximation sample, is chosen here log-randomly over $\mathcal{D}^\mu$; see Sections 3 and 4 for discussion of better optimal/adaptive alternatives for sample design. As expected from the arguments of Section 2.4.2, the bound conditioner has little (detrimental) effect on the effectivity. And, as expected from the arguments of Section 2.3.3, deflation has a modest (respectively, significant) positive effect on the error (respectively, effectivity).

	$\|e^{(D)}(\mu^{\text{TEST}})\|_{H^1(\Omega)}/\|u(\mu^{\text{TEST}})\|_{H^1(\Omega)}$		
N	I	II	V
4	2.7×10^{-1}	2.7×10^{-1}	8.4×10^{-2}
8	9.6×10^{-2}	9.6×10^{-2}	3.0×10^{-2}
12	6.2×10^{-2}	6.2×10^{-2}	5.8×10^{-2}
16	9.0×10^{-3}	9.0×10^{-3}	8.5×10^{-3}
20	3.5×10^{-4}	3.5×10^{-4}	3.5×10^{-4}
24	2.7×10^{-4}	2.7×10^{-4}	2.7×10^{-4}
28	1.2×10^{-4}	1.2×10^{-4}	1.2×10^{-4}
32	2.1×10^{-5}	2.1×10^{-5}	2.1×10^{-5}
36	8.9×10^{-6}	8.9×10^{-6}	8.9×10^{-6}
40	3.0×10^{-6}	3.0×10^{-6}	3.0×10^{-6}

Table 5.1: The normalized error as a function of N for cases I, II, and V.

We close with two related remarks. First, even if the regions $\mathcal{R}^{\overline{\mu}^j,\tau}$ include a resonance, this does not imply that the error (or error bound) remains finite as we approach the resonance. Second, round-off errors will become increasingly important, and ultimately dominant, in the very immediate vicinity of resonances; in particular, as we approach extremely close to a resonance, we may observe effectivities below unity. The reason is clear: our error bound assumes (5.67); however, in finite precision, this condition will be violated — and the resulting "round-off" error amplified by $1/\beta(\mu)$. Exact orthogonalization recovers the theoretical result — $\eta_N(\mu) \ge 1$; in more realistic models, damping will provide the necessary "cut-off."

	$\eta_N(\mu)$		
N	I	II	V
4	26.7	35.3	9.0
8	29.3	41.1	10.1
12	129.4	145.1	12.2
16	101.3	132.0	11.0
20	98.2	141.0	11.1
24	97.9	143.1	11.2
28	100.6	148.3	11.6
32	102.8	153.5	12.0
36	103.2	154.2	12.1
40	102.3	153.0	12.0

Table 5.2: The effectivity $\eta_N(\mu)$ as a function of N for cases I, II, and V.

5.3 3 Cubically Nonlinear Poisson Problem

5.3.1 3.1 Preliminaries

We consider a suitably regular domain $\Omega \subset \mathbb{R}^2$ with boundary $\partial\Omega = \partial\Omega_{\rm R} \cup \partial\Omega_{\rm N}$. We set $Y = H^1(\Omega)$, and $(\cdot\,,\cdot)_Y = (\cdot\,,\cdot)_{H^1(\Omega)}$ and $\|\cdot\|_Y = \|\cdot\|_{H^1(\Omega)}$ (as defined in (5.2)). The dual space of Y will be denoted Y', with norm defined as in (5.3) or, equivalently, (5.4), (5.5).

We next define our parameter set $(\mu_1, \mu_2) \in \mathcal{D}^\mu \equiv [\mu_1^{\min} > 0, \mu_1^{\max}] \times [\mu_2^{\min} > 0, \mu_2^{\max}] \subset \mathbb{R}^{P=2}_+$. Then, for any $\mu \in \mathcal{D}^\mu$, $a(\,\cdot\,,\,\cdot\,;\mu)\colon Y \times Y \to \mathbb{R}$ is given by

$$a(w, v; \mu) = a^{\rm L}(w, v; \mu) + a^{\rm NL}(w, v), \quad \forall\, w, v \in Y\ , \tag{5.75}$$

where

$$a^{\rm L}(w, v; \mu) = \mu_1 \int_\Omega \nabla w \cdot \nabla v + \mu_2 \int_{\partial\Omega_{\rm R}} wv\ , \tag{5.76}$$

and

$$a^{\rm NL}(w, v) = \int_\Omega w^3 v \tag{5.77}$$

represent a "Poisson-Robin" operator and cubic nonlinearity, respectively.

We note that $a^{\rm L}$ is continuous,

$$\sup_{w\in Y}\sup_{v\in Y} \frac{a^{\rm L}(w, v; \mu)}{\|w\|_Y\, \|v\|_Y} \le \gamma(\mu) < \infty, \quad \forall\, \mu \in \mathcal{D}^\mu\ , \tag{5.78}$$

symmetric, and coercive,

$$0 < \varepsilon_s \le \alpha(\mu) = \inf_{w\in Y} \frac{a^{\rm L}(w, w; \mu)}{\|w\|_Y^2},\ \forall\, \mu \in \mathcal{D}^\mu\ . \tag{5.79}$$

Furthermore, $a^{\rm L}(\,\cdot\,,\,\cdot\,;\mu)$ depends affinely on the parameter μ — the expansion (5.17) applies with $Q = 2$ and $\Theta_q = \mu_q$. In fact, our treatment of this section applies to any continuous, coercive, affine $a^{\rm L}(\,\cdot\,,\,\cdot\,;\mu)$.

5.3.2 3.2 Problem Formulation

3.2.1 Weak Statement

We introduce an output functional $\ell \in Y'$ and "data" functional $f \in Y'$; for our model problem we take $\ell(v) = f(v) = \int_{\partial\Omega_N} v$. Our weak statement of the partial differential equation is then: Given μ, find

$$s = \ell(u(\mu)) \ , \tag{5.80}$$

where $u(\mu) \in Y$ satisfies

$$a(u(\mu), v; \mu) = f(v), \quad \forall\, v \in Y \ . \tag{5.81}$$

In the language of the introduction, $s(\mu)$ is our output, and $u(\mu)$ is our field variable. It can be shown [19] that (5.75), (5.81) admits a unique solution.

As for our Helmholtz problem, in actual practice we replace $s(\mu)$ and $u(\mu)$ with corresponding "truth" Galerkin approximations $s^{\mathcal{N}}(\mu)$ and $u^{\mathcal{N}}(\mu)$, respectively (see Section 2.2.1).

3.2.2 Reduced-Basis Approximation

The focus of the current paper is *a posteriori* error estimation. We shall thus take our reduced-basis approximation as *given*. In particular, we assume that we are provided with a reduced-basis approximation to $u(\mu)$, $u_N(\mu) \in W_N$, where

$$W_N = \text{span}\ \{\zeta_n \equiv u(\mu^n),\ 1 \le n \le N\} \ , \tag{5.82}$$

$S_N = \{\mu^1 \in \mathcal{D}^\mu, \ldots, \mu^N \in \mathcal{D}^\mu\}$, and $u(\mu^n)$ satisfies (5.81) for $\mu = \mu^n$. It follows that $u_N(\mu)$ may be expressed as

$$u_N(\mu) = \sum_{n=1}^{N} u_{Nn}(\mu)\, \zeta_n \ . \tag{5.83}$$

The reduced-basis approximation to the output $s(\mu)$, $s_N(\mu)$, is given by $s_N(\mu) = \ell(u_N(\mu))$.

For our equations (5.75), (5.81), standard Galerkin projection is the best choice — as we shall see, there are no stability issues — and we thus select this (simplest) option: $a(u_N(\mu), v; \mu) = f(v)$, $\forall\, v \in W_N$.

3.2.3 Error Estimation: Objective

As for Helmholtz, we wish to provide an *a posteriori* error bound $\Delta_N(\mu)$ for $\|e(\mu)\|_{H^1(\Omega)}$ such that the effectivity, (5.30), satisfies

$$1 \le \eta_N(\mu) \le C_\eta \ , \tag{5.84}$$

for C_η independent of N and μ — and preferably close to unity. Error bounds for $|s(\mu) - s_N(\mu)|$ may also be developed.

5.3.3 3.3 A Posteriori Error Estimation

3.3.1 Error Bound

We assume that we are given an $\hat{\alpha}: \mathcal{D}^\mu \rightarrow \mathbb{R}_+$ such that

$$\alpha(\mu) \geq \hat{\alpha}(\mu) \geq (1-\tau)\,\varepsilon_s, \quad \forall\, \mu \in \mathcal{D}^\mu\ , \tag{5.85}$$

for given $\tau \in]0,1[$. We then define our error bound as

$$\Delta_N(\mu) \equiv \frac{\|\mathcal{Y} r(\,\cdot\,;\mu)\|_Y}{\hat{\alpha}(\mu)}\ , \tag{5.86}$$

where

$$r(v;\mu) \equiv f(v) - a(u_N(\mu), v; \mu), \quad \forall v \in Y\ , \tag{5.87}$$

is the residual.

We can then state

Proposition 7. *For the error bound $\Delta_N(\mu)$ of (5.86),*

$$\|e(\mu)\|_{H^1(\Omega)} \leq \Delta_N(\mu), \quad \forall\, \mu \in \mathcal{D}^\mu\ , \tag{5.88}$$

for all $N \in \mathbb{N}$.

Proof We know from (5.75), (5.76), (5.77) that (5.87) may be written as

$$a^{\mathrm{L}}(e(\mu), v; \mu) + \int_\Omega \left(u^3(\mu) - u_N^3(\mu)\right) v = r(v;\mu),\ \forall\, v \in Y\ , \tag{5.89}$$

where $e(\mu) \equiv u(\mu) - u_N(\mu)$. We now take $v = e(\mu)$, note that

$$\int_\Omega \left(u^3(\mu) - u_N^3(\mu)\right)\left(u(\mu) - u_N(\mu)\right) \geq 0\ , \tag{5.90}$$

and invoke (5.79) and (5.85) to obtain

$$\hat{\alpha}(\mu)\|e(\mu)\|_Y^2 \leq r(e(\mu);\mu)\ . \tag{5.91}$$

The desired result then follows from (5.5) and (5.91) (recall that $\|\cdot\|_Y \equiv \|\cdot\|_{H^1(\Omega)}$). □

We do not include here a uniform upper bound for the effectivity; however, it is clear from (5.89) that, as $\|r(\,\cdot\,;\mu)\|_{Y'} \rightarrow 0$, $\eta_N(\mu) \leq \gamma(\mu)/(1-\tau)\varepsilon_s$.

3.3.2 Coercivity Lower Bound Construction

Our approach to the inf-sup lower bound, described in Section 2.3.2, can also be adapted to general coercive problems [50]. For our purposes here, however, we consider a simple variant that exploits the monotonicity of $\alpha(\mu)$.

In particular, it can be shown that, for $a^{\mathrm{L}}(\,\cdot\,,\,\cdot\,;\mu)$ as defined in (5.76), $\alpha(\mu_1) \leq \alpha(\mu_2)$ for $\mu_2 \geq \mu_1$ (in the sense of each component); the proof follows directly from the Rayleigh

quotient definition, (5.79). Thus, given a sample $\mathcal{L}_J \equiv \{\overline{\mu}^1 = \mu^{\min}, \ldots, \overline{\mu}^J\}$, we may define a lower bound as

$$\hat{\alpha}(\mu) \equiv \max_{\{j\in\{1,\ldots,J\} \,|\, \overline{\mu}^j \le \mu\}} \alpha(\overline{\mu}^j) \ . \tag{5.92}$$

The best distribution of points — to minimize J given τ and our requirement (5.85) — is logarithmic. Further details on these and related bound conditioners for coercive problems may be found elsewhere.[82]

3.3.3 Offline/Online Computational Procedure

Summary. Our nonlinear problem admits an offline/online decomposition quite similar to that for linear problems. The key new issue is the higher order summations that perforce arise within the Galerkin context. Our focus here will be on efficient (or as efficient as possible) treatment of these new terms.

By way of summary, the *online* complexity to calculate $u_{Nn}(\mu)$, $1 \le n \le N$, and subsequently $s_N(\mu) = \ell(u_N(\mu))$, scales as $K^{\rm iter}(2N^2 + N^3 + N^4)$; here $K^{\rm iter}$ is the number of Newton iterations required to solve for $u_{Nn}(\mu)$, $1 \le n \le N$. The N^4 dependence — which arises due to the nonlinearity — is not pleasant, but for reasonably small N, not debilitating. In constrast, the online complexity to calculate $\Delta_N(\mu)$ scales, to leading order for "large" N, as $C_6 N^6$; the N^6 dependence — which arises due to the nonlinear contribution to the dual norm of the residual — is now more daunting. Fortunately, $C_6 = 1/72$, and thus, again for modest N, efficiency is preserved.

We focus our attention here on the dual norm calculation, and in particular on the origin of the constant C_6.

Calculation of the Dual Norm of the Residual. We first invoke (5.75), (5.76),(5.77), and (5.83) to write

$$\begin{aligned} r(v;\mu) = f(v) &- \sum_{n=1}^{N} u_{Nn}(\mu)\, a^{\rm L}(\zeta_n, v;\mu) \\ &- \sum_{n,n',n''=1}^{N} u_{Nn}(\mu) u_{Nn'}(\mu) u_{Nn''}(\mu) \int_\Omega \zeta_n \zeta_{n'} \zeta_{n''} v \quad . \end{aligned} \tag{5.93}$$

It thus follows from (5.76), (5.93) and linear superposition that

$$\begin{aligned} \mathcal{Y}r(\,\cdot\,;\mu) = \hat{z}^{\rm L}_{00} &- \mu_1 \sum_{n=1}^{N} u_{Nn}(\mu) \hat{z}^{\rm L}_{1n} - \mu_2 \sum_{n=1}^{N} u_{Nn}(\mu) \hat{z}^{\rm L}_{2n} \\ &- \sum_{n,n',n''=1}^{N} u_{Nn}(\mu) u_{Nn'}(\mu) u_{Nn''}(\mu)\, \hat{z}^{\rm NL}_{nn'n''} \ , \end{aligned} \tag{5.94}$$

where (for example) $\hat{z}^{\rm NL}_{nn'n''} \in Y$ satisfies

$$(\hat{z}^{\rm NL}_{nn'n''}, v)_Y = \int_\Omega \zeta_n \zeta_{n'} \zeta_{n''} v, \quad \forall\, v \in Y \ . \tag{5.95}$$

We thus obtain

$$\|\mathcal{Y}r(\,\cdot\,;\mu)\|_Y^2 = \cdots + \sum_{n,n',n''=1}^{N} \sum_{m,m',m''=1}^{N} u_{Nn}(\mu)\cdots u_{Nm''}(\mu)\,(\hat{z}^{\rm NL}_{nn'n''}, \hat{z}^{\rm NL}_{mm'm''})_Y \ , \tag{5.96}$$

where we shall focus only on the highest order (and hence most expensive) summations.

Obviously, naïve treatment of (5.96) directly yields N^6 operations. However, there are many symmetries that can be exploited. In particular, we note from (5.95) that $\hat{z}^{\rm NL}_{nn'n''} = \hat{z}^{\rm NL}_{mm'm''}$ for any $mm'm''$ triplet which is a permutation of $nn'n''$. We denote by $\mathcal{P}_3^N$ the set of unique *ordered* 3-tuples of integers $j \in \{1,\ldots,N\}$ — (j,j',j'') such that $j \le j' \le j''$; by $\mathcal{T}$ the cardinality of $\mathcal{P}_3^N$; and by $\Pi_k = (\Pi_k^1, \Pi_k^2, \Pi_k^3)$, $1 \le k \le \mathcal{T}$, the members of $\mathcal{P}_3^N$. We can thus write our sum of (5.96) as

$$\sum_{k=1}^{\mathcal{T}} \sum_{k'=1}^{\mathcal{T}} C_{\Pi_k} C_{\Pi_{k'}}\, u_{N\Pi_k^1}(\mu) u_{N\Pi_k^2}(\mu) u_{N\Pi_k^3} \times u_{N\Pi_{k'}^1}(\mu) u_{N\Pi_{k'}^2}(\mu) u_{N\Pi_{k'}^3}(\hat{z}^{\rm NL}_{\Pi_k}, \hat{z}^{\rm NL}_{\Pi_{k'}})_Y \ , \tag{5.97}$$

where the C_{Π_k} are multiplicity constants.

The online complexity of (5.97) clearly scales as $\mathcal{T}^2/2$. It is readily shown that the cardinality of $\mathcal{P}_\kappa^N$, the set of unique ordered κ-tuples of integers $j \in \{1,\ldots,N\}$ is given by

$$\mathbb{T}(N,\kappa) = \frac{((N-1)+\kappa)!}{(N-1)!\,\kappa!} \ . \tag{5.98}$$

In our particular case, $\mathcal{T} = \mathbb{T}(N,3) \sim N^3/6$ for N large; our sum (5.97) may thus be performed in $N^6/72$ operations — a considerable improvement over the naïve estimate of N^6. In general, $\mathbb{T}(N,\kappa) \sim N^6/\kappa!$: in *relative* terms, higher order (e.g., u^κ) nonlinearities thus enjoy greater economies; however, in *absolute* terms, $\mathbb{T}(N,\kappa)$ will grow very rapidly with N for larger κ. We must be content with relatively low-order (or low-order approximations of) nonlinearities.

5.3.4 3.4 Numerical Results

3.4.1 Model Problems

Our model problem has already been specified in Sections 3.1 and 3.2. It remains only to specify the physical domain, $\Omega \subset \mathbb{R}^2$ — a "tee"-shaped region with $\partial\Omega_N$ at the root — and the parameter domain — $\mu_1^{\min} = .1$, $\mu_1^{\max} = 10$, $\mu_2^{\min} = .01$, $\mu_2^{\max} = 1$. Note the nonlinearity will be most significant for μ_1 and μ_2 small.

3.4.2 Adaptive Reduced-Basis Approximation

Given the higher powers of N that now appear in our complexity estimates, it is crucial (both as regards online and offline effort) to control N more tightly. To this end, we may gainfully apply our *a posteriori* error bounds adaptively.

We first construct, *offline*, an approximation that, over most of the domain, exhibits an error (say, here, in the H^1-norm) less than $\varepsilon_d^{\text{prior}}$: we begin with a first sample point $\mu^1(S_{N'=1} = \{\mu^1\})$; we next (inexpensively) evaluate $\Delta_{N'=1}(\mu)$ over a large test sample of parameter points in $\mathcal{D}^\mu, \Xi^{\text{prior}}$; we then choose for μ^2 (and hence $S_{N'=2} = \{\mu^1, \mu^2\}$) the maximizer of $\Delta_{N'=1}(\mu)$ over Ξ^{prior}; we now repeat this process until the maximum of $\Delta_{N'=N^{\text{prior}}}(\mu)$ over Ξ^{prior} is less than $\varepsilon_d^{\text{prior}}$. Then, *online*, given a new value of the parameter, μ, and an error tolerance $\varepsilon_d^{\text{post}}(\mu)$, we essentially repeat this adaptive process — but now our sample points are drawn from $S_{N^{\text{prior}}}$, and the test sample is a singleton — μ. We typically choose $\varepsilon_d^{\text{prior}} \ll \varepsilon_d^{\text{post}}(\mu)$ since our prior test sample is not exhaustive; and therefore, typically, $N^{\text{post}}(\mu) \leq N^{\text{prior}}$.

We present in Table 5.3 the normalized error $\|e(\mu^*)\|_Y / \|u(\mu^*)\|_Y$, as a function of N, for the (log) random and adaptive sampling processes (note that, in the results for the random sampling process, the sample S_N is different for each N). We also indicate in Table 5.3 the online computational time,[†] which is largely independent of the sampling process but very *strongly* dependent on N (see Section 3.3.3). Here $\mu^* = \mu^*(N)$ is the point in Ξ^{prior} at which the maximum error bound $\Delta_N(\mu)$ occurs — note $\mu^*(N)$ will be different for the different sampling strategies. We observe that the adaptive sampling procedure yields higher accuracy at lower N; and that even these modest reductions in N can translate into measurable performance improvements. For purposes of comparison, calculation of $s^{\mathcal{N}}(\mu)$ requires 50 seconds — and thus, even for an accuracy of 2.0%, the reduced-basis approach is two to three orders of magnitude faster (marginally) than conventional techniques.

N	$\frac{\|e(\mu^*)\|_Y}{\|u(\mu^*)\|_Y}$ "Random"	$\frac{\|e(\mu^*)\|_Y}{\|u(\mu^*)\|_Y}$ "Adaptive"	Time (ms)
2	5.47×10^{-1}	4.38×10^{-1}	10.9
4	2.17×10^{-1}	1.28×10^{-1}	11.4
6	8.55×10^{-2}	7.27×10^{-2}	13.4
8	1.68×10^{-1}	5.10×10^{-2}	18.6
10	1.06×10^{-1}	2.09×10^{-2}	32.6

Table 5.3: Error bound for the cubically nonlinear Poisson problem for random and adaptive samples.

Of course, in actual practice, the savings indicated in Table 5.3 can only be realized if our error estimators are true bounds ($\eta_N(\mu) \geq 1$), and good bounds ($\eta_N(\mu) \approx 1$). We show in Table 5.4 the (normalized) error bound $\Delta_N(\mu^*)/\|u(\mu^*)\|_Y$, and effectivity, $\eta_N(\mu^*)$, as a function of N (for the adaptive case); as before, μ^* is the point in Ξ^{prior} at which the maximum error bound occurs. We observe that we do indeed obtain bounds, but that the bounds are not too sharp.

The main cause of the higher effectivities is the relatively small value of $\alpha(\mu)$ (and hence $\hat{\alpha}(\mu)$) for low μ_1 and μ_2. We present in Table 5.5 the effectivities for three test points, $\mu^{\text{TEST},1} = (0.01, 0.1)$, $\mu^{\text{TEST},2} = (0.1, 1)$, and $\mu^{\text{TEST},3} = (1, 10)$, for the $N = 10$ adaptive sample; except near $\mu_2 = .01$, the effectivities are quite close to unity. Numerous remedies exist for low μ_1, μ_2; we thus do not dwell on this further here.

[†] The calculations were performed on a Pentium®4 2.4GHz processor.

N	$\frac{\Delta_N(\mu^*)}{\|u(\mu^*)\|_Y}$	$\eta_N(\mu^*)$
2	$7.42 \times 10^{+1}$	169.3
4	$1.01 \times 10^{+1}$	79.3
6	$4.11 \times 10^{+0}$	56.6
8	$1.16 \times 10^{+0}$	22.8
10	5.34×10^{-1}	25.6

Table 5.4: Error bound and effectivity for the cubically nonlinear Poisson problem for an adaptive sample.

μ	$\frac{\|e(\mu)\|_Y}{\|u(\mu)\|_Y}$	$\frac{\Delta_N(\mu)}{\|u(\mu)\|_Y}$	$\eta_N(\mu)$
$\mu^{\text{TEST},1}$	1.45×10^{-4}	5.18×10^{-3}	35.7
$\mu^{\text{TEST},2}$	7.37×10^{-3}	4.33×10^{-2}	5.9
$\mu^{\text{TEST},3}$	6.91×10^{-3}	1.52×10^{-2}	2.2

Table 5.5: Error, error bound, and effectivity for the cubically nonlinear Poisson problem for the $N = 10$ adaptive sample; $\mu^{\text{TEST},1} = (0.01; 0.1), \mu^{\text{TEST},2} = (0.1; 1)$, and $\mu^{\text{TEST},3} = (1; 10)$.

5.4 4. The Burgers Equation

5.4.1 4.1 Preliminaries

We consider the domain $\Omega =]0, 1[$. We set $Y = H_0^1(\Omega)$, and $(\cdot\,,\cdot)_Y = (\cdot\,,\cdot)_{H^1(\Omega)}$, $\|\cdot\|_Y = \|\cdot\|_{H^1(\Omega)}$. The dual space of Y will be denoted Y', with the norm defined as in (5.3), or, equivalently, (5.4), (5.5).

In this case we have a single parameter, $\mu \in \mathcal{D}^\mu \equiv [\mu^{\min} > 0, \mu^{\max}] \subset \mathbb{R}_+^{P=1}$. For any $\mu \in \mathcal{D}^\mu$. $a(\cdot\,,\cdot\,;\mu)\colon Y \times Y \to \mathbb{R}$ is given by

$$a(w, v; \mu) = a^{\text{L}}(w, v; \mu) + a^{\text{NL}}(w, w, v), \quad \forall\, v \in Y\ , \tag{5.99}$$

where

$$a^{\text{L}}(w, v; \mu) \equiv \mu a_0(w, v) = \mu \int_0^1 w_x v_x \tag{5.100}$$

and

$$a^{\text{NL}}(w, z, v) = -\frac{1}{2} \int_0^1 w z v_x \tag{5.101}$$

are bilinear and trilinear forms, respectively.

For a given $z \in Y$, we define the bilinear form — associated to the derivative of our operator — $d(\cdot\,,\cdot\,;z;\mu)\colon Y \times Y \to \mathbb{R}$ as

$$d(w, v; z; \mu) = a^{\text{L}}(w, v; \mu) + 2a^{\text{NL}}(z, w, v)\ . \tag{5.102}$$

It shall prove convenient to introduce the supremizing operator $T^{z;\mu}: Y \to Y$ such that, for any $w \in Y$,

$$(T^{z;\mu}w, v)_Y = d(w, v; z; \mu), \quad \forall\, v \in Y\ ; \tag{5.103}$$

it is readily shown that

$$T^{z;\mu}w = \arg\sup_{v\in Y} \frac{d(w, v; z; \mu)}{\|v\|_Y}\ . \tag{5.104}$$

Furthermore, for

$$\beta^z(\mu) \equiv \inf_{w\in Y} \sup_{v\in Y} \frac{d(w, v; z; \mu)}{\|w\|_Y\, \|v\|_Y}\ , \tag{5.105}$$

and

$$\gamma^z(\mu) \equiv \sup_{w\in Y} \sup_{v\in Y} \frac{d(w, v; z; \mu)}{\|w\|_Y\, \|v\|_Y}\ , \tag{5.106}$$

we readily derive

$$\beta^z(\mu) = \inf_{w\in Y} \sigma^z(w; \mu) \le \sup_{w\in Y} \sigma^z(w; \mu) = \gamma^z(\mu)\ ,$$

where

$$\sigma^z(w; \mu) \equiv \frac{\|T^{z,\mu}w\|_Y}{\|w\|_Y}\ . \tag{5.107}$$

The development parallels Helmholtz except that we must include our linearization point z in the definitions.

Finally, we assume that we are given a constant ρ such that, for all $v \in Y\ (= H_0^1(\Omega))$,

$$\|v\|_{L^4(\Omega)} \le \rho\|v\|_Y\ ; \tag{5.108}$$

the existence of a finite ρ (for $\Omega \subset \mathbb{R}^{d=1,2,3}$) is guaranteed by the continuous embedding of Y in $L^4(\Omega)$ [68]. In $\mathbb{R}^1$ we also have $Y \subset L^\infty(\Omega)$; we thus readily derive, from the Cauchy-Schwarz inequality, $\rho = \frac{1}{2}$. In higher dimensions it is not difficult to develop bounds for ρ in general geometries.

5.4.2 4.2 Problem Formulation

4.2.1 Weak Statement

We introduce an output functional $\ell \in Y'$ and "data" functional $f \in Y'$. Our weak statement of the partial differential equation is then: Given μ, find

$$s(\mu) = \ell(u(\mu))\ , \tag{5.109}$$

where $u(\mu) \in Y$ satisfies

$$a(u(\mu), v; \mu) = f(v), \quad \forall\, v \in Y\ . \tag{5.110}$$

Equations (5.99), (5.100), (5.101), (5.110) are a very good model for the incompressible Navier-Stokes equations (see below), which is our ultimate goal. For sufficiently large μ, (5.99), (5.110) — and the incompressible Navier-Stokes equations — have a unique solution; for smaller μ, we can encounter non-uniqueness — multiple solution branches may exist.

As for our Helmholtz problem, in actual practice we replace $s(\mu)$ and $u(\mu)$ with corresponding "truth" Galerkin approximations $s^{\mathcal{N}}(\mu)$ and $u^{\mathcal{N}}(\mu)$, respectively (see Section 2.2.1).

4.2.2 Reduced-Basis Approximation

We assume that we are provided with a reduced-basis approximation to $u(\mu)$, $u_N(\mu) \in W_N$, where

$$W_N = \text{span}\ \{\zeta_n \equiv u^I(\mu^n),\ 1 \le n \le N\}\ , \tag{5.111}$$

$S_N = \{\mu^1 \in \mathcal{D}^\mu, \ldots, \mu^N \in \mathcal{D}^\mu\}$, and $u^I(\mu^n)$ satisfies (5.110) for $\mu = \mu^n$. It follows that $u_N(\mu)$ may be expressed as

$$u_N(\mu) = \sum_{n=1}^{N} u_{Nn}(\mu)\, u^I(\mu^n)\ . \tag{5.112}$$

The reduced-basis approximation to the output $s(\mu)$, $s_N(\mu)$, is given by $s_N(\mu) = \ell(u_N(\mu))$. Note $u^I(\mu^n)$ refers to solutions of (5.99), (5.110), which are assumed to reside on a "first" (particular) branch; although we do not dwell here on possible bifurcation structure, other "parametric manifolds" (say, $u^{II}(\mu)$) may, in general, exist.

For the purposes of this paper, we shall consider only standard Galerkin projections: $a(u_N(\mu), v; \mu) = f(v)$, $\forall\ v \in Y$. However, the *discrete* inf-sup parameter associated with the latter may not be "good," with corresponding detriment to the accuracy of $u_N(\mu)$ and hence $s_N(\mu)$. More sophisticated minimum-residual [44, 72] and in particular Petrov-Galerkin [45, 72] approaches restore (guaranteed) stability, albeit at some additional complexity and cost.

We comment that, for the case in which geometry is fixed and only viscosity varies, our reduced-basis approximation (and associated error estimation) procedure for the Burgers equation directly translates to the full incompressible Navier-Stokes equations — in particular, a divergence- (and hence pressure-) free formulation of the incompressible Navier-Stokes equations.

4.2.3 Error Estimation: Objective

As for the cubically nonlinear Poisson problem, we would like to provide an error bound $\Delta_N(\mu)$ for $\|e(\mu)\|_{H^1(\Omega)}$ (and, relatedly, bound $\Delta_N^s(\mu)$ for $|s(\mu) - s_N(\mu)|$) such that the effectivity satisfies (5.84). However, as we shall see, our error statement will no longer be unqualified — there will be a "choice" that reflects the possible existence of multiple solution branches.

5.4.3 4.3 A Posteriori Error Estimation

4.3.1 Preliminaries

We first define, in a slight abuse of notation, $T^\mu \equiv T^{u_N(\mu);\mu}$ where $T^{z;\mu}$ is given by (5.103); relatedly, we define $\beta(\mu) \equiv \beta^{u_N(\mu)}(\mu)$, $\gamma(\mu) \equiv \gamma^{u_N(\mu)}(\mu)$, and $\sigma(w;\mu) = \sigma^{u_N(\mu)}(w;\mu)$, for $\beta^z(\mu)$, $\gamma^z(\mu)$, and $\sigma^z(w;\mu)$ defined in (5.105), (5.106), and (5.107). We assume that $\beta(\mu) \ge \varepsilon_s$, $\forall\ \mu \in \mathcal{D}^\mu$. In what follows, we will explicitly highlight the N-dependence of $\beta(\mu)$, $\gamma(\mu)$, and $\sigma(w;\mu)$ only in those places where this dependence is either not obvious or potentially problematic.

We shall also require operators $T_n: Y \to Y$, $0 \le n \le N$: for any $w \in Y$,

$$(T_n w, v)_Y = a_n(w, v), \quad \forall\, v \in Y \ , \tag{5.113}$$

where

$$a_0(w, v) = \int_0^1 w_x v_x \ , \tag{5.114}$$

and

$$a_n(w, v) = 2a^{\rm NL}(\zeta_n, w, v), \quad 1 \le n \le N \ . \tag{5.115}$$

It follows from (5.102), (5.103), (5.113), (5.114), and (5.115) that

$$T^\mu w = \mu T_0 w + \sum_{n=1}^{N} u_{Nn}(\mu)\, T_n w \ .$$

Note that T_0 and the T_n are parameter-independent.

4.3.2 Error Bound

We assume that we are given a $\hat\beta(\mu)$ such that

$$\beta(\mu) \ge \hat\beta(\mu) \ge (1 - \tau)\, \varepsilon_s, \quad \forall\, \mu \in \mathcal{D} \ , \tag{5.116}$$

where $\tau \in]0, 1[$. As before, $\|\mathcal{Y} r(\,\cdot\,; \mu)\|_Y$ is the dual norm of the residual,

$$r(v; \mu) \equiv f(v) - a(u_N(\mu), v; \mu), \quad \forall\, v \in Y \ .$$

We then define, for $\|\mathcal{Y} r(\,\cdot\,; \mu)\|_Y \le \frac{\hat\beta^2(\mu)}{2\rho^2}$,

$$\Delta_N(\mu) \ \equiv \ [\hat\beta(\mu) - (\hat\beta^2(\mu) - 2\rho^2 \|\mathcal{Y} r(\,\cdot\,; \mu)\|_Y)^{1/2}]/\rho^2 \tag{5.117}$$

$$\Upsilon_N(\mu) \ \equiv \ [\hat\beta(\mu) + (\hat\beta^2(\mu) - 2\rho^2 \|\mathcal{Y} r(\,\cdot\,; \mu)\|_Y)^{1/2}]/\rho^2 \ . \tag{5.118}$$

We note that, as $\|\mathcal{Y} r(\,\cdot\,; \mu)\|_Y \to 0$,

$$\Delta_N(\mu) \sim \frac{\|\mathcal{Y} r(\,\cdot\,; \mu)\|_Y}{\hat\beta(\mu)}$$

and

$$\Upsilon_N(\mu) \sim \frac{2\hat\beta(\mu)}{\rho^2} \ ;$$

thus, $\Delta_N(\mu) \to 0$ but $\Upsilon_N(\mu) \to$ Const as the residual vanishes.

We can then state

Proposition 8. *Given $\mu \in \mathcal{D}^\mu$, for N sufficiently large such that*

$$\|\mathcal{Y}r(\,\cdot\,;\mu)\|_Y \le \frac{\hat\beta^2(\mu)}{2\rho^2}\ , \tag{5.119}$$

either

$$\|e(\mu)\|_{H^1(\Omega)} \le \Delta_N(\mu)\ , \tag{5.120}$$

or

$$\|e(\mu)\|_{H^1(\Omega)} \ge \Upsilon_N(\mu)\ , \tag{5.121}$$

where $\Delta_N(\mu)$ and $\Upsilon_N(\mu)$ are given by (5.117) and (5.118), respectively.

Proof It is a simple matter to show that $e(\mu) = u(\mu) - u_N(\mu)$ satisfies

$$\begin{aligned} d(e(\mu), v; u_N(\mu); \mu) = r(v;\mu) - a^{\rm NL}(e(\mu), e(\mu), v), \\ \forall\, v \in Y\ . \end{aligned} \tag{5.122}$$

Since from (5.101) and (5.108)

$$\begin{aligned} \left|a^{\rm NL}(w,w,v)\right| &\le \tfrac{1}{2}\|w\|^2_{L^4(\Omega)}\,\|v\|_Y \\ &\le \tfrac{1}{2}\,\rho^2\,\|w\|^2_Y\,\|v\|_Y\ , \end{aligned}$$

it follows from (5.122) (with $v = T^\mu e(\mu)$), (5.103), (5.105), and (5.116) that

$$\tfrac{1}{2}\rho^2\|e(\mu)\|^2_Y - \hat\beta(\mu)\|e(\mu)\|_Y + \|\mathcal{Y}r(\,\cdot\,;\mu)\|_Y \ge 0\ . \tag{5.123}$$

The desired result directly follows from solution of this quadratic equation for $\|e(\mu)\|_Y$. □

We note that an alternative proof — which directly places a restriction on $\|e(\mu)\|_Y$ that is subsequently self-consistently determined from the strength of the nonlinearity — is applicable to much more general nonlinearities. However, in the quadratic case, the proof above is simpler and slightly sharper.

We do not include here a uniform upper bound for the effectivity, however, it is clear from (5.123) that, as $\|\mathcal{Y}r(\,\cdot\,;\mu)\|_Y \to 0$, $\eta_N(\mu) \le \gamma(\mu)/(1-\tau)\varepsilon_s$ (recall that both $\gamma(\mu)$ and $\varepsilon_s(\mu)$ may depend on $u_N(\mu)$).

We now turn to an interpretation of the "choice" (5.120), (5.121). It is clear that, given W_N, (5.111) (and hence (5.112)), all evidence would suggest that, as $\|\mathcal{Y}r(\,\cdot\,;\mu)\|_Y \to 0$, $u_N(\mu)$ should well approximate $u^I(\mu)$; thus (5.120) — note $\Delta_N(\mu) \to 0$ as $N \to \infty$ — is the most obvious choice for $\|u^I(\mu) - u_N(\mu)\|_Y$. However, equally clearly, if a second branch, $u^{II}(\mu)$, exists, there is no reason that $u_N(\mu)$ should — in fact, there is every reason that $u_N(\mu)$ should not — well approximate $u^{II}(\mu)$; thus, for this (possible) second branch, (5.121) — note $\Upsilon_N(\mu) \to$ Const as $N \to \infty$ — is the most obvious choice. In short, the error bound "sees" only the residual, which in turn "sees" only the *branch-independent* projection of $u^I(\mu)$ (or $u^{II}(\mu)$), $f(v)$. Thus, absent other *a priori* information, the $\Upsilon_N(\mu)$ option is a nonlinear

necessity — a reflection of the potential existence of distinct multiple solutions.[†] Consistent with these arguments, we note that if $d(\cdot\,,\cdot\,;u_N(\mu);\mu)$ is *coercive* — certainly the case for sufficiently large μ — then it follows directly from (5.122), since $a^{\rm NL}(e(\mu),e(\mu);e(\mu))=0$, that we may obtain an *unconditional* bound on the error in $u_N(\mu)$ relative to the perforce *single* branch, $u^{(I)}(\mu)$.

However, there is a dark side: we can not rigorously preclude the possibility that $\|u^I(\mu)-u_N(\mu)\|_Y \geq \Upsilon_N(\mu)$. Although this is *extremely* unlikely as $N\to\infty$ — since $\Upsilon_N(\mu)\to \text{Const}$ as $N\to\infty$ — it can not be unambiguously ruled out for any fixed N. Clearly, in actual practice, the relative (and absolute) magnitude of $\Upsilon_N(\mu)$ will directly affect our comfort level in choosing (5.120). We discuss this again in the context of our numerical results.

4.3.2 Inf-Sup Lower Bound Construction

We assume that we are given a set of J parameter points, $\mathcal{L}_J \equiv \{\overline{\mu}^1\in\mathcal{D}^\mu,\ldots,\overline{\mu}^J\in\mathcal{D}^\mu\}$, and associated segments $\mathcal{R}^{\overline{\mu}^j,\tau}$, $1\leq j\leq J$, where

$$\mathcal{R}^{\overline{\mu},\tau} \equiv \{\mu\in\mathcal{D}^\mu \mid \mathcal{B}^{\overline{\mu}}(\mu)\leq \tau\,\beta(\overline{\mu})\}\ , \tag{5.124}$$

and

$$\mathcal{B}^{\overline{\mu}}(\mu) = |\mu-\overline{\mu}| + \rho\|u_N(\mu)-u_N(\overline{\mu})\|_{L^4(\Omega)}\ ; \tag{5.125}$$

we further assume that

$$\bigcup_{j=1}^{J}\mathcal{R}^{\overline{\mu}^j,\tau} = \mathcal{D}^\mu\ . \tag{5.126}$$

We then define $\mathcal{J}:\mathcal{D}^\mu\to\{1,\ldots,J\}$ such that, for a given μ, $\mathcal{R}^{\overline{\mu}^{\mathcal{J}(\mu)},\tau}$ is that segment (or a segment) which contains μ.

Our lower bound is then: Given $\mu\in\mathcal{D}^\mu$,

$$\hat{\beta}(\mu) \equiv \beta(\overline{\mu}^{\mathcal{J}(\mu)}) - \mathcal{B}^{\overline{\mu}^{\mathcal{J}(\mu)}}(\mu)\ , \tag{5.127}$$

for $\mathcal{B}^{\overline{\mu}}(\mu)$ defined in (5.125).

We can now state

Proposition 9. *The construction $\hat{\beta}(\mu)$ of (5.127) satisfies the inequality (5.116).*

Proof (Sketch) The proof is almost identical to the proof of Proposition 2 for the Helmholtz inf-sup lower bound construction. We need only replace the relation (5.44) and (5.45) with

$$\beta^2(\mu) \geq \inf_{w\in Y}\left\{\left[\sigma(w;\overline{\mu}) - \frac{\left\|(\mu-\overline{\mu})T_0 w + \sum_{n=1}^{N}(u_{Nn}(\mu)-u_{Nn}(\overline{\mu}))T_n w\right\|_Y}{\|w\|_Y}\right]^2\right\}. \tag{5.128}$$

[†] Note if we include both (say, in the case of two branches) branches, $u^I(\mu^n)$, $u^{II}(\mu^n)$, in W_N, then we will typically obtain good reduced-basis approximations to both branches — $u_N^I(\mu)$, $u_N^{II}(\mu)$. In this case our single bound $\Delta_N(\mu)$ ($\to 0$) would apply to both $\|u^I(\mu)-u_N^I(\mu)\|_Y$ and $\|u^{II}(\mu)-u_N^{II}(\mu)\|_Y$, and $\Upsilon_N(\mu)$ ($\to \text{Const}$) would apply to both $\|u^I(\mu)-u_N^{II}(\mu)\|_Y$ and $\|u^{II}(\mu)-u_N^I(\mu)\|_Y$.

and

$$\left\| (\mu - \overline{\mu}) T_0 w + \sum_{n=1}^{N} (u_{Nn}(\mu) - u_{Nn}(\overline{\mu})) T_n w \right\|_Y \leq \left(|\mu - \overline{\mu}| + \rho \|u_N(\mu) - u_N(\overline{\mu})\|_{L^4(\Omega)} \right) \|w\|_Y \ , \tag{5.129}$$

respectively. The continuity result (5.129) follows from (5.113)–(5.115), from

$$|a_0(w,v)| \leq \|w\|_Y \|v\|_Y, \quad \forall\, w, v \in Y \ ,$$

(recall $\|\cdot\|_Y \equiv \|\cdot\|_{H^1(\Omega)}$), and from

$$\begin{aligned} \left| \sum_{n=1}^{N} (u_{Nn}(\mu) - u_{Nn}(\overline{\mu})) a_n(w,v) \right| &= \left| \int_0^1 (u_N(\mu) - u_N(\overline{\mu})) w v_x \right| \\ &\leq \|u_N(\mu) - u_N(\overline{\mu})\|_{L^4(\Omega)} \rho \|w\|_Y \|v\|_Y \ . \end{aligned}$$

To derive this last expression we invoke (5.112), (5.115), Cauchy-Schwarz, and (5.108). It thus follows from (5.105), (5.106), (5.128), and (5.129) that

$$\beta^2(\mu) \geq \inf_{t \in [\beta(\overline{\mu}), \gamma(\overline{\mu})]} \left\{ \left[t - (|\mu - \overline{\mu}| + \rho \|u_N(\mu) - u_N(\overline{\mu})\|_{L^4(\Omega)}) \right]^2 \right\} \ . \tag{5.130}$$

We now choose, for any given μ, $\overline{\mu} = \overline{\mu}^{\mathcal{J}(\mu)}$. We then note that, from (5.124) and (5.125), the infimizer of (5.130) is $\beta(\overline{\mu}^{\mathcal{J}(\mu)})$; it thus follows that

$$\beta(\mu) \geq \beta(\overline{\mu}^{\mathcal{J}(\mu)}) - \mathcal{B}^{\overline{\mu}^{\mathcal{J}(\mu)}}(\mu) \ . \tag{5.131}$$

The desired result, (5.116), immediately follows from (5.131), (5.127), (5.124), and $\beta(\mu) \geq \varepsilon_s$. □

4.3.3 Offline/Online Computational Procedure

All the elements of the offline/online procedure for the construction of Burgers *a posteriori* error bounds have already been introduced in the context of the Helmholtz and cubically nonlinear Poisson problems. We thus restrict ourselves to a few brief comments.

First, in forming the segments $\mathcal{R}^{\overline{\mu}^j,\tau}$, $1 \leq j \leq J$, we do not need to (and could not...) exhaustively verify that for all $\mu \in \mathcal{R}^{\overline{\mu}^j,\tau}$, $|\mu - \overline{\mu}^j| + \rho \|u_N(\mu) - u_N(\overline{\mu}^j)\|_{L^4(\Omega)} \leq \tau \beta(\overline{\mu}^j)$. Rather, we can make plausible continuity assumptions to construct these intervals, and then verify this condition, *a posteriori*, online. Second, the computationally most intensive online calculation (for large N) is precisely this $\|u_N(\mu) - u_N(\overline{\mu}^j)\|_{L^4(\Omega)}$ evaluation; however, by invoking the symmetry summation techniques developed in Section 3.3.3, we can reduce the

relevant operation count to $\frac{1}{24}N^4$ — typically not dominant for the small N realized by our adaptive sampling process. Third, for Burgers equation in $\mathbb{R}^1$, our reduced-basis approach is not competitive (even as regards marginal cost) with standard techniques, that is, direct computation of $s^{\mathcal{N}}(\mu)$. However, our complexity estimates also apply to incompressible Navier-Stokes in $\mathbb{R}^{2,3}$, in which case we effect very considerable savings relative to finite element calculation of $s^{\mathcal{N}}(\mu)$.

5.4.4 4.4 Numerical Results

Our model problem is given by (5.99)–(5.101) and (5.110); we need only specify $f(v) = \int_0^1 v$, and $\mathcal{D}^\mu = [\mu^{\min} = .01,\ \mu^{\max} = 10]$. All results presented are for the *adaptive* sampling procedure.

To begin, we present in Figure 5 $\beta(\mu)$ and $\hat{\beta}(\mu)$ as a function of μ; we also indicate, on the $\log(\mu)$-axis, the segments $\mathcal{R}^{\overline{\mu}^j,\tau=3/4}$, $1 \le j \le J = 55$. There is clearly some deterioration in the length of our segments as $\overline{\mu}^j$ decreases — it would appear that J increases more rapidly than $\ln(\mu_{\max}/\mu_{\min})$. For problems with $P > 1$, this growth would probably not be tolerable. It is possible that deflation techniques — similiar to those introduced in the context of the Helmholtz problem in Section 2.4.3 — could considerably increase the effective inf-sup parameter, and hence considerably decrease J.

The first test case we consider is $\mu = 1.0$. We present in Table 5.6 $\|e(\mu)\|_Y/\|u(\mu)\|_Y$, $\Delta_N(\mu)/\|u(\mu)\|_Y$, $\eta_N(\mu)$, and $\Upsilon_N(\mu)/\|u(\mu)\|_Y$ as a function of N. We observe that the reduced-basis approximation converges very rapidly; that at least in this particular case, the "good" choice, (5.120), obtains — $\|e(\mu)\|_Y \le \Delta_N(\mu)$, $\forall\ N \in \mathbb{N}$; that the effectivities are, as desired, quite close to unity; and that $\Upsilon_N(\mu)$ is (constant and) very large. From the latter we could very plausibly select (5.120) over (5.121) even if — as will be the case in practice — we do not have access to the true error, $\|e(\mu)\|_Y$.

We now turn to $\mu = 0.01$. We present in Table 5.7 $\|e(\mu)\|_Y/\|u(\mu)\|_Y$, $\Delta_N(\mu)/\|u(\mu)\|_Y$, $\eta_N(\mu)$, and $\Upsilon_N(\mu)/\|u(\mu)\|_Y$ as a function of N. We observe two difficulties not encountered for $\mu = 1.0$. First, although the reduced-basis approximation in fact converges rather quickly, we can only begin to make an *a posteriori* error statement for $N \ge 11$ — for $N < 11$, condition (5.119) is not satisfied. (Recall that these results are for the *adaptive* sampling procedure; in the case of a random sample, condition (5.119) is not satisfied for *all* N.) Second, although the "good" choice, (5.120), in fact obtains — $\|e(\mu)\|_Y \le \Delta_N(\mu)$ with effectivities $O(5\text{–}10)$ for $N \ge 11$ — the value of $\Upsilon_N(\mu)$ is not as large as desirable; it would thus be difficult in practice (when $\|e(\mu)\|_Y$ is *not* known) to unambiguously rule out the "bad" choice (5.121). The origin of both these difficulties is the small value of $\beta(\mu = 0.01)$ (and $\beta(\mu)$, μ small, generally). We are hopeful that deflation ideas similar to those successful in the Helmholtz case (see Section 2.4.3) will also prove beneficial here — increasing the effective $\beta(\mu)$, and thereby increasing both our threshold in (5.119) and the value of $\Upsilon_N(\mu)$ in (5.118), (5.121).

5.5 Acknowledgements

We would like to acknowledge our longstanding collaborations with Professor Yvon Maday of University of Paris VI and Professor Jaime Peraire of MIT. We would also like thank

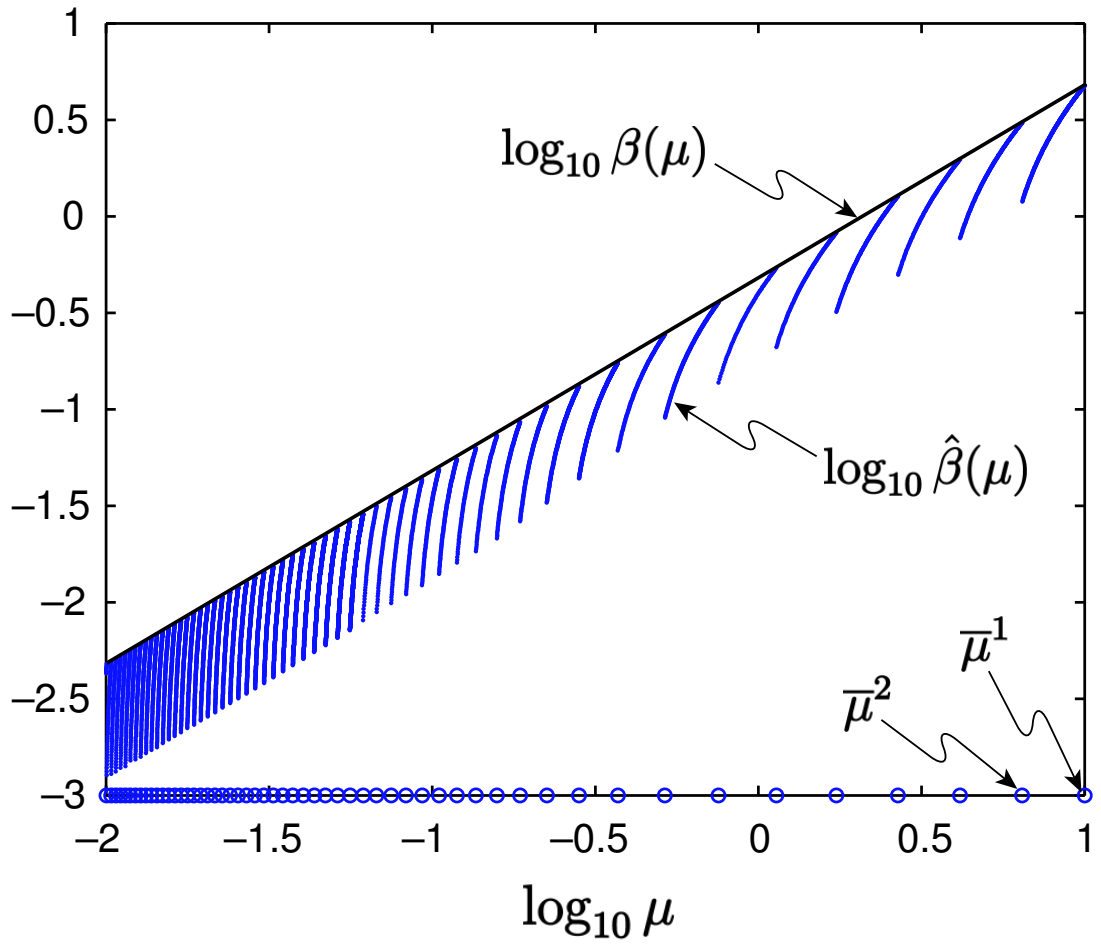


Figure 5.5: The inf-sup parameter $\beta(\mu)$, lower bound $\hat{\beta}(\mu)$, and segments $\mathcal{R}^{\overline{\mu}^j,\tau=3/4} = [\overline{\mu}^j, \overline{\mu}^{j+1}]$ for the model Burgers problem.

N	$\frac{\|e(\mu)\|_Y}{\|u(\mu)\|_Y}$	$\frac{\Delta_N(\mu)}{\|u(\mu)\|_Y}$	$\eta_N(\mu)$	$\frac{\Upsilon_N(\mu)}{\|u(\mu)\|_Y}$
3	1.9×10^{-2}	2.2×10^{-2}	1.17	19.4
6	4.7×10^{-3}	5.6×10^{-3}	1.17	19.4
9	2.9×10^{-3}	3.4×10^{-3}	1.18	19.4
12	2.6×10^{-4}	3.0×10^{-4}	1.18	19.4
15	2.9×10^{-5}	3.4×10^{-5}	1.18	19.4

Table 5.6: Error, error bounds, and effectivity as a function of N for the Burgers problem with $\mu = 1.0$.

N	$\frac{\|e(\mu)\|_Y}{\|u(\mu)\|_Y}$	$\frac{\Delta_N(\mu)}{\|u(\mu)\|_Y}$	$\eta_N(\mu)$	$\frac{\Upsilon_N(\mu)}{\|u(\mu)\|_Y}$
11	3.3×10^{-6}	1.3×10^{-5}	3.9	1.6×10^{-4}
12	1.8×10^{-6}	5.4×10^{-6}	3.1	1.6×10^{-4}
13	7.1×10^{-9}	1.5×10^{-8}	2.0	1.7×10^{-4}
14	7.1×10^{-9}	1.3×10^{-8}	1.9	1.7×10^{-4}
15	7.2×10^{-9}	1.8×10^{-8}	2.5	1.7×10^{-4}

Table 5.7: Error, error bounds, and effectivity as a function of N for the Burgers problem with $\mu = .01$.

Mr. Nguyen Ngoc Cuong of the National University of Singapore for helpful discussions. This work was supported by DARPA and AFOSR under Grant F49620-01-1-0458 and by the Singapore-MIT Alliance.

Chapter 6

Reduced-Basis Approximation of the Viscous Burgers Equation: Rigorous A Posteriori Error Bounds

Authors: K. Veroy, C. Prud'homme, A.T. Patera.

Chapter 7

Reduced-Basis Output Bounds for Approximately Parametrized Elliptic Coercive Partial Differential Equations

Authors: C. Prud'homme, A.T. Patera.

7.1 Introduction

The optimization, control, and characterization of an engineering component or system requires the prediction of certain "quantities of interest," or performance metrics, which we shall denote *outputs* — for example deflections, maximum stresses, maximum temperatures, heat transfer rates, flowrates, or lifts and drags. These outputs are typically expressed as functionals of field variables associated with a parametrized partial differential equation which describes the physical behavior of the component or system. The parameters, which we shall denote *inputs*, serve to identify a particular "configuration" of the component: these inputs may represent design or decision variables, such as geometry — for example, in optimization studies; control variables, such as actuator power — for example, in real-time applications; or characterization variables, such as physical properties — for example, in inverse problems. We thus arrive at an implicit *input-output* relationship, evaluation of which demands solution of the underlying partial differential equation.

Our goal is the development of computational methods that permit *rapid* and *reliable* evaluation of this partial-differential-equation-induced input-output relationship *in the limit of many queries* — that is, in the design, optimization, control, and characterization contexts. The "many queries" limit has certainly received considerable attention: from "fast loads" or multiple right-hand side notions (e.g., [21, 25]) to matrix perturbation theories (e.g., [3, 85]) to continuation methods (e.g., [4, 71]). Our particular approach is based on the reduced-basis method, first introduced in the late 1970s for nonlinear structural analysis [5, 51], and subsequently developed more broadly in the 1980s and 1990s [13, 14, 26, 55, 58, 70]. The reduced-basis method recognizes that the field variable is not, in fact, some arbitrary member of

the infinite-dimensional solution space associated with the partial differential equation; rather, it resides, or "evolves," on a much lower dimensional manifold induced by the parametric dependence.

The reduced-basis approach as earlier articulated is local in parameter space in both practice and theory. To wit, Lagrangian or Taylor approximation spaces for the low–dimensional manifold are typically defined relative to a particular parameter point; and the associated *a priori* convergence theory relies on asymptotic arguments in sufficiently small neighborhoods [26]. As a result, the computational improvements — relative to conventional (say) finite-element approximation — are often quite modest. Our work [37, 44, 46, 62, 82] differs from these earlier efforts in several important ways: first, we develop (in some cases, provably) *global* approximation spaces; second, we introduce rigorous *a posteriori error estimators*; and third, we exploit *off–line/on–line* computational decompositions (see [13] for an earlier application of this strategy within the reduced-basis context). These three ingredients allow us, for the restricted but important class of (approximately) "parameter-affine" problems, to reliably decouple the generation and projection stages of reduced-basis approximation, thereby effecting computational economies of several orders of magnitude.

In this paper we address the situation in which our parametrized mathematical model is not exact or "complete": we permit error or imprecision in the data that define our partial differential equation. This error may be introduced, for example, by imperfect specification, measurement, calculation, or parametric expansion of a coefficient function. Our goal is to develop both accurate predictions for the outputs of interest *and* associated rigorous *a posteriori* error bounds; and we require that the latter incorporate *both* numerical discretization *and* "model truncation" effects. The method we propose can be viewed as a "many parameters of small variation" extension to our earlier reduced-basis output bound approaches. (For an alternative treatment — based on additive Schwarz domain-decomposition bound conditioners — of many parameters, in particular "many parameters of small *support*," see [74]; this method is relevant, for example, to non-affine boundary shape variations.)

In Section 7.2 we present the general problem formulation. In Section 7.3 we develop our reduced-basis approximation, and prove associated *a priori* error estimates. In Section 7.4 we describe our *a posteriori* error estimators, and demonstrate the necessary effectivity results. And finally, in Section 7.5 , we present detailed numerical results for a (reasonably) realistic application.

7.2 Problem Formulation

We consider a suitably regular domain $\Omega \subset \mathbb{R}^d$, $d = 1,\ 2$, or 3, with boundary $\partial\Omega$, and associated function space X such that $H_0^1(\Omega) \subset X \subset H^1(\Omega)$, where $H^1(\Omega) = \{v \in L^2(\Omega),\ \nabla v \in (L^2(\Omega))^d\}$, $H_0^1(\Omega) = \{v \in H^1(\Omega) \mid v|_{\partial\Omega} = 0\}$, and $L^2(\Omega)$ is the space of square-integrable functions over Ω. The inner product and norm associated with X are given by $(\cdot\,,\,\cdot)_X$ and $\|\cdot\|_X = (\cdot\,,\,\cdot)_X^{1/2}$, respectively. The dual space to X is denoted X', with norm $\|\cdot\|_{X'}$; and the associated duality pairing is expressed as ${}_{X'}\langle\,\cdot\,,\,\cdot\,\rangle_X = \langle\,\cdot\,,\,\cdot\,\rangle$. We also define a closed, bounded parameter set $\mathcal{D}^\mu \in \mathbb{R}^P$, a particular point in which will be denoted μ. Note that Ω is a reference domain, and hence does *not* depend on the parameter.

We next introduce our distributional (second-order partial differential) operator $\mathcal{A}(\mu)\colon X \rightarrow$

X', which we shall assume is uniformly continuous and coercive over $\mathcal{D}^\mu$ (with coercivity constant $C_\mathcal{A}$ — $\langle \mathcal{A}(\mu)v, v\rangle \geq C_\mathcal{A} \|v\|_X^2$, $\forall\, v \in X$, $\forall\, \mu \in \mathcal{D}^\mu$); and we further define two linear, bounded functionals, $F \in X'$ and $L \in X'$. Our exact problem statement is then: Given $\mu \in \mathcal{D}^\mu$, find $s(\mu) = \langle L, u(\mu)\rangle$, where $u(\mu) \in X$ satisfies

$$\langle \mathcal{A}(\mu)\, u(\mu), v\rangle = \langle F, v\rangle, \quad \forall\, v \in X\ . \tag{7.1}$$

We may on occasion further abbreviate (7.1) as $\mathcal{A}(\mu)\ u(\mu) = F$, to be interpreted in the distributional sense. In the language of the introduction, $\mathcal{A}(\mu)$ is our partial differential operator, μ is our parameter, F is our data (which for simplicity we assume does not depend on μ), $u(\mu)$ is our field variable, L is our output functional, and $s(\mu)$ is our output of interest.

As indicated in the introduction, it will often be the case that we have access only to an approximation of $\mathcal{A}, \overline{\mathcal{A}}$: more precisely, $\mathcal{A}(\mu) = \overline{\mathcal{A}}(\mu) + \widehat{\mathcal{A}}(\mu)$, where $\widehat{\mathcal{A}}(\mu)$ is in some sense small. (We may also consider error in the data, $F = \overline{F} + \widehat{F}$; see [78].) We shall assume that $\overline{\mathcal{A}}$ remains coercive (with, for simplicity, coercivity constant $C_{\overline{\mathcal{A}}} = C_\mathcal{A}$). Our "model-truncated" problem is thus: Given $\mu \in \mathcal{D}^\mu$, find $\overline{s}(\mu) = \langle L, \overline{u}(\mu)\rangle$, where $\overline{u}(\mu) \in X$ satisfies

$$\langle \overline{\mathcal{A}}(\mu)\, \overline{u}(\mu), v\rangle = \langle F, v\rangle, \quad \forall\, v \in X\ . \tag{7.2}$$

In Sections 7.3 and 7.4 we shall quantify $s(\mu) - \overline{s}(\mu)$ in terms of the size, though not necessarily the details, of $\widehat{\mathcal{A}}(\mu)$; but we must first provide more information on the structure of $\widehat{\mathcal{A}}(\mu)$. (For a concrete instantiation of the quantities and assumptions introduced below, the reader may refer to Section 7.5.1.)

We shall first make the assumption of affine parameter dependence. In particular we shall suppose that

$$\overline{\mathcal{A}}(\mu) = \sum_{q=1}^{\overline{Q}} \alpha_q(\mu)\, A_q\ , \tag{7.3}$$

and

$$\widehat{\mathcal{A}}(\mu) = \sum_{q=1}^{\widehat{Q}} \beta_q(\mu)\, B_q\ ; \tag{7.4}$$

here $\alpha\colon \mathcal{D}^\mu \to \mathbb{R}^{\overline{Q}}$, $\beta\colon \mathcal{D}^\mu \to \mathbb{R}^{\widehat{Q}}$ are parameter *dependent* functions, and $A_q\colon X \to X'$, $1 \leq q \leq \overline{Q}$, $B_q\colon X \to X'$, $1 \leq q \leq \widehat{Q}$, are parameter-*independent* operators.[†] We shall further suppose that $\overline{Q}$ is relatively small, for example $O(10\text{–}100)$; but that $\widehat{Q}$ may be quite

[†] In many cases, the B_q *will* depend (affinely) on μ,

$$B_q(\mu) = \sum_{q'=1}^{\widetilde{Q}} g_{qq'}(\mu)\, B_{qq'},\ 1 \leq q \leq \widehat{Q}\ ,$$

where the $g_{qq'}\colon \mathcal{D}^\mu \to \mathbb{R}$, $1 \leq q \leq \widehat{Q}$, $1 \leq q' \leq \widetilde{Q}$, are parameter-*dependent* functions, and the $B_{qq'}\colon X \to X'$, $1 \leq q \leq \widehat{Q}$, $1 \leq q' \leq \widetilde{Q}$, are parameter-*independent* operators. Our method extends readily to this case; however, for simplicity of exposition, we shall assume here that the B_q do *not* depend on μ.

large — indeed $\widehat{Q}$ will in many cases tend to infinity. Note that, in contrast to our earlier work, our assumption that $\overline{Q}$ is small is now more general — in the sense that the "remainder" may now be absorbed in $\widehat{\mathcal{A}}(\mu)$.

In actual practice, we replace (7.1) and (7.2) with a "truth approximation" defined over a finite element space $X^{\mathcal{N}}$ of dimension $\mathcal{N}$; this also ensures that $\widehat{Q}$ may be presumed finite without loss of generality. We assume that $X^{\mathcal{N}}$ is sufficiently rich such that $u^{\mathcal{N}}, s^{\mathcal{N}}$ and $\overline{u}^{\mathcal{N}}, \overline{s}^{\mathcal{N}}$ are indistinguishable from u, s and $\overline{u}, \overline{s}$, respectively; we thus anticipate that, in general, $\mathcal{N}$ (and potentially $\widehat{Q}$) will be very large. For the purposes of this paper we shall effectively equate X and $X^{\mathcal{N}}$ (and hence the "exact" and "truth approximation" problem statements) except as regards two points. First, in our computational estimates, we shall verify that the "on-line" computational complexity is independent of $\mathcal{N}$ (and $\widehat{Q}$). Second, in our analyses, we shall verify that our numerical results — for example, our error bounds — are stable and convergent as $\mathcal{N}$ (and potentially $\widehat{Q}$) tends to infinity.

In preparation for the latter, we shall require certain hypotheses on $\beta_q(\mu)$ and B_q in the case in which (*i*) these quantities depend on $\widehat{Q}$, and (*ii*) $\widehat{Q}$ tends to infinity. (Note we assume that, in all cases, $\overline{Q}$ is fixed — or, in any event, independent of $\mathcal{N}$.) By way of preliminaries, we first introduce $\widehat{Q}$ suitably regular open subdomains, $D_q \subset \Omega$, $1 \le q \le \widehat{Q}$. We assume that a given subdomain D_q intersects only finitely many other subdomains $D_{q'}$ (as $\widehat{Q} \to \infty$). We next define parameter-independent inner products and norms over D_q, $((\,\cdot\,,\,\cdot\,))_q$ and $|||\,\cdot\,|||_q = ((\,\cdot\,,\,\cdot\,))_q^{1/2}$, respectively. We assume that $|||\,\cdot\,|||_q$ is *uniformly* equivalent to $\|\,\cdot\,\|_{H^1(D_q)}$ for all functions in $H^1(D_q)$ in the sense that the relevant (ratio) constants may be bounded *independent* of μ and $\widehat{Q}$. It then follows from our assumptions that there exists a positive finite constant $\hat{\rho}_\Sigma$ — *independent* of μ and $\widehat{Q}$ — such that

$$\sum_{q=1}^{\widehat{Q}} |||\, v|_{D_q} \,|||_q^2 < (\hat{\rho}_\Sigma)^2 \, \|v\|_X^2, \quad \forall\, v \in X \ . \tag{7.5}$$

Note that, if $\|\,\cdot\,\|_X = \|\,\cdot\,\|_{H^1(\Omega)}$ and $|||\,\cdot\,|||_q = \|\,\cdot\,\|_{H^1(D_q)}$, then $\hat{\rho}_\Sigma$ is bounded from above by $C_{\rm mult}$, the maximum number of subdomains D_q in which any point in Ω may reside.

We now define

$$\beta_{\widehat{Q}}^{\max}(\mu) = \max_{q \in \{1,\ldots,\widehat{Q}\}} |\beta_q(\mu)| \ , \tag{7.6}$$

and require that

$$\beta_{\widehat{Q}}^{\max}(\mu) \le \hat{\beta}^{\max} \tag{7.7}$$

for $\hat{\beta}^{\max} \in \mathbb{R}_+$ independent of μ and $\widehat{Q}$ — and preferably *small*. We shall presume that, in general, we have access only (and directly) to $\beta_{\widehat{Q}}^{\max}(\mu)$ — and that we need not appeal to the individual $\beta_q(\mu)$. We next define $\gamma_q \colon X \to \mathbb{R}$, $1 \le q \le \widehat{Q}$, as

$$\gamma_q(w) \equiv \sup_{v \in X} \frac{\langle B_q w, v\rangle}{|||\, v|_{D_q} \,|||_q} \ , \tag{7.8}$$

and require that

$$\sup_{w \in X} \frac{\left(\sum_{q=1}^{\widehat{Q}} \gamma_q^2(w) \right)^{1/2}}{\|w\|_X} \leq \widehat{\Gamma} \ , \tag{7.9}$$

for $\widehat{\Gamma} \in \mathbb{R}$ *independent* of μ and $\widehat{Q}$. Note that our condition (7.9) implies, and hence we may assume, that $\langle B_q w, v \rangle = \langle B_q w, v|_{D_q} \rangle$, $\forall\, v \in X$: we may (often) interpret $\beta_q(\mu)$ as the error in our data over a small region D_q; and $\beta_q(\mu)\langle B_q w, v\rangle$ as the contribution of this error to our remainder term, $\widehat{\mathcal{A}}(\mu)$.

7.3 Reduced-Basis Approximation

7.3.1 Formulation

We first introduce a sample $S_N = \{\mu_1 \in \mathcal{D}^\mu, \ldots, \mu_N \in \mathcal{D}^\mu\}$. We then define $X_N = \text{span}\ \{\zeta_n \equiv \overline{u}(\mu_n),\ n = 1, \ldots, N\}$, where $\overline{u}(\mu_n)$ satisfies (7.2) for $\mu = \mu_n$ (Option I); or $X_N = \text{span}\ \{\zeta_n \equiv u(\mu_n),\ n = 1, \ldots, N\}$, where $u(\mu_n)$ satisfies (7.1) for $\mu = \mu_n$ (Option II). Our reduced-basis approximation for the model-truncated problem is then: Given $\mu \in \mathcal{D}^\mu$, find $\overline{s}_N(\mu) = \langle L, \overline{u}_N(\mu)\rangle$, where $\overline{u}_N(\mu) \in X_N$ satisfies

$$\langle \overline{\mathcal{A}}(\mu)\, \overline{u}_N(\mu), v\rangle = \langle F, v\rangle, \quad \forall v \in X_N \ . \tag{7.10}$$

Thus $\overline{u}_N(\mu)$ is the standard Galerkin approximation of $\overline{u}(\mu)$ over the space $X_N \subset X$.

We can also express this approximation in terms of our particular basis: Given $\mu \in \mathcal{D}^\mu$, find $\overline{s}_N(\mu) = \underline{L}_N^T \overline{\underline{u}}_N(\mu)$, where $\overline{\underline{u}}_N(\mu) \in \mathbb{R}^N$ satisfies

$$\overline{\underline{\mathcal{A}}}_N(\mu)\, \overline{\underline{u}}_N(\mu) = \overline{\underline{F}}_N \ . \tag{7.11}$$

Here, $\overline{L}_{N\,i} = \langle L, \zeta_i\rangle$, $\overline{F}_{N\,i} = \langle F, \zeta_i\rangle$, $1 \leq i \leq N$; and $\overline{\mathcal{A}}_{N\,i,j} = \langle \overline{\mathcal{A}}(\mu)\, \zeta_j, \zeta_i\rangle$, $1 \leq i, j \leq N$. Note that

$$\overline{u}_N(\mu) = \sum_{n=1}^{N} \overline{u}_{Nn}(\mu)\, \zeta_n = \overline{\underline{u}}_N(\mu)^T\, \underline{\zeta} \ , \tag{7.12}$$

where $\overline{\underline{u}}_N = (\overline{u}_{N1}\ \cdots\ \overline{u}_{NN})^T$ and $\underline{\zeta} = (\zeta_1\ \cdots\ \zeta_N)^T$; here ^{T} refers to algebraic transpose.

As noted, we in fact have two options for the construction of our reduced-basis space. Option I, $X_N = \text{span}\ \{\overline{u}(\mu_n),\ n = 1, \ldots, N\}$, does not require us to incorporate the details of $\widehat{\mathcal{A}}(\mu)$ — in particular, the $\beta_q(\mu)$, $1 \leq q \leq \widehat{Q}$ — in our formation of X_N; but does require us to *reform* X_N each time we improve $\overline{\mathcal{A}}(\mu)$ (so as to reduce $\widehat{\mathcal{A}}(\mu)$). In contrast, Option II, $X_N = \text{span}\ \{u(\mu_n),\ n = 1, \ldots, N\}$, does require us to incorporate the details of $\widehat{\mathcal{A}}(\mu)$ — the $\beta_q(\mu)$, $1 \leq q \leq \widehat{Q}$ — in our formation of X_N; but does not (necessarily) require us to reform X_N each time we improve $\overline{\mathcal{A}}(\mu)$ (so as to reduce $\widehat{\mathcal{A}}(\mu)$). Clearly, the best choice will be problem-dependent.

7.3.2 A Priori Theory

We wish to bound $s(\mu) - \overline{s}_N(\mu)$. We do so in

Proposition 10. *For $\beta_{\widehat{Q}}^{\max}(\mu)$ and $\gamma_q(w)$ satisfying (7.7) and (7.9), respectively,*

$$|s(\mu) - \overline{s}_N(\mu)| \leq \|L\|_{X'} \Big(C_1 \inf_{v \in X_N} \|\overline{u}(\mu) - v\|_X + C_2 \, \hat{\rho}_\Sigma \, \hat{\beta}^{\max} \widehat{\Gamma} \Big), \qquad \forall \, \mu \in \mathcal{D}^\mu \; ; \tag{7.13}$$

here C_1 and C_2 depend only on coercivity and continuity constants and on the data F.

Proof. We first introduce $e(\mu) = u(\mu) - \overline{u}_N(\mu)$, and further define $\overline{e}(\mu) = \overline{u}(\mu) - \overline{u}_N(\mu)$, $\hat{e}(\mu) = u(\mu) - \overline{u}(\mu)$, such that $e(\mu) = \overline{e}(\mu) + \hat{e}(\mu)$. We can thus write $|s(\mu) - \overline{s}_N(\mu)| = |\langle L, e(\mu) \rangle| \leq \|L\|_{X'} \, \|e(\mu)\|_X$, and hence

$$|s(\mu) - \overline{s}_N(\mu)| \leq \|L\|_{X'} \; (\|\overline{e}(\mu)\|_X + \|\hat{e}(\mu)\|_X) \; , \tag{7.14}$$

by the triangle inequality.

It immediately follows from standard Galerkin theory that

$$\|\overline{e}(\mu)\|_X \leq C_1 \inf_{v \in X_N} \|\overline{u}(\mu) - v\|_X \; , \tag{7.15}$$

where C_1 depends only on the coercivity and continuity constants associated with $\overline{\mathcal{A}}(\mu)$.

It remains only to bound $\|\hat{e}(\mu)\|_X$; we apply the techniques first developed for analysis of "variational crimes" within the finite element context [75]. In particular, we first note that

$$\langle \mathcal{A}(\mu)(u(\mu) - \overline{u}(\mu)), v \rangle = -\langle \widehat{\mathcal{A}}(\mu) \, \overline{u}(\mu), v \rangle, \quad \forall \, v \in X \; .$$

Expanding $\widehat{\mathcal{A}}(\mu)$, and choosing $v = \hat{e}(\mu) \equiv u(\mu) - \overline{u}(\mu)$, we obtain

$$\begin{aligned}
\langle \mathcal{A}(\mu) \, \hat{e}(\mu), \hat{e}(\mu) \rangle &= -\sum_{q=1}^{\widehat{Q}} \beta_q(\mu) \langle B_q \, \overline{u}(\mu), \hat{e}(\mu) \rangle \\
&\leq \sum_{q=1}^{\widehat{Q}} |\beta_q(\mu)| \left(\sup_{v \in X} \frac{\langle B_q \, \overline{u}(\mu), v \rangle}{|||\, v|_{D_q} \,|||_q} \right) |||\, \hat{e}(\mu)|_{D_q} \,|||_q \qquad (7.16) \\
&\leq \beta_{\widehat{Q}}^{\max}(\mu) \left(\sum_{q=1}^{\widehat{Q}} \gamma_q^2 \, (\overline{u}(\mu)) \right)^{1/2} \\
&\quad \left(\sum_{q=1}^{\widehat{Q}} |||\, \hat{e}(\mu)|_{D_q} \,|||_q^2 \right)^{1/2} \qquad (7.17) \\
&\leq \hat{\beta}^{\max} \, \widehat{\Gamma} \, \|\overline{u}(\mu)\|_X \, \hat{\rho}_\Sigma \, \|\hat{e}(\mu)\|_X \; ,
\end{aligned}$$

from (7.4), (7.8), the Cauchy-Schwarz inequality, (7.7), (7.9), and (7.5). Finally, we obtain

$$\|\hat{e}(\mu)\|_X \leq C_2 \, \hat{\rho}_\Sigma \, \hat{\beta}^{\max} \, \widehat{\Gamma} \; , \tag{7.18}$$

where C_2 is simply $\|F\|_{X'}$ divided by $C_{\mathcal{A}}^2$.

We now assemble (7.14), (7.15), and (7.18); this completes the proof. □ □

We expect that our bound should be reasonably sharp — except in the case in which $\overline{\mathcal{A}}$ is symmetric and $L \to F$. In particular, for $\overline{\mathcal{A}}$ symmetric and $L = F$, $\langle L, \overline{e} \rangle = \langle F, \overline{e} \rangle = \langle \overline{\mathcal{A}} \, \overline{u}, \overline{e} \rangle = \langle \overline{\mathcal{A}} \, \overline{e}, \overline{u} \rangle = \langle \overline{\mathcal{A}} \, \overline{e}, \overline{e} \rangle$ (by Galerkin orthogonality) — $|\overline{s} - \overline{s}_N|$ thus converges quadratically, not linearly, with $\|\overline{e}\|_X$ for this special "compliance" case. Note that we can, in fact, recover quadratic convergence (in $\|\overline{e}\|_X$) for general $\overline{\mathcal{A}}$, L, by introduction of the reduced-basis adjoint techniques described in [62].

In general, we observe that the error in the best approximation, $\inf_{v \in X_N} \|\overline{u}(\mu) - v\|_X$, vanishes very rapidly. Indeed, in certain special cases, we can prove that $\inf_{v \in X_N} \|\overline{u}(\mu) - v\|_X$ vanishes exponentially fast as $N \to \infty$ [46].† It thus follows that N may be chosen quite small. However, we must still develop a computational approach that exploits this dimension reduction.

7.3.3 Computational Strategem: $\overline{s}_N(\mu)$

The essential point to note [37, 62] is that, thanks to the affine parameter dependence of $\overline{\mathcal{A}}(\mu)$, (7.3), we may write

$$\begin{aligned} \overline{\mathcal{A}}_{N\,i,j}(\mu) &\equiv \langle \overline{\mathcal{A}}(\mu) \, \zeta_j, \zeta_i \rangle \\ &= \sum_{q=1}^{\overline{Q}} \alpha_q(\mu) \, \langle A_q \, \zeta_j, \zeta_i \rangle \\ &= \sum_{q=1}^{\overline{Q}} \alpha_q(\mu) \, [A_{N\,q}]_{i,j} \end{aligned} \tag{7.19}$$

where $\underline{A}_{N\,q} \in \mathbb{R}^{N \times N}$ is given by $[A_{N\,q}]_{i,j} = \langle A_q \, \zeta_j, \zeta_i \rangle$, $1 \leq q \leq \overline{Q}$. An *off-line/on-line* decomposition can now be readily identified.

In the *off-line* stage, we compute the $\zeta_n \equiv \overline{u}(\mu_n)$ (say, in Option I) and then form the $\underline{A}_{N\,q}$, $1 \leq q \leq \overline{Q}$ (and $\overline{F}_N, \underline{L}_N$); this requires N expensive "$\overline{\mathcal{A}}$" finite element solutions and $O(\overline{Q} N^2)$ finite-element-vector ($O(\mathcal{N})$) inner products. In the *on-line* stage, for any given new μ, we first form $\overline{\mathcal{A}}_N(\mu)$ from (7.19), then solve (7.11) for $\overline{u}_N(\mu)$, and finally evaluate $\overline{s}_N(\mu) = \overline{L}_N^T \, \overline{u}_N(\mu)$: this requires only $O(\overline{Q} N^2) + O(N^3)$ operations, and only $O(\overline{Q} N^2)$

† In Option I, $\inf_{v \in X_N} \|\overline{u}(\mu) - v\|_X \to 0$ as $N \to \infty$; hence, for fixed $\hat{\beta}^{\max}$, we expect the model truncation contribution to the error (estimate) to dominate the reduced-basis discretization contribution to the error as $N \to \infty$. In Option II, $\inf_{v \in X_N} \|\overline{u}(\mu) - v\|_X$ will approach a constant — presumably $O(\hat{\beta}^{\max})$ — as $N \to \infty$; hence, we expect the model truncation contribution to the error and the reduced-basis discretization contribution to the error to be roughly commensurate as $N \to \infty$.

storage. Thus, as desired, the incremental cost to evaluate $\overline{s}_N(\mu)$ for any given new μ — as proposed, on-line, in a characterization, optimization, or control context — is very small; we can, in many cases, achieve real-time response. (See [13] for an earlier application of this strategy within the reduced-basis context.)

7.4 *A Posteriori* Error Estimation

Although we can, in theory, choose N small, we do not yet know, in practice, just *how small* we can choose N; similarly, we do not yet know, in practice, *how large* we can choose (or tolerate) $\beta_{\widehat{Q}}^{\max}(\mu)$. Conversely, for any given N and $\beta_{\widehat{Q}}^{\max}(\mu)$, we do not yet have an efficiently calculable bound for the error $s(\mu) - \overline{s}_N(\mu)$, and hence we can not determine whether N (respectively, $\beta_{\widehat{Q}}^{\max}(\mu)$) is sufficiently *large* (respectively, sufficiently *small*) for our purposes. We develop here the necessary *a posteriori* error bounds — a necessity for both efficiency and certainty.

7.4.1 Preliminaries: Bound Conditioner $\mathcal{C}(\mu)$

We introduce a symmetric, coercive, continuous bound conditioner [37, 62, 82] $\mathcal{C}(\mu)\colon X \to X'$ that satisfies

$$1 \le \frac{\langle \mathcal{A}^S(\mu)v, v\rangle}{\langle \mathcal{C}(\mu)v, v\rangle} \le \rho, \quad \forall\, v \in X,\ \forall\, \mu \in \mathcal{D}^\mu, \tag{7.20}$$

for some (preferrably small) constant $\rho \in \mathbb{R}$. Here $\mathcal{A}^S(\mu)$ is the symmetric (positive-definite) part of $\mathcal{A}(\mu)$, defined as $\langle \mathcal{A}^S(\mu)w, v\rangle \equiv \frac{1}{2}(\langle \mathcal{A}(\mu)w, v\rangle + \langle \mathcal{A}(\mu)v, w\rangle)$, $\forall\, w, v \in X$. It follows from the coercivity of $\mathcal{C}(\mu)$ that there exists a bound $\hat{\rho}'_\Sigma(\mu)$, independent of $\widehat{Q}$, such that

$$\sum_{q=1}^{\widehat{Q}} |||\, v|_{D_q}\, |||_q^2 \le (\hat{\rho}'_\Sigma(\mu))^2\, \langle \mathcal{C}(\mu)v, v\rangle,\ \forall\, v \in X\ . \tag{7.21}$$

Indeed, we may require

$$\hat{\rho}'_\Sigma(\mu) \le \frac{\rho^{1/2}}{C_{\mathcal{A}}^{1/2}}\, \hat{\rho}_\Sigma, \quad \forall\, \mu \in \mathcal{D}^\mu\ , \tag{7.22}$$

since

$$\begin{aligned}(\hat{\rho}_\Sigma)^2 \ \ge\ & \left(\min_{v\in X} \frac{\sum_{q=1}^{\widehat{Q}} |||\, v|_{D_q}\, |||_q^2}{\langle \mathcal{C}(\mu)v, v\rangle}\right) \\ & \left(\min_{v\in X} \frac{\langle \mathcal{C}(\mu)v, v\rangle}{\langle \mathcal{A}^S(\mu)v, v\rangle}\right) \left(\min_{v\in X} \frac{\langle \mathcal{A}^S(\mu)v, v\rangle}{\|v\|_X^2}\right)\ ;\end{aligned}$$

recall that $C_{\mathcal{A}}$ is the coercivity constant of $\mathcal{A}(\mu)$.

In addition to the spectral condition (7.20), we also require a "computational invertibility" hypothesis. In particular, we shall require that $\mathcal{C}^{-1}(\mu)$ be of the form

$$\mathcal{C}^{-1}(\mu) = \mathcal{C}^{-1}_{\mathcal{J}(\mu)}\ , \tag{7.23}$$

where $\mathcal{J}: \mathcal{D}^\mu \to \{1, \ldots, J\}$ is a parameter-dependent indicator function, and the $\mathcal{C}_j: X \to X'$, $1 \le j \le J$, are parameter-*independent* symmetric, coercive operators. (Note the requirement (7.23) is a special case of the more general computational invertibility condition described in [82].) We shall not dwell here on the many possible fashions in which bound conditioners can be constructed (see [62, 82]); however, we shall give a particular example of a (simple) bound conditioner in our application of Section 7.5.

In what follows, we will require the following simple lemma related to our bound conditioner:

Lemma 5. *Given any $G \in X'$, $\mathcal{C}: X \to X'$ satisfying (7.20), and $U \in X, W \in X$ satisfying $\langle \mathcal{A}U, v\rangle = \langle G, v\rangle$, $\forall\, v \in X$, and $\langle \mathcal{C}W, v\rangle = \langle G, v\rangle$, $\forall\, v \in X$, respectively, then $\langle \mathcal{A}^S U, U\rangle \le \langle \mathcal{C}W, W\rangle$.*

Proof. We have that $\langle \mathcal{A}U, v\rangle = \langle \mathcal{C}W, v\rangle$, $\forall\, v \in X$; choosing $v = U$ thus gives

$$\begin{aligned} \langle \mathcal{A}^S U, U\rangle &= \langle \mathcal{A}U, U\rangle \\ &= \langle \mathcal{C}W, U\rangle \\ &\le \langle \mathcal{C}W, W\rangle^{1/2}\, \langle \mathcal{C}U, U\rangle^{1/2} \\ &\le \langle \mathcal{C}W, W\rangle^{1/2}\, \langle \mathcal{A}^S U, U\rangle^{1/2}\ , \end{aligned}$$

by the Cauchy-Schwarz inequality and our spectral condition (7.20). Thus $\langle \mathcal{A}^S U, U\rangle \le \langle \mathcal{C}W, W\rangle$, as desired. □ □

7.4.2 Error Bound

Estimator $\Delta_N(\mu)$

Our error estimator for $|s(\mu) - \overline{s}_N(\mu)|$ is given by

$$\Delta_N(\mu) \equiv |||L|||_{X'} \left(\overline{\delta}_N(\mu) + \hat{\delta}_N(\mu) \right)\ , \tag{7.24}$$

where

$$|||L|||_{X'} = \sup_{v \in X} \frac{\langle L, v\rangle}{\langle \mathcal{C}(\mu)v, v\rangle^{1/2}}\ , \tag{7.25}$$

and

$$\overline{\delta}_N(\mu) \equiv \langle \mathcal{C}(\mu)\, \overline{E}(\mu), \overline{E}(\mu)\rangle^{1/2}\ , \tag{7.26}$$

$$\hat{\delta}_N(\mu) \equiv \hat{\rho}'_\Sigma(\mu)\, \beta^{\max}_{\widehat{Q}}(\mu) \left(\sum_{q=1}^{\widehat{Q}} \gamma_q^2(\overline{u}_N(\mu)) \right)^{1/2}\ . \tag{7.27}$$

Here $\overline{E}(\mu) \in X$ satisfies

$$\langle \mathcal{C}(\mu)\, \overline{E}(\mu), v\rangle = \langle F, v\rangle - \langle \overline{\mathcal{A}}(\mu)\, \overline{u}_N(\mu), v\rangle,\ \forall\, v \in X\ ; \tag{7.28}$$

and $\beta^{\max}_{\widehat{Q}}(\mu) \in \mathbb{R}$ is the *given* model truncation error (implicitly related to $\widehat{Q}$ and the $\beta_q(\mu)$ through (7.6)). It is clear that $\overline{\delta}_N(\mu)$ measures the error due to discretization — reduced-basis approximation of $\overline{u}(\mu) \in X$ by $\overline{u}_N(\mu) \in X_N$; and $\hat{\delta}_N(\mu)$ measures the error due to model truncation — approximation of $\mathcal{A}(\mu)$ by $\overline{\mathcal{A}}(\mu)$.

Analysis of $\Delta_N(\mu)$

We now show that our estimator $\Delta_N(\mu)$ is, in fact, a rigorous *upper bound* for $|s(\mu) - \overline{s}_N(\mu)|$:

Proposition 11. *For $\mathcal{C}: X \to X'$ satisfying (7.20)[†], and $\hat{\rho}'_\Sigma(\mu)$ satisfying (7.21), $|s(\mu) - \overline{s}_N(\mu)| \le \Delta_N(\mu)$, $\forall\, \mu \in \mathcal{D}^\mu$, and $\forall\, N \in \mathbb{N}$.*

Proof. We first note that our error $e(\mu) = u(\mu) - \overline{u}_N(\mu)$ satisfies

$$\begin{aligned} \langle \mathcal{A}(\mu)\, e(\mu), v\rangle = \langle F, v\rangle \quad &- \quad \langle \overline{\mathcal{A}}(\mu)\, \overline{u}_N(\mu), v\rangle \\ &- \quad \langle \widehat{\mathcal{A}}(\mu)\, \overline{u}_N(\mu), v\rangle, \quad \forall\, v \in X\ . \end{aligned}$$

We next define $\overline{\varepsilon}(\mu) \in X$ and $\hat{\varepsilon}(\mu) \in X$ by

$$\langle \mathcal{A}(\mu)\, \overline{\varepsilon}(\mu), v\rangle = \langle F, v\rangle - \langle \overline{\mathcal{A}}(\mu)\, \overline{u}_N(\mu), v\rangle,\ \forall\, v \in X\ , \tag{7.29}$$

and

$$\langle \mathcal{A}(\mu)\, \hat{\varepsilon}(\mu), v\rangle = -\langle \widehat{\mathcal{A}}(\mu)\, \overline{u}_N(\mu), v\rangle, \quad \forall\, v \in X\ , \tag{7.30}$$

such that $e(\mu) = \overline{\varepsilon}(\mu) + \hat{\varepsilon}(\mu)$. We thus obtain

$$\begin{aligned} \langle \mathcal{A}^S(\mu)\, e(\mu), e(\mu)\rangle^{1/2} \ \le\ & \langle \mathcal{A}^S(\mu)\, \overline{\varepsilon}(\mu), \overline{\varepsilon}(\mu)\rangle^{1/2} \\ & + \langle \mathcal{A}^S(\mu)\, \hat{\varepsilon}(\mu), \hat{\varepsilon}(\mu)\rangle^{1/2}\ , \end{aligned} \tag{7.31}$$

by the triangle inequality.

We now write

$$\begin{aligned} |s(\mu) - \overline{s}_N(\mu)| &= |\langle L, e(\mu)\rangle| \\ &\le \left(\sup_{v\in X} \frac{\langle L, v\rangle}{\langle \mathcal{A}^S(\mu) v, v\rangle^{1/2}}\right) \langle \mathcal{A}^S(\mu)\, e(\mu), e(\mu)\rangle^{1/2} \\ &\le \left(\sup_{v\in X} \frac{\langle L, v\rangle}{\langle \mathcal{C}(\mu) v, v\rangle^{1/2}}\right) \left(\langle \mathcal{A}^S(\mu)\, \overline{\varepsilon}(\mu), \overline{\varepsilon}(\mu)\rangle^{1/2} \right. \\ &\qquad \left. + \langle \mathcal{A}^S(\mu)\, \hat{\varepsilon}(\mu), \hat{\varepsilon}(\mu)\rangle^{1/2}\right) \end{aligned} \tag{7.32}$$

from (7.20) and (7.31). We next note from Lemma 5, (7.29), and (7.28), that

$$\langle \mathcal{A}^S(\mu)\, \overline{\varepsilon}(\mu), \overline{\varepsilon}(\mu)\rangle^{1/2} \le \langle \mathcal{C}(\mu)\, \overline{E}(\mu), \overline{E}(\mu)\rangle^{1/2}\ . \tag{7.33}$$

It remains only to treat $\langle \mathcal{A}^S(\mu)\, \hat{\varepsilon}(\mu), \hat{\varepsilon}(\mu)\rangle^{1/2}$.

We first define $\widehat{E}(\mu) \in X$ as the solution to

$$\langle \mathcal{C}(\mu)\, \widehat{E}(\mu), v\rangle = -\langle \widehat{A}(\mu)\, \overline{u}_N(\mu), v\rangle, \quad \forall\, v \in X\ ;$$

[†] In fact, for this bound proof, we only require the "left" spectral condition of (7.20), $1 \le \frac{\langle \mathcal{A}^S(\mu) v, v\rangle}{\langle \mathcal{C}(\mu) v, v\rangle}$, $\forall\, v \in X$, $\forall\, \mu \in \mathcal{D}^\mu$.

it thus follows from Lemma 5 that $\langle \mathcal{A}^S(\mu)\,\hat{\varepsilon}(\mu), \hat{\varepsilon}(\mu)\rangle \leq \langle \mathcal{C}(\mu)\,\widehat{E}(\mu), \widehat{E}(\mu)\rangle$. We now note that

$$\begin{aligned}
\langle \mathcal{C}(\mu)\,\widehat{E}(\mu), \widehat{E}(\mu)\rangle &= -\sum_{q=1}^{\widehat{Q}} \beta_q(\mu) \langle B_q\,\overline{u}_N(\mu), \widehat{E}(\mu)\rangle \\
&\leq \sum_{q=1}^{\widehat{Q}} |\beta_q(\mu)|\,\gamma_q(\overline{u}_N(\mu))\,|||\,\widehat{E}(\mu)|_{D_q}\,|||_q \\
&\leq \beta_{\widehat{Q}}^{\max}(\mu) \left(\sum_{q=1}^{\widehat{Q}} \gamma_q^2(\overline{u}_N(\mu))\right)^{1/2} \\
&\quad \left(\sum_{q=1}^{\widehat{Q}} |||\,\widehat{E}(\mu)|_{D_q}\,|||_q^2\right)^{1/2} \\
&\leq \hat{\rho}'_\Sigma(\mu)\,\beta_{\widehat{Q}}^{\max}(\mu) \left(\sum_{q=1}^{\widehat{Q}} \gamma_q^2(\overline{u}_N(\mu))\right)^{1/2} \\
&\quad \langle \mathcal{C}(\mu)\,\widehat{E}(\mu), \widehat{E}(\mu)\rangle^{1/2}\ ,
\end{aligned}$$

from (7.4), (7.8), (7.6), the Cauchy-Schwarz inequality, and (7.21). We conclude that

$$\begin{aligned}
&\langle \mathcal{A}^S(\mu)\,\hat{\varepsilon}(\mu), \hat{\varepsilon}(\mu)\rangle^{1/2} \leq \\
&\qquad \hat{\rho}'_\Sigma(\mu)\,\beta_{\widehat{Q}}^{\max}(\mu) \left(\sum_{q=1}^{\widehat{Q}} \gamma_q^2(\overline{u}_N(\mu))\right)^{1/2}\ . \qquad (7.34)
\end{aligned}$$

It follows from (7.32), (7.33), and (7.34) that $\Delta_N(\mu)$ of (7.24)–(7.27) is, indeed, an upper bound for $|s(\mu) - \overline{s}_N(\mu)|$. □ □

Of course we wish not only that $\Delta_N(\mu)$ is an upper bound for $|s(\mu) - \overline{s}_N(\mu)|$, but also that $\Delta_N(\mu)$ is a *good* (sharp) upper bound for $|s(\mu) - \overline{s}_N(\mu)|$. In particular, if we define the effectivity, $\eta_N(\mu)$, as the ratio of the estimated error to the true error,

$$\eta_N(\mu) \equiv \frac{\Delta_N(\mu)}{|s(\mu) - \overline{s}_N(\mu)|}\ ,$$

we would like to prove not only, as in Proposition 11, that $1 \leq \eta_N(\mu)$ — thus establishing bounds — but also that $\eta_N(\mu) \leq \Upsilon$, Υ not too large — thus establishing *good* bounds.

It is clear from our *a priori* and *a posteriori* bounds for the error, (7.32), (7.13), and (7.24)–(7.27), respectively, that

$$\frac{|||L|||_{X'}}{\|L\|_{X'}} \leq \Upsilon_1\ , \qquad (7.35)$$

for Υ_1 independent of μ and N,

$$\frac{\overline{\delta}_N(\mu)}{\langle \mathcal{A}^S(\mu)\,\overline{\varepsilon}(\mu), \overline{\varepsilon}(\mu)\rangle^{1/2}} \leq \Upsilon_2 \ , \tag{7.36}$$

for Υ_2 independent of μ and N, and

$$\frac{\hat{\delta}_N(\mu)}{C_2\,\hat{\rho}_\Sigma\,\hat{\beta}^{\max}\,\widehat{\Gamma}} \leq \Upsilon_3 \ , \tag{7.37}$$

for Υ_3 independent of μ, N, *and* $\widehat{Q}$, are meaningful *necessary* conditions for good effectivities. We now prove that (7.35)–(7.37) are, indeed, satisfied:

Proposition 12. *For $\mathcal{C}\colon X \to X'$ satisfying (7.20)[†], $\beta^{\max}_{\widehat{Q}}(\mu)$ satisfying (7.7), and $\hat{\rho}'_\Sigma(\mu)$ satisfying (7.22),*

$$\frac{|||L|||_{X'}}{\|L\|_{X'}} \leq \frac{\rho^{1/2}}{C_{\mathcal{A}}^{1/2}}, \ \forall\, \mu \in \mathcal{D}^\mu, \ \forall\, N \in \mathbb{N}, \tag{7.38}$$

$$\frac{\overline{\delta}_N(\mu)}{\langle \mathcal{A}^S(\mu)\,\overline{\varepsilon}(\mu), \overline{\varepsilon}(\mu)\rangle^{1/2}} \leq C_3\,\rho^{1/2}, \quad \forall\, \mu \in \mathcal{D}^\mu, \ \forall\, N \in \mathbb{N}, \tag{7.39}$$

and

$$\frac{\hat{\delta}_N(\mu)}{C_2\,\hat{\rho}_\Sigma\,\hat{\beta}^{\max}\,\widehat{\Gamma}} \leq C_{\mathcal{A}}^{1/2}\,\rho^{1/2} \ , \quad \forall\, \mu \in \mathcal{D}^\mu, \ \forall\, N \in \mathbb{N}, \ \forall\, \widehat{Q} \in \mathbb{N} \ , \tag{7.40}$$

where $C_{\mathcal{A}}$ is the coercivity constant of $\mathcal{A}$, C_2 is defined as in Proposition 10, and C_3 is a continuity constant.

Proof. To demonstrate (7.38), we simply note that

$$\begin{aligned}
&\sup_{v\in X} \frac{\langle L, v\rangle}{\langle \mathcal{C}(\mu)v, v\rangle^{1/2}} \\
&\quad = \sup_{v\in X} \left(\frac{\langle L, v\rangle}{\|v\|_X} \frac{\|v\|_X}{\langle \mathcal{A}^S(\mu)v, v\rangle^{1/2}} \frac{\langle \mathcal{A}^S(\mu)v, v\rangle^{1/2}}{\langle \mathcal{C}(\mu)v, v\rangle^{1/2}} \right) \\
&\quad \leq C_{\mathcal{A}}^{-1/2}\,\rho^{1/2}\,\|L\|_{X'} \ ,
\end{aligned}$$

from our spectral condition (7.20) and the coercivity of $\mathcal{A}$.

[†] In fact, for this effectivity proof, we only require the "right" spectral condition of (7.20), $\frac{\langle \mathcal{A}^S(\mu)v, v\rangle}{\langle \mathcal{C}(\mu)v, v\rangle} \leq \rho$, $\forall\, v \in X$, $\forall\, \mu \in \mathcal{D}^\mu$.

To demonstrate (7.39), we first define

$$C_3 = \sup_{\mu\in\mathcal{D}^\mu} \sup_{v\in X,\, w\in X} \frac{\langle \mathcal{A}(\mu)w, v\rangle}{\langle \mathcal{A}^S(\mu)w, w\rangle^{1/2}\, \langle \mathcal{A}^S(\mu)v, v\rangle^{1/2}} \ ,$$

which is perforce finite due to the continuity and coercivity of $\mathcal{A}$. We then note that

$$\begin{aligned}\langle \mathcal{C}(\mu)\, \overline{E}(\mu), \overline{E}(\mu)\rangle &= \langle \mathcal{A}(\mu)\, \overline{\varepsilon}(\mu), \overline{E}(\mu)\rangle \\ &\leq C_3\, \rho^{1/2} \langle \mathcal{A}^S(\mu)\, \overline{\varepsilon}(\mu), \overline{\varepsilon}(\mu)\rangle^{1/2} \langle \mathcal{C}(\mu)\, \overline{E}(\mu), \overline{E}(\mu)\rangle^{1/2} \ ,\end{aligned}$$

from (7.28), (7.29), and (7.20). The desired result (7.39) immediately follows.

Finally, to demonstrate (7.40), we note that

$$\beta_{\widehat{Q}}^{\max}(\mu) \ \leq \ \hat{\beta}^{\max} \ ,$$

$$\left(\sum_{q=1}^{\widehat{Q}} \gamma_q^2(\overline{u}_N(\mu))\right)^{1/2} \leq \|\overline{u}_N(\mu)\|_X\, \widehat{\Gamma} \leq C_2\, C_{\mathcal{A}}\, \widehat{\Gamma} \ ,$$

and

$$\hat{\rho}'_\Sigma(\mu) \leq \frac{\rho^{1/2}}{C_{\mathcal{A}}^{1/2}}\, \hat{\rho}_\Sigma \ ,$$

from (7.7), (7.9) and the definition of C_2 in Proposition 10, and (7.22), respectively. The desired result then directly follows from the definition of $\hat{\delta}_N(\mu)$, (7.27). □

□

We thus observe that good bound conditioners are a prerequisite for good effectivities — in particular, we expect that $\eta_N(\mu)$ will scale with ρ, the constant associated with the "upper" spectral condition of (7.20). In practice, it is often possible to develop (e.g., multipoint [82]) bound conditioners in which ρ is $O(10)$ or better.

We re-emphasize that (7.38)–(7.40) are only *necessary* conditions for good effectivities, and thus the estimate $\eta_N(\mu) \leq O(\rho)$ is not rigorous. In particular, (7.38)–(7.40) relate not to the ratio of $\Delta_N(\mu)$ to the true error, but rather to the ratio of $\Delta_N(\mu)$ to our *a priori bounds* for the true error; if the latter are not sharp, our effectivities may be worse than $O(\rho)$. One such situation is the compliance case, $\mathcal{A} = \mathcal{A}^S$ and $L \to F$, in which, indeed, we will *not* obtain good effectivities as $N \to \infty$ (unless we modify our estimators accordingly — adjoint methods automatically provide the necessary correction [62]). In general, however, our necessary conditions should be meaningful.

Computational Strategem: $\Delta_N(\mu)$

We indicate here the off-line/on-line treatment of each of the three ingredients that compose $\Delta_N(\mu)$: $|||L|||_{X'}$; $\overline{\delta}_N(\mu)$; and $\hat{\delta}_N(\mu)$.

Calculation of $|||\boldsymbol{L}|||_{\boldsymbol{X'}}(\boldsymbol{\mu})$. We first note that $|||L|||_{X'}$ can be expressed as

$$\begin{aligned}|||L|||_{X'}(\mu) &= \langle \mathcal{C}(\mu) w_L(\mu), w_L(\mu)\rangle^{1/2} \ , \\ &= \langle L, w_L(\mu)\rangle^{1/2} \ ,\end{aligned}$$

where $w_L(\mu) \in X$ satisfies

$$\langle \mathcal{C}(\mu) w_L(\mu), v \rangle = \langle L, v \rangle, \quad \forall\, v \in X \ .$$

We now invoke our "computational invertibility" hypothesis on $\mathcal{C}(\mu)$, (7.23), to write

$$w_L(\mu) = \mathcal{C}^{-1}_{\mathcal{J}(\mu)}\, L \ ;$$

and hence $|||L|||_{X'}(\mu)$ can be evalulated as

$$|||L|||_{X'}(\mu) = \langle L, \mathcal{C}^{-1}_{\mathcal{J}(\mu)}\, L \rangle^{1/2} \ .$$

The off-line/on-line decomposition is now clear.

In the *off-line* stage, we compute the μ-independent quantities $\omega_j \equiv \langle L, \mathcal{C}_j^{-1}\, L \rangle^{1/2}$, $1 \le j \le J$. (Note in this paper we shall not give detailed complexity estimates for the off-line stage of the bound calculations; our focus here will be on the on-line stage.) Then, in the *on-line* stage, given any new value of μ, we simply evaluate

$$|||L|||_{X'}(\mu) = \omega_{\mathcal{J}(\mu)} \ .$$

The on-line complexity is thus $O(1)$ (and certainly independent of $\mathcal{N}$).

Calculation of $\overline{\delta}_N(\mu)$. We first note from (7.26) and (7.28) that $\overline{\delta}_N(\mu)$ may be expressed as

$$\overline{\delta}_N(\mu) = \langle \overline{R}(\mu), \overline{E}(\mu) \rangle^{1/2} \ ,$$

where $\overline{E}(\mu) \in X$ satisfies

$$\langle \mathcal{C}(\mu) \overline{E}(\mu), v \rangle = \langle \overline{R}(\mu), v \rangle, \quad \forall v \in X \ ,$$

and

$$\langle \overline{R}(\mu), v \rangle = \langle F, v \rangle - \langle \overline{\mathcal{A}}(\mu)\, \overline{u}_N(\mu), v \rangle, \quad \forall v \in X \ ,$$

is the "reduced-basis" part of the residual.

We now invoke (7.12), the affine dependence of (7.3), and the computational invertibility hypothesis (7.23) to express $\overline{R}(\mu)$ and $\overline{E}(\mu)$ as

$$\overline{R}(\mu) = F - \sum_{n=1}^{N} \sum_{q=1}^{\overline{Q}} \overline{u}_{Nn}(\mu)\, \alpha_q(\mu)\, A_q \zeta_n \ ,$$

and

$$\begin{aligned} \overline{E}(\mu) \ &= \ \mathcal{C}^{-1}_{\mathcal{J}(\mu)}\, F \\ &\quad - \sum_{n'=1}^{N} \sum_{q'=1}^{\overline{Q}} \overline{u}_{Nn'}(\mu)\, \alpha_{q'}(\mu)\, \mathcal{C}^{-1}_{\mathcal{J}(\mu)}\, A_{q'} \zeta_{n'} \ , \end{aligned}$$

respectively. We can thus evaluate $\overline{\delta}_N(\mu)$ of (7.26) as

$$\overline{\delta}_N(\mu) = \left(F - \sum_{n=1}^{N} \sum_{q=1}^{\overline{Q}} \overline{u}_{Nn}(\mu)\, \alpha_q(\mu)\, A_q \zeta_n, \; \mathcal{C}^{-1}_{\mathcal{J}(\mu)} F - \sum_{n'=1}^{N} \sum_{q'=1}^{\overline{Q}} \overline{u}_{Nn'}(\mu)\, \alpha_{q'}(\mu)\, \mathcal{C}^{-1}_{\mathcal{J}(\mu)}\, A_{q'} \zeta_{n'} \right)^{1/2} .$$

The off-line/on-line decomposition is now clear.

In the *off-line* stage, we compute the μ-independent inner products

$$\begin{aligned} \Lambda^0_j &= \langle F, \mathcal{C}_j^{-1} F \rangle, && 1 \le j \le J , \\ \Lambda^1_{j:\, nq} &= -2\langle F, \mathcal{C}_j^{-1} A_q \zeta_n \rangle, && 1 \le n \le N,\ 1 \le q \le \overline{Q} , \\ \Lambda^2_{j:\, nqn'q'} &= \langle A_q \zeta_n, \mathcal{C}_j^{-1} A_{q'} \zeta_{n'} \rangle, && 1 \le n, n' \le N,\ 1 \le q, q' \le \overline{Q} . \end{aligned}$$

Then, in the *on-line* stage, given any new value of μ, we simply perform the sum

$$\overline{\delta}_N(\mu) = \left(\Lambda^0_{\mathcal{J}(\mu)} + \sum_{n=1}^{N} \sum_{q=1}^{\overline{Q}} \overline{u}_{Nn}(\mu)\, \alpha_q(\mu) \left(\Lambda^1_{\mathcal{J}(\mu):\, nq} + \sum_{n'=1}^{N} \sum_{q'=1}^{\overline{Q}} \overline{u}_{Nn'}(\mu)\, \alpha_{q'}(\mu)\, \Lambda^2_{\mathcal{J}(\mu):\, nqn'q'} \right) \right)^{1/2} .$$

The on-line complexity is thus, to leading order, $O(\overline{Q}^2 N^2)$ — and hence independent of $\mathcal{N}$.

Furthermore, for $\overline{Q}$ small — certainly plausible given our $\mathcal{A}(\mu) = \overline{\mathcal{A}}(\mu) + \widehat{\mathcal{A}}(\mu)$ decomposition — the on-line complexity will be small not only in relative, but also in absolute, terms. However, were we to treat the $\widehat{\mathcal{A}}(\mu)$ terms similarly, the resulting complexity — $O(\widehat{Q}^2 N^2)$ — would be prohibitive, since we may anticipate $\widehat{Q} \approx O(\mathcal{N})$ in many situations. It is for this reason — and the fact that we may not have detailed knowledge of the $\beta_q(\mu)$ — that we must treat the model truncation *a posteriori* contribution, $\hat{\delta}_N(\mu)$, differently from the discretization *a posteriori* contribution, $\overline{\delta}_N(\mu)$.

Calculation of $\hat{\delta}_N(\mu)$. We first define

$$\hat{\delta}^2_N(\mu) \equiv \sum_{q=1}^{\widehat{Q}} \gamma_q^2(\overline{u}_N(\mu)) ,$$

in terms of which $\hat{\delta}_N(\mu)$ can be computed as $\hat{\delta}_N(\mu) = \hat{\rho}'_\Sigma(\mu)\, \beta^{\max}_{\widehat{Q}}(\mu)\, \hat{\hat{\delta}}_N(\mu)$. We shall not discuss in detail the (solely *off-line*) calculation of $\hat{\rho}'_\Sigma(\mu)$: it may either be determined by inspection (see our example of Section 7.5); or computed more precisely as $\hat{\rho}'_\Sigma(\mu) =$

$\lambda_{\mathcal{J}(\mu)}^{\Sigma,\max}$, where $\lambda_j^{\Sigma,\max}$, $1 \le j \le J$, is the maximum eigenvalue associated with the parameter-*independent* Rayleigh quotient

$$\frac{\sum_{q=1}^{\widehat{Q}} ||| v|_{D_q} |||_q^2}{\langle \mathcal{C}_j v, v\rangle} .$$

Finally, we shall assume, as discussed earlier, that we are directly given $\beta_{\widehat{Q}}^{\max}(\mu)$.[†] We shall thus focus our attention on $\hat{\hat{\delta}}_N(\mu)$.

It is a simple matter to show that

$$\gamma_q^2(\overline{u}_N(\mu)) = ||| \sigma_q(\mu) |||_q^2 ,$$

where $\sigma_q(\mu) \in X_q$ satisfies

$$\big((\sigma_q(\mu), v|_{D_q})\big)_q = \langle B_q\, \overline{u}_N(\mu), v|_{D_q}\rangle, \quad \forall\, v \in X . \tag{7.41}$$

Here X_q is the space of functions in $H^1(D_q)$ that vanish on that part of ∂D_q that coincides with the Dirichlet part of $\partial\Omega$. If we thus define the parameter-independent functions $\sigma_{nq} \in X_q$, $1 \le n \le N$, $1 \le q \le \widehat{Q}$, as

$$\big((\sigma_{nq}, v|_{D_q})\big)_q = \langle B_q\, \zeta_n, v|_{D_q}\rangle, \quad \forall\, v \in X , \tag{7.42}$$

it follows from (7.41), (7.12), and (7.42) that

$$||| \sigma_q(\mu) |||_q^2 = \sum_{n=1}^{N}\sum_{n'=1}^{N} \overline{u}_{Nn}(\mu)\, \overline{u}_{Nn'}(\mu)\, \big((\sigma_{nq}, \sigma_{n'q})\big)_q$$

and hence

$$\hat{\hat{\delta}}_N^2(\mu) = \sum_{n=1}^{N}\sum_{n'=1}^{N} \overline{u}_{Nn}(\mu)\, \overline{u}_{Nn'}(\mu) \left(\sum_{q=1}^{\widehat{Q}} \big((\sigma_{nq}, \sigma_{n'q})\big)_q\right) .$$

Note that for D_q a small region of Ω, (7.42) is essentially local (in practice, of course, we need only test on functions $v \in X_q$), and hence very inexpensive. The off-line/on-line decomposition is now clear.

In the *off-line* stage, we form the μ-independent quantity

$$\Xi_{nn'} = \sum_{q=1}^{\widehat{Q}} \big((\sigma_{nq}, \sigma_{n'q})\big)_q \quad 1 \le n, n' \le N .$$

Then, in the *on-line* stage, given μ, $\hat{\rho}'_\Sigma(\mu)$, and $\beta_{\widehat{Q}}^{\max}(\mu)$, we evaluate

$$\hat{\delta}_N(\mu) = \hat{\rho}'_\Sigma(\mu)\, \beta_{\widehat{Q}}^{\max}(\mu) \left(\sum_{n=1}^{N}\sum_{n'=1}^{N} \overline{u}_{Nn}(\mu)\, \overline{u}_{Nn'}(\mu)\, \Xi_{nn'}\right)^{1/2} .$$

[†] Note that there may be *off-line* calculations required for subsequent rapid on-line calculation of $\beta_{\widehat{Q}}^{\max}(\mu)$ — we consider this problem-dependent collateral computation as "external" to our general procedure. We discuss this further in Section 7.5.

The on-line complexity is thus $O(N^2)$, and hence *independent* of $\widehat{Q}$ (and $\mathcal{N}$).[†] Indeed, the fraction of the on-line computational cost attributable to $\hat{\delta}_N(\mu)$ is typically negligible.

7.5 Illustrative Application

We consider here a three-dimensional conjugate (solid-fluid) heat transfer application — a model system for multifunctional (thermostructural) microtrusses relevant, for example, to lightweight walls for regeneratively cooled combustion chambers. The mathematical model is a coupled Poisson-Convection Diffusion problem in a finite-thickness-wall rectangular duct.

This problem will serve two functions. First, in its generality, it will serve as an example of (some of) the ways in which our framework may, in practice, be applied. Second, in one particular case, it will serve as a vehicle for numerical evidence to support the theoretical claims of earlier sections.

7.5.1 Problem Statement

Governing Equations

The domain is given by $\Omega =]0, L_x[\times]0, L_y[\times]0, L_z[$; we further define the fluid domain as $\Omega^f =]t_x^w, L_x - t_x^w[\times]t_y^w, L_y - t_y^w[\times]0, L_z[$, and the solid domain as $\Omega^s = \Omega\backslash\overline{\Omega}^f$ (where $\overline{\ }$ here refers to closure); we shall also require $\Omega_{2d} =]0, L_x[\times]0, L_y[$, $\Omega_{2d}^f =]t_x^w, L_x - t_x^w[\times]t_y^w, L_y - t_y^w[$, and $\Omega_{2d}^s = \Omega_{2d}\backslash\overline{\Omega}_{2d}^f$. The geometry is depicted in Figure 7.1.

We next define the requisite function spaces. We first introduce $X \equiv \{v \in H^1(\Omega) | v|_{(x,y)\in\Omega_{2d}^f,\, z=0} = 0\}$. We shall also require $X^f \equiv H_0^1(\Omega^f) = \{v \in H^1(\Omega^f) \mid v|_{\partial\Omega_{\rm side}^f} = 0\}$, where $\partial\Omega_{\rm side}^f = \partial\Omega_{2d}^f \times]0, L_z[$, and $\partial\Omega_{2d}^f$ is the boundary of Ω_{2d}^f; and we shall need $X_{2d}^f \equiv H_0^1(\Omega_{2d}^f) = \{v \in H^1(\Omega_{2d}^f) \mid v|_{\partial\Omega_{2d}^f} = 0\}$. In addition, we shall require the space $L^\infty(\Omega_{2d}^f)$ of functions v for which $\|v\|_{L^\infty(\Omega_{2d}^f)} \equiv \operatorname{ess\,sup}_{(x,y)\in\Omega_{2d}^f} |v(x,y)|$ is finite.

We now present the exact problem statement: For specified $\mu \equiv (\mu_1, \mu_2) \in \mathcal{D}^\mu \subset \mathbb{R}^{P=2}$, find $s(\mu) = \langle L, u(\mu)\rangle$, where $u(\mu) \in X$ satisfies

$$\langle \mathcal{A}(\mu)\, u(\mu), v\rangle = \langle F, v\rangle, \quad \forall\, v \in X\ .$$

Here

$$\langle L, v\rangle \equiv \int_{\substack{x=L_x \\ 0<y<L_y,\, 0<z<L_z}} v\ , \tag{7.43}$$

$$\langle F, v\rangle \equiv \int_{\substack{y=L_y \\ 0<x<L_x,\, 0<z<L_z}} v\ , \tag{7.44}$$

[†] Note in the situation in which B_q depends on μ affinely through (say) $\widetilde{Q}$ parameter functions (see Section 7.2, Footnote [†]), our on-line complexity for $\hat{\delta}_N(\mu)$ will now be $O(\widetilde{Q}^2N^2)$; the calculation remains tractable if $\widetilde{Q}$ is not too large.

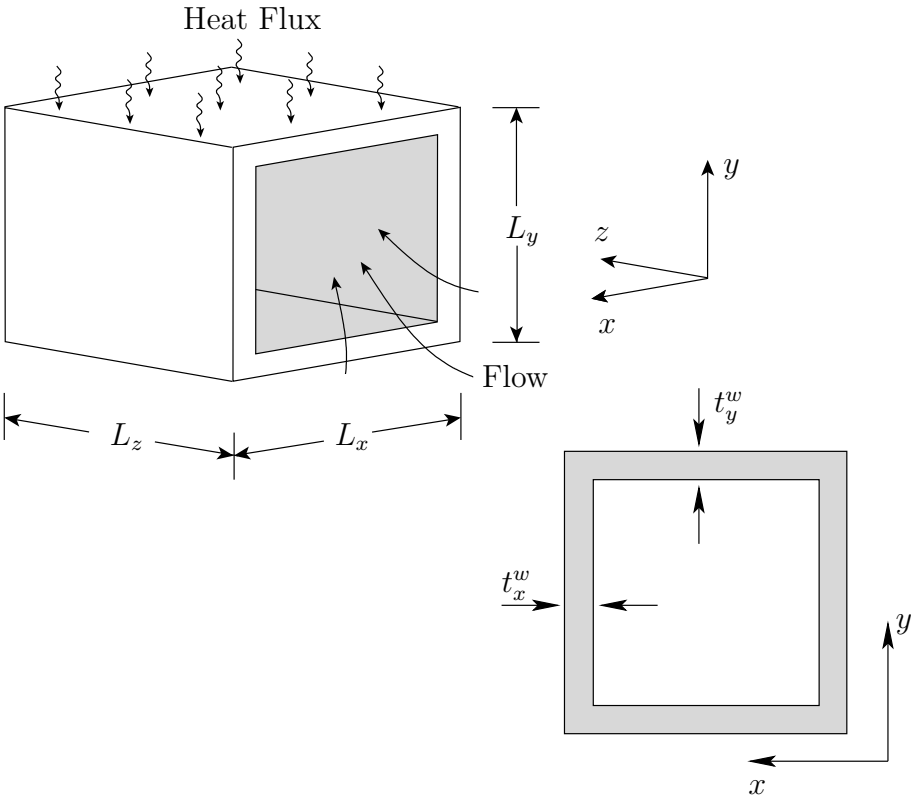


Figure 7.1: Three-dimensional finite-thickness-wall duct.

and

$$\langle \mathcal{A}(\mu)\, w, v\rangle \equiv \int_{\Omega^f} \nabla w \cdot \nabla v + \theta \int_{\Omega^f} U(\mu)\, \frac{\partial w}{\partial z}\, v + \mu_1 \int_{\Omega^s} \nabla w \cdot \nabla v + \mu_2 \int_{\Omega^s} y\, \nabla w \cdot \nabla v \;; \tag{7.45}$$

θ and U will be defined below.

We thus have conduction in the solid, and conduction and convection in the fluid. As regards the former, we assume that the thermal conductivity (normalized relative to the fluid conductivity) is given by $\mu_1 + \mu_2 y$. As regards the latter, we assume that the viscous flow is (laminar and) fully developed such that the velocity $\mathcal{U}(\mu) \in (X^f)^3$ can be expressed as $\mathcal{U}(x, y, z; \mu) = U(x, y; \mu)\, \hat{z}$ (here $\hat{z}$ is the unit normal in the z direction) for $U(\mu) \in X^f_{2d}$; we shall also assume that $U(\mu)$ is *non-negative* and bounded over Ω^f_{2d}. The velocity $U(\mu)$ is defined relative to a unit pressure gradient; $\theta \in \mathbb{R}_+$ thus determines the importance of convection (the conventional Peclet number is given, roughly, by θL_x^3), or, more precisely, the entrance length associated with the developing thermal boundary layer on $\partial\Omega^f_{\text{side}}$. In actual practice, for our particular problem, $U(\mu)$ will not be a function of μ; but we retain this dependence in our exposition since more generally (e.g., if μ reflects piecewise-continuous affine geometric transformations) $U(\mu)$ *will* depend on μ.

Our essential and natural boundary conditions on the temperature u are implicit in F, $\mathcal{A}(\mu)$, and X. In particular, we impose inhomogeneous Neumann (prescribed heat flux) on the top external side wall; homogeneous Neumann (insulated) on the bottom, left, and right external side walls, as well as on $(x, y) \in \Omega^s_{2d}$, $z = 0$, and $(x, y) \in \Omega_{2d}$, $z = L_z$; and homogeneous Dirichlet (zero temperature) at "inflow," $(x, y) \in \Omega^f_{2d}$, $z = 0$. Continuity of temperature and flux are imposed on the internal walls, $\partial\Omega^f_{\text{side}}$. The output of interest, $s(\mu) = \langle L, u(\mu)\rangle$, is the average temperature over the left external side wall.

Model Truncation and Interpretations

We now assume that we are given some low-dimensional (but presumably reasonably accurate) approximation to $U(\mu)$, $U_M(\mu) \in X^f_{2d}$, of the form

$$U_M(x, y; \mu) = \sum_{m=1}^{M} U_{Mm}(\mu)\, \xi_m(x, y) = \underline{U}_M(\mu)^T\, \underline{\xi}\;, \tag{7.46}$$

where $\underline{U}_M(\mu) = (U_{M1}(\mu) \cdots U_{MM}(\mu))^T \in \mathbb{R}^M$, and $\underline{\xi} = (\xi_1 \ldots \xi_M)^T \in (X^f_{2d})^M$. We further assume that we are *given* a bound for the L^∞-norm of the error in the velocity, $\mathcal{E}_M(\mu)$, such that

$$\|U(\mu) - U_M(\mu)\|_{L^\infty(\Omega^f_{2d})} \le \mathcal{E}_M(\mu)\;. \tag{7.47}$$

Finally, we shall also suppose (plausibly) that $U_M(x, y; \mu)$ is *non-negative* for all $(x, y) \in \Omega^f_{2d}$.

We then define our model-truncated form (for given M) as

$$\begin{aligned}\langle \overline{\mathcal{A}}(\mu)w, v\rangle = & \\ & \int_{\Omega^f} \nabla w \cdot \nabla v + \theta \int_{\Omega^f} U_M(\mu)\, \frac{\partial w}{\partial z}\, v \\ & + \mu_1 \int_{\Omega^s} \nabla w \cdot \nabla v + \mu_2 \int_{\Omega^s} y\, \nabla w \cdot \nabla v \ , \end{aligned} \tag{7.48}$$

with remainder

$$\langle \widehat{\mathcal{A}}(\mu)\, w, v\rangle = \theta \int_{\Omega^f} (U(\mu) - U_M(\mu))\, \frac{\partial w}{\partial z}\, v \ . \tag{7.49}$$

Clearly, we recover $\mathcal{A}(\mu) = \overline{\mathcal{A}}(\mu) + \widehat{\mathcal{A}}(\mu)$, as desired.

We now consider the scenarios — contexts — in which the decomposition (7.48)–(7.49) might be relevant. These scenarios are not just pertinent to our particular problem; however, our problem will serve as a concrete vehicle with which to motivate the scenarios. Three scenarios relate to efficient approximation and associated error control:

1. The convection velocity, $U(\mu)$, is known, but the dependence on the parameter μ is not affine. We may thus choose to approximate $U(\mu)$ by $\sum_{m=1}^{M} U_{Mm}(\mu)\xi_m$ — thereby rendering $\overline{\mathcal{A}}(\mu)$ affine (see Section 7.5.1.0); and to absorb the remainder, $U(\mu) - U_M(\mu)$, in $\widehat{\mathcal{A}}(\mu)$ — thereby "controlling" the associated error. (In this case, Option II is preferred for X_N.)

2. The (partial differential) equation governing the convection velocity is known, but there is no analytical expression available for $U(\mu)$. We may thus choose to calculate $U(\mu)$ by a (say) collateral reduced-basis approximation, $U_M(\mu) = \sum_{m=1}^{M} U_{Mm}(\mu)\xi_m$ — thereby permitting prediction of the temperature; and to absorb the remainder, $U(\mu) - U_M(\mu)$, in $\widehat{\mathcal{A}}(\mu)$ — thereby assessing the effect of the error in $U(\mu)$ on the error in the temperature. (Again, Option II is preferred for X_N.)

3. The convection velocity is specified (in a device) or measured (in a process), but only to within some known experimental error, $\mathcal{E}_M(\mu)$. We may thus choose to approximate $U(\mu)$ by the mean of the measurements ξ_m, $U_M(\mu) = \frac{1}{M}\sum_{m=1}^{M} \xi_m$ — thereby permitting *numerical* prediction of the temperature; and to absorb the remainder, $U(\mu) - U_M(\mu)$, in $\widehat{\mathcal{A}}(\mu)$ — thereby assessing the effect of experimental characterization errors on subsequent numerical results (and system performance). (Here, Option I is preferred.)

The fourth scenario relates to sensitivity, in which the model error is synthetic — in this case our method can be interpreted as rigorous "sensitivity differences":

4. The convection velocity $U(\mu)$ is known in detail, but certain aspects of the profile (e.g., high wavenumber components) are arguably unimportant. We may thus choose to propose

a dimension reduction, $U(\mu) \approx U_M(\mu) = \sum_{m=1}^{M} U_{Mm}(\mu)\,\xi_m$ — thereby permitting rapid evaluation and optimization; and to absorb the remainder, $U(\mu) - U_M(\mu)$, in $\widehat{\mathcal{A}}(\mu)$ — thereby confirming (or denying) our hypothesis through *a posteriori* error estimation. (Here, Option II is again preferred.)

In Section 7.5.3 we present numerical results for a model version of the second scenario above.

Truth Approximation

We first introduce a quasi-uniform, regular sequence of triangulations [68] of $]0, L_x[\times]0, L_y[$, $\mathcal{T}_{2d}^{h_{2d}}$, that honors the solid and fluid subdomains in the sense that $\partial\Omega_{2d}^f$ intersects no (open) triangle of $\mathcal{T}_{2d}^{h_{2d}}$; here h_{2d} is the diameter of the triangulation. We then take the tensor product of this two-dimensional (x, y) triangulation with a one-dimensional uniform segmentation in z to construct a triangular-prismatic elemental decomposition of Ω, $\mathcal{T}_{3d}^{h_{2d},h_z}$; here h_z is the length of the triangular prisms in z. The associated $\mathcal{N}$-dimensional linear finite element space is denoted $X^{\mathcal{N}}$.

The nodes of our (quasi-uniform, regular) triangulation $\mathcal{T}_{2d}^{h_{2d}}$ that reside in Ω_{2d}^f (*not* including $\partial\Omega_{2d}^f$) engender a linear finite element approximation space $(X_{2d}^f)^{\mathcal{N}_{2d}^f}$ of dimension $\mathcal{N}_{2d}^f$. We can also write $(X_{2d}^f)^{\mathcal{N}_{2d}^f} = \text{span}\,\{\varphi_1, \ldots, \varphi_{\mathcal{N}_{2d}^f}\}$, where φ_j is the nodal basis function associated with the j^{th} node, $\boldsymbol{x}_j = (x, y)_j$, of $\mathcal{T}_{2d}^{h_{2d}}$ in Ω_{2d}^f. For future reference we define the support of φ_j — a nodal patch of area $O(h_{2d}^2)$ — as $\mathcal{R}_j$; we show in Figure 7.2 a typical node/nodal patch of $\mathcal{T}_{2d}^{h_{2d}}$. We require now that the ξ_m, $1 \le m \le M$, of (7.46) reside in $(X_{2d}^f)^{\mathcal{N}_{2d}^f}$; and we take for our truth approximation to $U(\mu)$ an appropriate (i.e., $H_0^1(\Omega_{2d}^f)$-"equivalent") projection of the convection velocity $U(\mu)$ onto the space $(X_{2d}^f)^{\mathcal{N}_{2d}^f}$, $U^{\mathcal{N}_{2d}^f}(\mu)$. We then redefine our error estimate of (7.47) to read,

$$\|U^{\mathcal{N}_{2d}^f}(\mu) - U_M(\mu)\|_{L^\infty(\Omega_{2d}^f)} \le \mathcal{E}_M^{\mathcal{N}}(\mu)\ . \tag{7.50}$$

We suppose that $\|U(\mu) - U^{\mathcal{N}_{2d}^f}(\mu)\|_{L^\infty(\Omega_{2d}^f)} \to 0$, and hence $\mathcal{E}_M^{\mathcal{N}}(\mu) \to \mathcal{E}_M(\mu)$, as $\mathcal{N}$ (and hence $\mathcal{N}_{2d}^f$) $\to \infty$.

Our truth approximation is thus: Given $\mu \in \mathcal{D}^\mu$, find $s^{\mathcal{N}}(\mu) = \langle L, u^{\mathcal{N}}(\mu)\rangle$, where $u^{\mathcal{N}}(\mu) \in X^{\mathcal{N}}$ satisfies

$$\langle \mathcal{A}^{\mathcal{N}}(\mu)\, u^{\mathcal{N}}(\mu), v\rangle = \langle F, v\rangle, \quad \forall\, v \in X^{\mathcal{N}}\ ,$$

and

$$\begin{aligned}\langle \mathcal{A}^{\mathcal{N}}(\mu) w, v\rangle \equiv\ & \int_{\Omega^f} \nabla w \cdot \nabla v + \theta \int_{\Omega^f} U^{\mathcal{N}_{2d}^f}(\mu)\, \frac{\partial w}{\partial z}\, v \\ & + \mu_1 \int_{\Omega^s} \nabla w \cdot \nabla v + \mu_2 \int_{\Omega^s} y\, \nabla w \cdot \nabla v\ .\end{aligned}$$

We then write $\mathcal{A}^{\mathcal{N}}(\mu) = \overline{\mathcal{A}}^{\mathcal{N}}(\mu) + \widehat{\mathcal{A}}^{\mathcal{N}}(\mu)$, where $\overline{\mathcal{A}}^{\mathcal{N}}(\mu)$

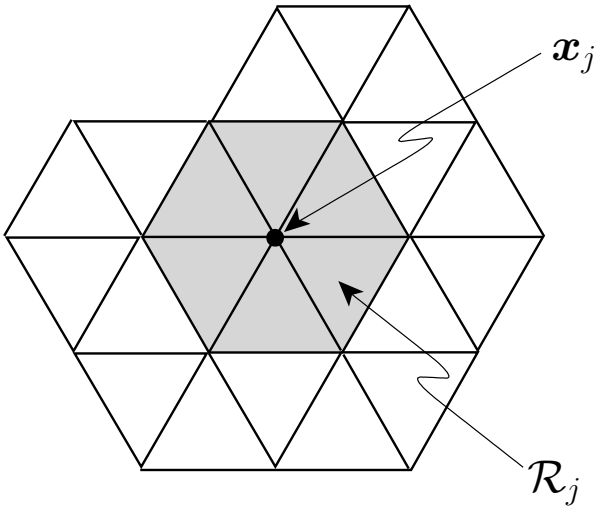


Figure 7.2: A node/nodal patch in $\mathcal{T}_{2d}^{h_{2d}}$.

$= \overline{\mathcal{A}}(\mu)$ (we commit no variational crimes), and

$$\langle \widehat{\mathcal{A}}^{\mathcal{N}}(\mu) w, v \rangle \equiv \theta \int_{\Omega} (U^{\mathcal{N}_{2d}^f}(\mu) - U_M(\mu)) \frac{\partial w}{\partial z} v \ .$$

Finally, we replace our subproblem space X_q associated with (7.41) with a corresponding truth-approximation space $(X_q)^{\mathcal{N}_q}$ (of dimension $\mathcal{N}_q = O(\frac{L_z}{h_z})$, independent of h_{2d}).

It should be patently clear that, as $\mathcal{N} \to \infty$, $u^{\mathcal{N}}(\mu) \to u(\mu)$, and $s^{\mathcal{N}}(\mu) \to s(\mu)$. The central issue is stability of our *a posteriori* error bounds, in particular of $\hat{\delta}_N(\mu)$, as $\mathcal{N}$ tends to infinity.

Verification of Hypotheses

Affine Structure. We show here that $\overline{\mathcal{A}}^{\mathcal{N}}(\mu)$ and $\widehat{\mathcal{A}}^{\mathcal{N}}(\mu)$ admit the affine decompositions of (7.3) and (7.4), respectively.

For $\overline{\mathcal{A}}^{\mathcal{N}}(\mu)$, it is immediately clear that $\overline{Q} = 1 + P(=2) + M$, where $\alpha_1(\mu) = 1$, $\alpha_2(\mu) = \mu_1$, $\alpha_3(\mu) = \mu_2$, and $\alpha_q(\mu) = U_{M(q-3)}(\mu)$ for $q = 4, \ldots, M+3$; and that

$$\begin{aligned}
\langle A_1 \, w, v \rangle &= \int_{\Omega^f} \nabla w \cdot \nabla v \ , \\
\langle A_2 \, w, v \rangle &= \int_{\Omega^s} \nabla w \cdot \nabla v \ , \\
\langle A_3 \, w, v \rangle &= \int_{\Omega^s} y \, \nabla w \cdot \nabla v \ , \\
\langle A_q \, w, v \rangle &= \theta \int_{\Omega^s} \xi_{q-3} \frac{\partial w}{\partial z} v, \quad q = 4, \ldots, M+3 \ .
\end{aligned}$$

We thus verify (7.3).

For $\widehat{\mathcal{A}}^{\mathcal{N}}(\mu)$, we first note that $U^{\mathcal{N}^f_{2d}}(\mu) - U_M(\mu)$ may be expressed (for given M) as

$$U^{\mathcal{N}^f_{2d}}(\mu) - U_M(\mu) = \sum_{q=1}^{\mathcal{N}^f_{2d}} \beta_q(\mu)\, \varphi_q(x,y) \ .$$

It thus directly follows that $\widehat{Q} = \mathcal{N}^f_{2d}$; that $\beta_q(\mu)$ is the value of the error $U^{\mathcal{N}^f_{2d}}(\mu) - U_M(\mu)$ at node $\boldsymbol{x}_q = (x,y)_q$, and

$$\langle B_q w, v\rangle = \theta \int_{\Omega^f} \varphi_q(x,y)\, \frac{\partial w}{\partial z}\, v, \quad q = 1,\ldots,\widehat{Q} \ , \tag{7.51}$$

thus confirming (7.4); and that $D_q = \mathcal{R}_q \times]0, L_z[$ — small prisms, or "pencils." Note that, as $\mathcal{N} \to \infty$, $\mathcal{N}^f_{2d} \to \infty$ ($h_{2d} \to 0$), and hence $\widehat{Q} \to \infty$.

Stability and Continuity. Under the assumptions that $\mu_1 > 0$, $\mu_2 \geq 0$, and $U(x,y) \geq 0$, $U_M(x,y) \geq 0$, $\forall (x,y) \in \Omega^f_{2d}$, it is standard to show that $\mathcal{A}$ and $\overline{\mathcal{A}}$ are coercive and continuous over X; the proof of coercivity is contained within Proposition 14. It remains to verify (7.7) and (7.9).

As regards, (7.7), we know from the nodal interpretation of the $\beta_q(\mu)$ that we may simply take $\beta^{\max}_{\widehat{Q}}(\mu) = \mathcal{E}^{\mathcal{N}}_M(\mu)$ and $\hat{\beta}^{\max} = \sup_{\mu\in\mathcal{D}^\mu} \mathcal{E}_M(\mu)$. The existence of an $\mathcal{N}$- (and hence $\widehat{Q}$-) independent bound for the L^∞-norm of the error, $\mathcal{E}_M(\mu)$, must of course be verified on a case-by-case basis; see Section 7.5.3.0 for further discussion. As regards, (7.9), we provide

Proposition 13. *For B_q as defined in (7.51), and (for simplicity) $\|\cdot\|_X = \|\cdot\|_{H^1(\Omega)}$,*

$$\sup_{w\in X} \frac{\left(\sum_{q=1}^{\widehat{Q}} \gamma_q^2(w)\right)^{1/2}}{\|w\|_X^2} \leq \sqrt{3}\,\theta\, C_4 \ ,$$

where $C_4 = \sup_{q\in\{1,\ldots,\widehat{Q}\}} \left(\sup_{v\in H^1(D_q)} \frac{\|v\|_{H^1(D_q)}}{|||\, v\, |||_q} \right)$.

Proof. We note that

$$\begin{aligned}
\langle B_q w, v\rangle &= \theta \int_{D_q} \varphi_q(x,y)\, \frac{\partial w}{\partial z}\, v \\
&\leq \theta\, \|\varphi_q\|_{L^\infty(\mathcal{R}_q)}\, \|w\|_{H^1(D_q)}\, \|v\|_{H^1(D_q)} \\
&= \theta\, \|w\|_{H^1(D_q)}\, \|v\|_{H^1(D_q)} \\
&\leq \theta\, C_4\, \|w\|_{H^1(D_q)}\, |||\, v\, |||_q \ ,
\end{aligned}$$

from the Cauchy-Schwarz inequality, $\|\varphi_q\|_{L^\infty(\mathcal{R}_q)} = 1$ (for linear nodal basis functions), and the definition of C_4. (Note that C_4 must be finite from our assumption that $|||\cdot|||_q$ and $\|\cdot\|_{H^1(D_q)}$ are uniformly equivalent.)

It then follows that $\gamma_q(w) \le \theta\, C_4\, \|w\|_{H^1(D_q)}$, and hence

$$\begin{aligned}\sum_{q=1}^{\widehat{Q}} \gamma_q^2(w) &\le \theta^2\, C_4^2 \sum_{q=1}^{\widehat{Q}} \|w\|^2_{H^1(D_q)} \\ &\le 3\,\theta^2\, C_4^2\, \|w\|^2_{H^1(\Omega)}\ ,\end{aligned}$$

since each point of Ω appears in at most $C_{\rm mult} = 3$ regions D_q (each triangle of $\mathcal{T}_{2d}^{h_{2d}}$ has three nodes). This concludes the proof. □ □

Note it follows from Proposition 13 that we may take $\widehat{\Gamma}$ of (7.9) to be $\sqrt{3}\,\theta\, C_4$.

7.5.2 Reduced-Basis Output Bound

Here we must select our bound conditioner and, relatedly, our inner products and norms over the D_q. We first define

$$\begin{aligned}\langle \mathcal{C}^0 w, v\rangle \;\equiv\; &\int_{\Omega^f} \nabla w\cdot\nabla v + \mu_1^{\min}\int_{\Omega^s} \nabla w\cdot\nabla v \\ &+ \mu_2^{\min}\int_{\Omega^s} y\,\nabla w\cdot\nabla v, \quad \forall\, w, v\in X\ ,\end{aligned} \tag{7.52}$$

where $\mu_1^{\min} = \max_{\{t\in\mathbb{R}_+\,|\,\mathcal{D}^\mu\subset[t,\infty[\times\mathbb{R}\}} t$ and $\mu_2^{\min} = \max_{\{t\in\mathbb{R}_+\,|\,\mathcal{D}^\mu\subset\mathbb{R}\times[t,\infty[\}} t$. One possible choice for our bound conditioner — which we shall denote BC1 — is then $\mathcal{C}(\mu) = \mathcal{C} \equiv \mathcal{C}^0$. Relatedly, we choose

$$((w,v))_q = \int_{D_q} \nabla w\cdot\nabla v\ ,$$

and

$$|||v|||_q^2 = \int_{D_q} |\nabla v|^2\ ;$$

we may thus satisfy (7.21) with the choice $\rho'_\Sigma(\mu) = \sqrt{3}$. It can readily be shown that, as desired, BC1 verifies both the computational invertibility condition (7.23) (for $j = 1$, $\mathcal{J}(\mu) = 1$, and $\mathcal{C}_1 = \mathcal{C}$) and our spectral condition (7.20). However, BC1 and the associated inner product $((\,\cdot\,,\,\cdot\,))_q$ can, in other circumstances, prove problematic: well-posedness of the "pencil" problems (and relatedly, uniform equivalence of $|||\cdot|||_q$ and $\|\cdot\|_{H^1(D_q)}$) relies on the presence of the Dirichlet boundary at $z = 0$.

We thus introduce a second bound conditioner — denoted BC2 — which is stable even for all-Neumann pencils. We first introduce[†]

$$\lambda^{\min} = \min_{v\in X} \frac{\langle \mathcal{C}^0 v, v\rangle}{\int_\Omega v^2}\ , \tag{7.53}$$

[†] Of course, $\lambda^{\min}$ is strictly positive precisely because of the Dirichlet part of $\partial\Omega$; but, in general, for BC2 (unlike BC1), we no longer require that this Dirichlet part of the boundary intersect the D_q.

and then define $\mathcal{C}(\mu) = \mathcal{C}$ as

$$\langle \mathcal{C} w, v \rangle \equiv \frac{1}{2} \left(\langle \mathcal{C}^0 w, v \rangle + \lambda^{\min} \int_\Omega wv \right), \ \forall\, w, v \in X \ . \tag{7.54}$$

Relatedly, we choose

$$((w,v))_q \equiv \frac{1}{2} \left(\int_{D_q} \nabla w \cdot \nabla v + \lambda^{\min} \int_{D_q} wv \right), \ 1 \le q \le \widehat{Q} \ ,$$

and hence

$$|||\, v \,|||_q^2 \equiv \frac{1}{2} \left(\int_{D_q} |\nabla v|^2 + \lambda^{\min} \int_{D_q} v^2 \right), \ 1 \le q \le \widehat{Q} \ , \tag{7.55}$$

for our subdomain inner products and norms, respectively; we may thus satisfy (7.21) with the choice $\rho'_\Sigma(\mu) = \sqrt{3}$.

It is clear that our computational invertibility condition, (7.23), is satisfied: we may take $J = 1$, $\mathcal{J}(\mu) = 1$, and $\mathcal{C}_1 = \mathcal{C}$. Note that $\mathcal{C}$ (and hence $\mathcal{C}_1$) does not depend on μ; and, in particular, that $\lambda^{\min}$ (or, more precisely, our truth approximation to $\lambda^{\min}$, $(\lambda^{\min})^{\mathcal{N}}$) does not depend on μ, and hence may be computed off-line. It thus remains only to verify (7.20). We do so in

Proposition 14. *For $\mu_1^{\min}$ strictly positive, $\mu_2^{\min}$ non-negative, and $U(x,y;\mu)$ a non-negative function in $L^\infty(\Omega_{2d}^f)$,*

$$1 \le \frac{\langle \mathcal{A}^S(\mu) v, v \rangle}{\langle \mathcal{C} v, v \rangle} \le \rho, \quad \forall v \in X \ , \tag{7.56}$$

for ρ independent of μ and N. Here $\mathcal{C}$ is defined in (7.52)–(7.54).

Proof. We first consider the "left" inequality associated with the spectral condition (7.56). In particular, from (7.45) and the definition of X we obtain

$$\begin{aligned} \langle \mathcal{A}^S(\mu) v, v \rangle &= \langle \mathcal{A}(\mu) v, v \rangle \\ &= \int_{\Omega^f} |\nabla v|^2 + \frac{\theta}{2} \int_{\substack{z = L_z \\ (x,y) \in \Omega_{2d}^f}} U(x,y;\mu)\, v^2 \\ &\quad + \mu_1 \int_{\Omega^s} |\nabla v|^2 + \mu_2 \int_{\Omega^s} y\, |\nabla v|^2 \ ; \end{aligned} \tag{7.57}$$

coercivity of $\mathcal{A}(\mu)$ then directly follows from our hypotheses on μ and $U(x,y)$. Furthermore, we immediately observe from (7.52) that $\langle \mathcal{C}^0 v, v \rangle \le \langle \mathcal{A}^S(\mu) v, v \rangle$, $\forall\, v \in X$; and since from (7.53) $\lambda_{\min} \int_\Omega v^2 \le \langle \mathcal{C}^0 v, v \rangle$, $\forall\, v \in X$, it follows that

$$\begin{aligned} \langle \mathcal{C} v, v \rangle &= \frac{1}{2} \langle \mathcal{C}^0 v, v \rangle + \frac{\lambda_{\min}}{2} \int_\Omega v^2 \\ &\le \langle \mathcal{C}^0 v, v \rangle \\ &\le \langle \mathcal{A}^S(\mu) v, v \rangle, \quad \forall\, v \in X \ , \end{aligned}$$

thus verifying the left inequality of (7.56).[†]

We now consider the right inequality of (7.56). We first recall from the Trace Theorem [68] and Poincaré-Friedrichs inequality that

$$\int_{\substack{z=L_z \\ (x,y)\in\Omega^f_{2d}}} U(x,y;\mu)\, v^2$$

$$\leq C_{\rm tr}\, \|U(\mu)\|_{L^\infty(\Omega^f_{2d})} \int_{\Omega^f} |\nabla v|^2, \quad \forall\, v \in X^f \ ;$$

for some constant $C_{\rm tr}$ that depends only on Ω^f. The desired result then follows with (crudely)

$$\rho = \frac{\max\limits_{\mu\in\mathcal{D}^\mu} \max(1,\, C_{\rm tr}\, \frac{\theta}{2}\, \|U(\mu)\|_{L^\infty(\Omega^f_{2d})},\, \mu_1,\, L_y\mu_2)}{\frac{1}{2}\, \min(1,\, \mu_1^{\rm min})} \ ;$$

ρ is bounded thanks to our hypotheses on $\mathcal{D}^\mu$. □ □

We note that our bound conditioner can be improved, for example, by considering the multi-point extensions described in [82].

The bound procedure then follows the outline of Section 7.4.2.0. We assume that, given a new μ in the on-line stage, $\hat{\rho}'_\Sigma(\mu)$ $(= \sqrt{3})$, $\underline{U}_M(\mu)$, and $\beta^{\rm max}_{\widehat{Q}}(\mu) = \mathcal{E}^{\mathcal{N}}_M(\mu)$ can be *rapidly* calculated — as rapidly as $\overline{s}_N(\mu)$ and $\Delta_N(\mu)$. This will be the case, for example, in Scenario 2 of Section 7.5.1.0, in which $\underline{U}_M(\mu)$ and $\mathcal{E}^{\mathcal{N}}_M(\mu)$ are the result of a collateral reduced-basis approximation (see also Section 7.5.3.0).

7.5.3 Numerical Study

Particular Examples

We consider a particularly simple case of Scenario 2 of Section 7.5.1.0. We assume that the flow is laminar and fully-developed. The exact velocity field $U \in X^f_{2d}$ thus satisfies a Poisson problem,

$$\int_{\Omega^f_{2d}} \nabla U \cdot \nabla v = \int_{\Omega^f_{2d}} v, \quad \forall\, v \in X^f_{2d} \ ;$$

and the truth approximation $U^{\mathcal{N}^f_{2d}} \in (X^f_{2d})^{\mathcal{N}^f_{2d}} \subset X^f_{2d}$ then satisfies

$$\int_{\Omega^f_{2d}} \nabla U^{\mathcal{N}^f_{2d}} \cdot \nabla v = \int_{\Omega^f_{2d}} v, \quad \forall\, v \in (X^f_{2d})^{\mathcal{N}^f_{2d}} \ .$$

We observe that $U^{\mathcal{N}^f_{2d}}$ does not depend on μ, and hence the reduced-basis approximation ($M = 1$, $U_{M1} = 1$, $\xi_1 = U^{\mathcal{N}^f_{2d}}$) will be exact, and will *not* exercise the model truncation terms

[†] In this case, since the errors $\beta_q(\mu)$ multiply lower-order derivatives in the "right" way, they do not contribute to (in)stability; more generally, for example, if the $\beta_q(\mu)$ multiply the higher derivatives, $\mathcal{C}(\mu)$ may need to reflect $\beta^{\rm max}_{\widehat{Q}}(\mu)$ in order to satisfy (7.20).

that are the focus of this paper. In order to introduce a non-zero model truncation error, we thus take $M = 1$, $U_{M1} = 1$, and $\xi_1 = U^{\mathcal{M}^f_{2d}}$, where $U^{\mathcal{M}^f_{2d}}$ is a finite element approximation to $U(\mu)$ defined on a triangulation that is (significantly) *coarser* than the coarsest truth triangulation to be considered. As regards many of the critical points we wish to address here (for example, stability of $\hat{\delta}_N(\mu)$), our simple surrogate suffices. In Section 7.5.3.0 we briefly discuss the main new complication that arises when $U(\mu)$ *does* depend on μ: on-line calculation of $\mathcal{E}^{\mathcal{N}}_M(\mu)$, our bound for the L^∞ error in $U(\mu)$.

In what follows, we consider $L_x = 10$, $L_y = 10$, $L_z = 10$, $t^w_x = 1$, $t^w_y = 1$, and $\theta = 1$. For these values (note $\|U\|_{L^\infty(\Omega^f_{2d})} \approx 4$), the thermal boundary layer is *just* developed at the channel exit ($z = L_z$), and thus convection effects are, indeed, important. (Our ultimate interest is high-Prandtl number hydrocarbons, and thus the assumption of fully-developed flow — but not fully-developed heat transfer — is not unreasonable.) We consider the parameter domain $\mathcal{D}^\mu = [.1, 10] \times [.1, 10]$, that is $.1 \le \mu_1, \mu_2 \le 10$.

We shall label our truth meshes by $[K_1, K_2]$, where K_1 is the number of nodes in $\mathcal{T}^{h_{2d}}_{2d}$ (most of which are in Ω^f_{2d}), and K_2 is the number of segments in z; hence the total degrees of freedom number $\mathcal{N} \approx K_1 K_2$. We shall consider two truth meshes: $\mathrm{Mesh_I} = [1969, 11]$, corresponding to $\mathcal{N}_\mathrm{I} = 21659$; and a refinement, $\mathrm{Mesh_{II}} = [7681, 11]$, corresponding to $\mathcal{N}_\mathrm{II} = 84491$. (Note that we refine only in (x, y) so as to provide the largest possible variation in $\widehat{Q}$; in actual practice, we would also refine in z.) The $\mathcal{M}^f_{2d} = 142$-node coarse mesh which serves to define $U^{\mathcal{M}^f_{2d}}$ is a de-refinement of the (x, y) triangulation associated with $\mathrm{Mesh_I}$. For $\mathrm{Mesh_I}$, $\mathcal{E}^{\mathcal{N}_\mathrm{I}}_M = 0.389$, whereas for $\mathrm{Mesh_{II}}$, $\mathcal{E}^{\mathcal{N}_\mathrm{II}}_M = 0.398$ (the slight increase of the latter relative to the former is largely an artifice of our surrogate — $\mathcal{E}^{\mathcal{N}}_M$ vanishes as $\mathcal{N}^f_{2d}$ approaches $\mathcal{M}^f_{2d}$); these errors are readily calculated off-line.

As regards our reduced-basis approximation, we first sample 30 points $(a, b)_i$ randomly from a bi-variate uniform probability distribution over $[\ln .1, \ln 10.] \times [\ln .1, \ln 10.]$; we next define $(\overline{a}, \overline{b})_i = (e^{a_i}, e^{b_i})$, $i = 1, \ldots, 30$. (The particular advantage of this log-random distribution is described in [46, 62], and will be demonstrated empirically below.) We then define $S_{30} = \{(\overline{a}, \overline{b})_1, \ldots, (\overline{a}, \overline{b})_{30}\}$, and subsequently form $S_1 \subset S_2 \subset \cdots \subset S_{30}$ as nested random subsamples (i.e., S_{29} is S_{30} with one point randomly removed, S_{28} is S_{29} with one point randomly removed, ...). Finally, these samples S_N, $N = 1, \ldots, 30$, induce our reduced-basis spaces X_N, $N = 1, \ldots, 30$; we elect Option I here simply because this choice more cleanly decouples the discretization and model truncation contributions to the error. The results below are (mostly) for BC2; similar results obtain for BC1.

Numerical Results

We present in Table 7.1, for $N = 5$ and $\mathrm{Mesh_I}$, $\overline{s}_N(\mu)$, $|s^{\mathcal{N}}(\mu) - \overline{s}_N(\mu)|$, $\Delta_N(\mu)$, and $\eta_N(\mu)$ for five different points $\mu \in \mathcal{D}^\mu$; the latter is hereafter referred to as the "test sample." We observe that $\overline{s}_N(\mu)$ does vary nontrivially over the parameter domain; that the error $|s^{\mathcal{N}}(\mu) - \overline{s}_N(\mu)|$ is already reasonably small at this very modest N; that $|s^{\mathcal{N}}(\mu) - \overline{s}_N(\mu)| \le \Delta_N(\mu)$ (and hence $\eta_N(\mu) \ge 1$) — we obtain strict bounds, as expected; but that, for the larger values of μ, the effectivities are much larger than desired — we grossly overestimate the error.

To better understand the undesirably large effectivites, we present in Table 7.2 the ratios

(μ_1, μ_2)	$\overline{s}_N(\mu)$	$\lvert s^{\mathcal{N}}(\mu) - \overline{s}_N(\mu)\rvert$	$\Delta_N(\mu)$	$\eta_N(\mu)$
(1.0e-01, 1.0e-01)	9.67e-01	6.85e-02	6.16e-01	8.98e+00
(1.0e-01, 1.0e+00)	5.92e-01	2.51e-02	4.93e-02	1.96e+00
(1.0e-01, 1.0e+01)	1.67e-01	1.93e-03	3.76e-02	1.94e+01
(5.0e+00, 5.0e+00)	2.45e-01	4.04e-03	1.12e-01	2.77e+01
(1.0e+01, 1.0e+01)	1.44e-01	8.08e-04	1.77e-01	2.19e+02

Table 7.1: Numerical results for an $N = 5$ reduced-basis approximation and the $\mathrm{Mesh}_{\mathrm{I}}$ truth approximation. Note that, for our particular output functional, bound conditioner, and $\mathrm{Mesh}_{\mathrm{I}}$ truth approximations, $|||L|||_{X'} = 0.312$.

(μ_1, μ_2)	$\dfrac{\lvert\lvert\lvert L\rvert\rvert\rvert_{X'}\overline{\delta}_N(\mu)}{\lvert\overline{s}^{\mathcal{N}}(\mu) - \overline{s}_N(\mu)\rvert}$	$\dfrac{\lvert\lvert\lvert L\rvert\rvert\rvert_{X'}\hat{\delta}_N(\mu)}{\lvert s^{\mathcal{N}}(\mu) - \overline{s}^{\mathcal{N}}(\mu)\rvert}$
(1.0e-01, 1.0e-01)	3.08e+01	1.58e+00
(1.0e-01, 1.0e+00)	6.84e+01	1.76e+00
(1.0e-01, 1.0e+01)	1.67e+02	4.74e+00
(5.0e+00, 5.0e+00)	2.48e+02	3.50e+00
(1.0e+01, 1.0e+01)	2.20e+02	5.62e+00

Table 7.2: Numerical results for the discretization and model truncation "effectivities" for an $N = 5$ reduced-basis approximation and the $\mathrm{Mesh}_{\mathrm{I}}$ truth approximation.

$|||L|||_{X'}\overline{\delta}_N(\mu)/|\overline{s}^{\mathcal{N}}(\mu) - \overline{s}_N(\mu)|$ and $|||L|||_{X'}\hat{\delta}_N(\mu)/|s^{\mathcal{N}}(\mu) - \overline{s}^{\mathcal{N}}(\mu)|$, which may be interpreted as the (reduced-basis) discretization "effectivity" and the model truncation "effectivity," respectively. The discretization effectivities can be quite large for the larger μ values. There are two causes: the *nonsymmetric* (convection) contributions to $\mathcal{A}(\mu)$ that are perforce not present in $\mathcal{C}(\mu)$; and the departure of μ from $\mu^{\min}$, the point at which the *symmetric* part of $\mathcal{A}(\mu)$ is best represented by $\mathcal{C}(\mu)$. The former are more difficult to mitigate; however, the latter — more significant here, judging from the μ-dependence of the discretization effectivity — can be readily treated by a multi-point bound conditioner [82]. In general, the model truncation effectivity is quite good — $O(10)$ at worst — perhaps due to more explicit incorporation of the convection continuity contributions.

We now investigate the effect of N. We consider the particular parameter point $\mu = (0.1, 0.1)$ so as to minimize the (remediable) "symmetric" contribution to the discretization effectivity. We present in Table 7.3 (for $\mu = (0.1, 0.1)$) $|s^{\mathcal{N}} - \overline{s}_N|$, $|||L|||_{X'}\overline{\delta}_N$, $|||L|||_{X'}\hat{\delta}_N$, Δ_N, and η_N, for $N = 5$, 10, 20, and 30. We observe that the model truncation contribution to the error bound, $|||L|||_{X'}\,\hat{\delta}_N$, is largely insensitive to N, as expected; and that the (reduced-basis) discretization contribution to the error bound, $|||L|||_{X'}\,\overline{\delta}_N$, decreases quite rapidly with N — such that for $N = 10$ the model truncation effect, $|||L|||_{X'}\,\hat{\delta}_N$, already dominates. For large N, the discretization contribution to the error

N	$\lvert s^{\mathcal{N}} - \overline{s}_N \rvert$	$\lvert\lvert\lvert L \rvert\rvert\rvert_{X'} \overline{\delta}_N$	$\lvert\lvert\lvert L \rvert\rvert\rvert_{X'} \hat{\delta}_N$	Δ_N	η_N
5	6.85e-02	5.35e-01	8.11e-02	6.16e-01	8.98e+00
10	4.39e-02	6.28e-02	8.01e-02	1.43e-01	3.25e+00
20	5.10e-02	7.54e-03	7.99e-02	8.75e-02	1.71e+00
30	5.12e-02	2.27e-05	8.00e-02	8.00e-02	1.56e+00

Table 7.3: Numerical results (for $\mu = (0.1, 0.1)$) for the error and error bound for $N = 5,\ 10,\ 20$, and 30 reduced-basis approximations and the $\mathrm{Mesh_I}$ truth approximation.

N	$\frac{\lvert\lvert\lvert L \rvert\rvert\rvert_{X'} \overline{\delta}_N}{\lvert \overline{s}^{\mathcal{N}} - \overline{s}_N \rvert}$	$\frac{\lvert\lvert\lvert L \rvert\rvert\rvert_{X'} \hat{\delta}_N}{\lvert s^{\mathcal{N}} - \overline{s}^{\mathcal{N}} \rvert}$
5	3.08e+01	1.58e+00
10	8.62e+00	1.56e+00
20	5.44e+01	1.56e+00
30	3.21e+01	1.56e+00

Table 7.4: Numerical results (for $\mu = (0.1, 0.1)$) for the discretization and model truncation effectivities for $N = 5,\ 10,\ 20$, and 30 reduced-basis approximations and the $\mathrm{Mesh_I}$ truth approximation.

is negligible.[†]

In an actual calculation we would exploit our *a posteriori* information to avoid this computationally wasteful discretization error–model truncation error imbalance (and associated stagnation of $\lvert s^{\mathcal{N}} - \overline{s}_N \rvert(\mu)$ and $\Delta_N(\mu)$). In particular, we would simultaneously improve both $\overline{s}_N$ (increase N) and U_M ("increase M") so as to maintain $\overline{\delta}_N(\mu) \approx \hat{\delta}_N(\mu)$; and we would terminate this double refinement when $\Delta_N(\mu)$ reaches the desired tolerance. Note the latter will yield an efficient approximation only if the individual (discretization and model truncation) effectivities are relatively small and, in particular, insensitive to N; this is, indeed, roughly the case, as we see in Table 7.4 — a summary (for $\mu = (0.1, 0.1)$) of $\lvert\lvert\lvert L \rvert\rvert\rvert_{X'} \overline{\delta}_N / \lvert \overline{s}^{\mathcal{N}} - \overline{s}_N \rvert$ and $\lvert\lvert\lvert L \rvert\rvert\rvert_{X'} \hat{\delta}_N / \lvert s^{\mathcal{N}} - \overline{s}^{\mathcal{N}} \rvert$ for $N = 5,\ 10,\ 20$, and 30.

We now consider the effect of the truth approximation resolution, $\mathcal{N}$, on $\hat{\delta}_N(\mu)$. In particular, we present in Table 7.5, again for our test sample, $\lvert\lvert\lvert L \rvert\rvert\rvert_{X'} \hat{\delta}_N(\mu) / \mathcal{E}_M^{\mathcal{N}}$ for the truth approximations $\mathrm{Mesh_I}$ and $\mathrm{Mesh_{II}}$. (Table 7.5 is for the case $N = 5$ — the results are quite insensitive to N — and for BC1.) We observe very little variation in $\lvert\lvert\lvert L \rvert\rvert\rvert_{X'} \hat{\delta}_N(\mu) / \mathcal{E}_M^{\mathcal{N}}$ as we increase $\mathcal{N}$ (and hence $\widehat{Q}$), consistent with our theoretical predictions. Table 7.5 provides empirical evidence that our method is indeed stable as $\widehat{Q}$ tends to infinity; we may thus choose $\mathcal{N}$ as large as deemed necessary to ensure adequate fidelity of the truth approximation.

As regards on-line complexity, we know that the on-line computational effort (and storage)

[†] Our convergence results here, and the refinement strategy proposed, reflect our "Option I" choice for X_N. Option II will yield slightly different convergence behavior, and hence require a slightly different refinement strategy.

(μ_1, μ_2)	$\frac{\|\|\|L\|\|\|_{X'}\hat{\delta}_N(\mu)}{\mathcal{E}_M^{\mathcal{N}_I}}$	$\frac{\|\|\|L\|\|\|_{X'}\hat{\delta}_N(\mu)}{\mathcal{E}_M^{\mathcal{N}_{II}}}$
(1.0e-01, 1.0e-01)	1.56e-01	1.70e-01
(1.0e-01, 1.0e+00)	8.52e-02	9.26e-02
(1.0e-01, 1.0e+01)	1.91e-02	2.05e-02
(5.0e+00, 5.0e+00)	2.99e-02	3.22e-02
(1.0e+01, 1.0e+01)	1.69e-02	1.82e-02

Table 7.5: Numerical results for the model truncation contribution to the error bound for an $N = 5$ reduced-basis approximation and the $\mathrm{Mesh_I}$ (second column) and $\mathrm{Mesh_{II}}$ (third column) truth approximations.

is also independent of $\mathcal{N}$ and $\widehat{Q}$. In particular, for $N = 10$, our on-line prediction of $\overline{s}_N(\mu)$ and $\Delta_N(\mu)$ is 1,000-fold (respectively, 10,000-fold) faster than the corresponding direct calculation of $s^{\mathcal{N}}(\mu)$ on $\mathrm{Mesh_I}$ (respectively, $\mathrm{Mesh_{II}}$). We may thus predict a *rigorous* and *tight* range for $s^{\mathcal{N}}(\mu)$, $\overline{s}_N(\mu) - \Delta_N(\mu) \le s^{\mathcal{N}}(\mu) \le \overline{s}_N(\mu) + \Delta_N(\mu)$, at a very small fraction of the cost of direct prediction of $s^{\mathcal{N}}(\mu)$; furthermore, our framework identifies the source of the "bound gap" so that adaptive refinement can be efficiently pursued.

The L^∞ Error Bound

As indicated earlier, with the introduction of geometric variations (e.g., in $t_\cdot^w$, $L_\cdot$), $U(\mu)$ *will* now depend on μ, and the reduced-basis approximation $U_M(\mu)$ will now be nontrivial — these calculations, which fully exercise all aspects of our framework, are reported elsewhere. We briefly discuss here the main new complication that arises: on-line calculation of $\mathcal{E}_M^{\mathcal{N}}(\mu)$, a bound for $\|U^{\mathcal{N}_{2d}^f}(\mu) - U_M(\mu)\|_{L^\infty(\Omega_{2d}^f)}$. (Note that, even in the case of geometry variations, Ω_{2d}^f — a *reference* domain — is independent of μ.)

To make our discussion sufficiently precise, we assume that $U(\mu) \in X_{2d}^f$ satisfies

$$\langle \mathcal{F}(\mu)\, U(\mu), v\rangle_{X_{2d}^f} = \langle 1, v\rangle_{X_{2d}^f}, \quad \forall\, v \in X_{2d}^f\ , \tag{7.58}$$

where $\langle \cdot, \cdot \rangle_{X_{2d}^f}$ is the duality pairing associated with X_{2d}^f, and $\mathcal{F}(\mu)\colon X_{2d}^f \to (X_{2d}^f)'$ is a (say) symmetric, coercive, continuous operator with affine parameter dependence. The corresponding truth approximation $U^{\mathcal{N}_{2d}^f}(\mu) \in (X_{2d}^f)^{\mathcal{N}_{2d}^f}$ satisfies (7.58) with (X_{2d}^f) replaced by $(X_{2d}^f)^{\mathcal{N}_{2d}^f}$. We now apply the techniques of Sections 7.3 and 7.4 to develop a reduced-basis approximation $U_M(\mu)$, and associated error bound in the energy norm, $\Delta_M^U(\mu)$, such that

$$\langle \mathcal{F}(U^{\mathcal{N}_{2d}^f}(\mu) - U_M(\mu)), U^{\mathcal{N}_{2d}^f}(\mu) - U_M(\mu)\rangle^{1/2} \le \Delta_M^U(\mu)\ . \tag{7.59}$$

We know that $U_M(\mu)$ and $\Delta_M^U(\mu)$ admit an efficient off-line/on-line computational procedure, and that the bound (7.59) will be relatively sharp.

Unfortunately, for $D \subset \mathbb{R}^2$, $H^1(D)$ is not continuously embedded in $L^\infty(D)$, and thus we can not hope to develop an L^∞ error bound of the form $\mathcal{E}_M^{\mathcal{N}}(\mu) = \overline{C}_{L^\infty,H^1}\, \Delta_M^U(\mu)$ for $\overline{C}_{L^\infty,H^1}$ independent of h_{2d} (and hence $\mathcal{N}$ and $\widehat{Q}$). Fortunately, for functions in $(X_{2d}^f)^{\mathcal{N}_{2d}^f}$, the inverse inequality is relatively benign for $D \subset \mathbb{R}^2$: it can be shown [83] that

$$C_{L^\infty,H^1}(h_{2d}) \leq \overline{C}\ \ln\left(\frac{1}{h_{2d}}\right)\ ,$$

where

$$C_{L^\infty,H^1}(h_{2d}) = \sup_{v\in(X_{2d}^f)^{\mathcal{N}_{2d}^f}} \frac{\|v\|_{L^\infty(\Omega_{2d}^f)}}{\langle \mathcal{F}(\mu)v, v\rangle^{1/2}}\ ,^{\dagger} \tag{7.60}$$

and $\overline{C}$ is independent of h_{2d}. We must thus violate one of our conditions: either $\hat{\beta}^{\max}$ (and hence $\hat{\delta}_N(\mu)$) will increase slightly with $\widehat{Q}$; or M (and hence $\overline{Q}$) must increase slightly with $\mathcal{N}$ (so as to decrease $\Delta_M^U(\mu)$ and hence maintain $\hat{\beta}^{\max}$ constant). But, the effect is only logarithmic, and thus presumably not cause for great concern. (We do not yet have numerical results for $C_{L^\infty,H^1}(h_{2d})$, and hence we can not report representative values for $\overline{C}$.)

However, for $D \subset \mathbb{R}^3$, the constant in the inverse inequality is worse than logarithmic. In this case, our approach will undoubtedly need to be modified. One possibility is to treat the summation over q in (7.16) differently: to work in a weighted ℓ^2, rather than ℓ^∞, norm for the $\beta_q(\mu)$ (related to an L^2 error estimate for $U_M(\mu)$), and in an ℓ^∞, rather than ℓ^2, norm for the γ_q; the latter can perhaps be bounded thanks to the greater regularity of $\overline{u}(\mu)$ (and $\overline{u}_N(\mu)$) [41]. Future work will explore this approach further, and develop associated off-line/on-line computational procedures.

Acknowledgements. We would like to thank Professor Yvon Maday of the University of Paris VI for a careful critique of an earlier approach; his analysis stimulated the revised and much-improved technique presented here. We also thank Ivan Oliveira, Dimitrios Rovas, and Karen Veroy for their careful reading of the manuscript and many helpful recommendations. This work was supported by DARPA and AFOSR under Grant F49620-01-1-0458 and by the Singapore-MIT Alliance.

[†] It is a simple matter to develop an off-line bound, $C_{L^\infty,H^1}^{\rm UB}$, for C_{L^∞,H^1}: we first introduce a (say) μ-independent bound conditioner for $\mathcal{F}$, $\mathcal{C}^{\mathcal{F}}$; and we then calculate

$$C_{L^\infty,H^1}^{\rm UB}(h_{2d}) = \max_{q\in\{1,\ldots,\widehat{Q}\}} \left(\sup_{v\in(X_{2d}^f)^{\mathcal{N}}} \frac{v(\boldsymbol{x}_q)}{\langle \mathcal{C}^{\mathcal{F}} v, v\rangle^{1/2}} \right)\ ,$$

where we recall that the $\boldsymbol{x}_q$ are the nodes of $\mathcal{T}_{2d}^{h_{2d}}$ that reside in Ω_{2d}^f.

Chapter 8

A Priori Convergence Of Multi-Dimensional Parametrized Reduced Basis

Authors: A. Buffa, Y.Maday, A.T. Patera, C. Prud'homme, G. Turinici.

8.1 Introduction

The purpose of this work is to prove a priori convergence of the reduced basis schemes for the situation when the parameters are multi-dimensional. In previous works [48, 49, 63] exponential convergence with respect to the number N of basis functions is proved for one dimensional parameters case and the proof there can be extended by tensorization to obtain behavior that scales as $e^{-N^{1/P}}$ where P is the dimension of the parameter space and N is the number of reduced basis functions. But the numerical experiments show much faster convergence [7] which leads us to inquire for different, much stronger estimations. Such a result is provided here which, under assumption (8.13) shows that exponential convergence is recovered as e^{-cN} and thus is independent of the dimension P of the parameter space.

Let us introduce the notations: $u(x,\mu) \in Y$ is the solution of a parametrically dependent partial differential equation (PDE) set on a spatial domain $\Omega \subset \mathbb{R}^d$ and on a parametric domain $D \subset \mathbb{R}^P$. Here $Y \subset L^2(\Omega)$ is a functional space adapted to the PDE, e.h. $Y = H_0^1(\Omega)$. We will suppose D to be compact, but we take no other hypothesis on Ω than those required by the PDE itself.

Our partial differential equation under weak form reads as follows: given $\mu \in D^\mu$, find $u(\mu) \in Y$ which satisfies

$$\mathcal{A}(u(\mu), v; \mu) = 0 \quad \forall v \in Y \tag{8.1}$$

where the form $\mathcal{A}$ encodes the description of the PDE.

For reasons that will become clear in Section 8.2 we will in fact suppose that we can define $u(x,\mu)$ for μ in some neighborhood D_r of D

$$D_r = \cup_{\mu \in D} B_\mu(r) = \{\mu \in \mathbb{R}^d | dist(\mu, D) \leq r\} \tag{8.2}$$

where $B_a(r)$ is the ball of center a and radius r. Note that since D is compact D_r will also be compact for any $r \geq 0$. In particular the **fixed constant** r can be as small as we want.

Remark 8.1.1. *This construction is not unique, for all that follows even D would have been enough provided it satisfies some additional geometric properties on which we do not want to elaborate here.*

The dependence of u on μ is also supposed smooth enough, but at least Lipschitz of constant L i.e.

$$\|u(x,\mu) - u(x,\mu')\|_Y \leq L\|\mu - \mu'\|_{\mathbb{R}^P}. \tag{8.3}$$

We will introduce the operators K and respectively T which are convolution with the kernels

$$K(\mu,\mu') = \int_\Omega u(x,\mu)u(x,\mu')dx \tag{8.4}$$

$$T(x,x') = \int_{D_r} u(x,\mu)u(x',\mu)d\mu \tag{8.5}$$

i.e. the operators that ct by

$$(Kf)(\mu) = \int_{D_r} K(\mu,\mu')f(\mu')d\mu' \tag{8.6}$$

$$(Tu)(x) = \int_\Omega T(x,x')f(x')dx' \tag{8.7}$$

It is immediate to show that both operators are self-adjoint. We also introduce the set W which is the L^2 closure of the set of all solutions $u(x,\mu)$ for $\mu \in D_r$.

$$W = \overline{\{u(x;\mu); \mu \in D_r\}} \tag{8.8}$$

We can prove the following properties concerning the operators T and K

- $T(x,x') \in L^2(\Omega\times\Omega)$ (this follows from the boundedness properties of the solutions $u(\mu)$

- $T(x,x')$ is positive and compact (from the Lebesgue theorem weak L^2 convergence of f_n to f implies strong L^2 convergence of Tf_n to Tf)

- $Ker(T) = W^\perp$ and $Ran(T) \subset W$

Thus T is a symmetric, positive definite compact operator from W to W. As such it admits a spectral decomposition in terms of its eigenfunctions $(\psi_k(x))_{k=1}^\infty$ corresponding to the eigenvalues $(\lambda_k)_{k=1}^\infty$:

$$Tf = \sum_{k=1}^\infty \lambda_k < f, \psi_k > \psi_k \tag{8.9}$$

Same considerations of compacteness hold for operator K too. Let us remark that

$$\phi_k(\mu) = \frac{1}{\sqrt{\lambda_k}} \int_\Omega u(x,\mu)\psi_k(x)dx$$

are eigenfunctions of the operator K which correspond to the same eigenvalues λ_k. Both $\phi_k(\mu)$, $\psi_k(x)$ are normalized in L^2. We have thus

$$K\phi_k = \lambda_k \phi_k \tag{8.10}$$
$$T\psi_k = \lambda_k \psi_k \tag{8.11}$$

Then the following summation formula holds:

$$u(x,\mu) = \sum_{k=1}^{\infty} \sqrt{\lambda_k}\phi_k(\mu)\psi_k(x). \tag{8.12}$$

It was numerically verified (see section 8.4) that for some parametric PDEs the following estimation is valid

$$\lambda_k \leq ce^{-f(k)}, \tag{8.13}$$

where the function f is at least linear $f(x) \geq \rho k$ for some $\rho > 0$.

Among others (see Section 8.2), eqns. (8.12) and (8.13) imply the existence of a space of small dimension that contains

$$U = \{u(\cdot,\mu) | \mu \in D\}. \tag{8.14}$$

Remark 8.1.2. *Note that in the definition of U we only consider parameters in D. The convergence proof will in fact only work for them and not for parameters in D_r.*

Because μ belongs to a compact space and u comes from a PDE we can suppose that each $u \in U$ is bounded (in L^2_x) by some constant $M > 0$.

8.2 Low dimensional manifold

The purpose of this section is to show that U is close to a linear space of small dimension Ψ_k spanned by first functions $\psi_1, ..., \psi_n$:

$$\Psi_k = \text{span}\{\psi_1, \ldots, \psi_k\} \tag{8.15}$$

Under this assumption, we prove that Ψ_k is exponentially approximating. However, due to some technicalities (basically we want a L^∞ norm but we only have a L^2 norm) the proof is not completely trivial.

Lemma 6. *Define $l_m = \sqrt{\sum_{k=m+1}^{\infty} \lambda_k}$. Then:*

$$\forall\, u \in U \ \|u - \Pi_{L^2}^{\Psi_k} u\|_{L^2} \leq C_k^{1/P}. \tag{8.16}$$

where $\Pi_{L^2}^{\Psi_k} u$ is the L^2 projection of u on Ψ_k i.e.

$$\Pi_{L^2}^{\Psi_k} u = \text{arginf}_{\psi \in \Psi_k} \|u - \psi\|_{L^2}.$$

Proof. Denote for $\mu \in D_r$

$$f_m(\mu) = \|u(x,\mu) - \sum_{k=1}^{m} \sqrt{\lambda_k}\phi_k(\mu)\psi_k(x)\|_{L^2_x} \tag{8.17}$$

It is immediate to see that

$$\|f_m(\mu)\|_{L^2_\mu(D_r)} \leq l_m \tag{8.18}$$

and thus, the measure of the set of μ where $f_m(\mu)$ is large can be bounded:

$$\text{meas}\{\mu \in D_r | f_m(\mu) > l_m^{1/2}\} \leq l_m \tag{8.19}$$

Let $\mu \in D$ be such that $f_m(\mu) > l_m^{1/2}$. Denote by z the distance from this point μ to the closest point μ' with $f_m(\mu') \leq l_m^{1/2}$. Then, $B_z(\mu) \subset \{\mu \in D_r | f_m(\mu) > l_m^{1/2}\}$ and thus, by passing to measures: $z^P \leq c l_m$ (c is a constant depending on the dimension P). Thus, there exists a point μ' at distance $z \leq c l_m^{1/P}$ with $f_m(\mu') \leq l_m^{1/2}$. It follows from the Lipschitz property (8.3) that

$$\begin{aligned} &\|u(x,\mu) - \sum_{k=1}^{m} \sqrt{\lambda_k}\phi_k(\mu')\psi_k(x)\|_{L^2_x} \leq \|u(x,\mu) - u(x,\mu')\| \\ &+\|u(x,\mu') - \sum_{k=1}^{m} \sqrt{\lambda_k}\phi_k(\mu')\psi_k(x)\|_{L^2_x} \leq c(l_m)^{1/P} + l_m^{1/2}, \end{aligned} \tag{8.20}$$

hence the conclusion. □

8.3 A priori convergence result

We give in this section details on the exponentially accurate approximation of U by Ψ_k According to hypothesis (8.13) we suppose thereafter that the function f in (8.13) is such that

$$\forall\, u \in U \ \|u - \Pi^{\Psi_k}_{L^2} u\|_{L^2} \leq C e^{-\alpha k} \tag{8.21}$$

with $\alpha > \ln(2)$. This amounts to asking that $\rho > 2P\ln(2)$.

We construct the following reduced basis:

$$\mu_1 = \text{argsup}_{\mu\in D}\|u(\cdot,\mu)\|_{L^2_x} \qquad u_1 = u_{\mu_1} = u(\cdot,\mu_1) \tag{8.22}$$

$$\mu_{i+1} = \text{argsup}_{\mu\in D}\|u(\cdot,\mu) - P_i u(\cdot,\mu)\|_{L^2_x} \ u_{i+1} = u(\cdot,\mu_{i+1}) - P_i u(\cdot,\mu_{i+1}) \tag{8.23}$$

where P_i is the orthogonal projection onto $V_i = \text{span}\{u_1,\ldots,u_i\}$. Note that:

1. The functions $\{u_i\}_{i\geq 1}$ are orthogonal, but not orthonormal.
2. The projection $P_i u(\cdot,\mu)$ verifies the following:

$$P_i u_\mu = \sum_{\ell=1}^{i} \alpha_\ell(\mu) u_\ell, \quad \alpha_\ell = \frac{\int_\Omega u_\mu u_\ell}{\|u_\ell\|^2}.$$

Indeed, $\alpha_\ell \|u_\ell\|^2 = \int_\Omega u_\ell(u_\mu - \sum_{m=1}^{\ell-1} \beta_m u_m)$, since $\int_\Omega u_\ell u_m = 0$, for all $m < \ell$. Thus:

$$\begin{aligned} \alpha_\ell &\leq \|u_\ell\|^{-1} \inf_{\{\beta_m\}} \|u_\mu - \sum_{m=1}^{\ell-1} \beta_m u_m\| \\ &\leq \|u_\ell\|^{-1} \|u_{\mu_\ell} - P_{\ell-1} u_{\mu_\ell}\| \leq 1 \end{aligned} \tag{8.24}$$

where in the last estimate we have used the very definition of μ_ℓ, and u_ℓ. Indeed, we may improve the next estimation by proving that $\exists \bar{\alpha}$ such that $\alpha_\ell \leq \bar{\alpha} < 1$, for all ℓ — in particular, one might weaken the assumption $\alpha \geq 1$.

Lemma 7. *For each u_ℓ, $\ell \geq 1$, there exists a $v_\ell \in \Psi_k$ such that:*

$$\|u_\ell - v_\ell\| \leq C 2^{\ell+1} e^{-\alpha k}. \tag{8.25}$$

Proof. Given u_μ we denote by v_μ the best fit which realizes (8.21). Note that, if we expand recursively $u_2\,,\ u_3\,,\ u_4 \ldots$ we obtain:

$$u_2 = u_{\mu_2} - \alpha_1^2 u_{\mu_1} \tag{8.26}$$
$$u_3 = u_{\mu_3} - \alpha_1^3 u_{\mu_1} - \alpha_2^3 (u_{\mu_2} - \alpha_1^2 u_{\mu_1}) \tag{8.27}$$
$$u_4 = u_{\mu_4} - \alpha_1^4 u_{\mu_1} - \alpha_2^4 (u_{\mu_2} - \alpha_1^2 u_{\mu_1}) - \alpha_3^4 (u_{\mu_3} - \alpha_1^3 u_{\mu_1} - \alpha_2^3 (u_{\mu_2} - \alpha_1^2 u_{\mu_1})) \tag{8.28}$$
$$u_5 = \ldots \tag{8.29}$$

We set $v_1 = v_{\mu_1}$, $v_2 = v_{\mu_2} - \alpha_1^2 v_{\mu_1} \ldots$ and so on. It is a matter of checking to see that, given u_ℓ, the term u_{μ_i}, $i < \ell$ is appearing $2^{\ell-i}$ times. Thus, choosing v_ℓ as just explained, making use of (8.24) (namely $\alpha_i^j \leq 1$), and triangle inequality, we obtain:

$$\|u_\ell - v_\ell\| \leq \sum_{i=1}^{\ell} 2^{\ell-i} \|u_{\mu_i} - v_{\mu_i}\| \leq 2^{\ell+1} e^{-\alpha k}.$$

□

Thus, take $k = n - 1$, this results is saying the reduced basis functions are exponentially approximated by the space Ψ_{n-1} in the following sense:

$$\exists \gamma > 0 \ \|u_\ell - v_\ell\| \leq C e^{-\gamma n} \quad \forall\, \ell \leq n.$$

Lemma 8.

$$\exists \beta > 0 \mid \forall\, u \in U \ \inf_{u_n \in V_n} \|u - u_n\| \leq C e^{-\beta n}. \tag{8.30}$$

Proof. Fixed n, let $v_1\,, \ldots\, v_n$ be the approximation of $u_1\,, \ldots, u_n$ in Ψ_{n-1}. Thus, there exist coefficents β_i, $\|\beta\| = 1$, such that $\sum_{i=1}^n \beta_i v_i = 0$. We then know, by means of Lemma 7, that

$$\|\sum_{i=1}^n \beta_i u_i\| = \|\sum_{i=1}^n \beta_i (u_i - v_i)\| \leq \sqrt{n} e^{-\gamma n}$$

We know that there exists a j such that $\beta_j > 1/n$. Thus,

$$\|u_j + \beta_j^{-1} \sum_{i<j} \beta_i u_i + \beta_j^{-1} \sum_{i>j} \beta_i u_i\| \leq C n^{3/2} e^{-\gamma n}.$$

Now, since the functions u_i are orthogonal, we obtain:

$$\|u_j\| \leq n^{3/2} e^{-\gamma n}.$$

Recalling the very definition of u_j, we have that, for all $\mu \in D$,

$$\|u_\mu - P_j(u_\mu)\| \leq \|u_j\| \leq C n^{3/2} e^{-\gamma n}.$$

Thus, the result is true. □

8.4 Eigenvalue decay

8.4.1 Preliminaries

The results of the previous section build on the assumption that the eigenvalues λ_k of the operators T and K decay at least linearly and with a slope greater than $2P\ln(2)$. We purpose of this section is to bring numerical evidence to support this hypothesis.

To test our hypothesis, we introduce the elliptic coercive bilinear form $a(\cdot,\cdot;\mu) : Y \times Y \times D^\mu \to \mathbb{R}$:

$$a(u, v; \mu) = \int_\Omega \nabla u \cdot \nabla v \; + \; g(x;\mu) u \; v, \quad \forall u, v \in Y \tag{8.31}$$

where $g : \Omega \times \mathbb{R} \to \mathbb{R}$.

We introduce the functional $f \in Y'$. Our partial differential equation under weak form reads as follows: Given $\mu \in D^\mu$, find $u(\mu) \in Y$ which satisfies

$$a(u(\mu), v; \mu) \; = \; f(v) \quad \forall v \in Y \tag{8.32}$$

In actual practice, we use a finite element approximation: Given $\mu \in D^\mu$, find $u^{\mathcal{N}}(\mu) \in Y^{\mathcal{N}} \subset Y$ which satisfies

$$a(u^{\mathcal{N}}(\mu), v; \mu) \; = \; f(v) \quad \forall v \in Y^{\mathcal{N}} \tag{8.33}$$

8.4.2 Reduced Basis Approximation

Following the methodology proposed in [64], we develop a reduced basis approximation to $u^{\mathcal{N}}(\mu)$, $u_N(\mu) \in W_N$, where

$$W_N \; = \; \text{span}\{\xi_n \equiv u(\mu^n), 1 \leq n \leq N\}, \tag{8.34}$$

$S_N = \{\mu^1 \in D^\mu, \ldots, \mu^N \in D^\mu\}$, and $u(\mu^n)$ satisfies (8.33). We have that

$$u_N(\mu) \; = \; \sum_{n=1}^{N} u_{N_n}(\mu) \xi_n \tag{8.35}$$

which is solution of the following equation using a standard Galerkin projection

$$a(u_N(\mu), v; \mu) \; = \; f(v) \quad \forall v \in W_N \tag{8.36}$$

8.4.3 Error Estimation

We used the error estimations proposed in [64] albeit slightly modified to provide online rapid error bounds of the reduced basis approximation with respect to the finite element approximation. As we shall see later, this was necessary for a number of results. We developed error bounds for $\|u_{\mathcal{N}}(\mu) - u_N(\mu)\|_Y$ which we shall denote $\Delta_N(\mu)$.

8.4.4 Construction and Eigenvalue Solves

In order to construct K, we generate a cartesian grid $\mathcal{G}^\mu$ of D^μ such that we have at least the extrema in each parameter direction. If L points are constructed in each direction of $\mathbb{R}^P$, we have defined $M = L^P$ points μ that are used to compute K entries. We shall denote $S_M(\mu) = \{\mu \in \mathcal{G}^\mu\}$.

8.4.5 Orthonormalization of W_N

Orthonormalisation is also for reduced basis approximation as it can be shown that the condition number of the reduced matrix is bounded for all $\mu \in D^\mu$. In particular, rounding errors due to ill conditionning are reduced.

If we orthonormalize the basis functions of W_N using the L_2 norm, we have that the entries of K take the following form

$$K^{L_2}(\mu^m, \mu^n) \;=\; K^{L_2}_{m,n} \;=\; \sum_{i=1}^{N} u_i(\mu^m)\; u_i(\mu^n), \quad \mu^m, \mu^n \in D^\mu \tag{8.37}$$

We can also orthonormalize with respect to the energy norm — at the minimum parameter value over D^μ.

8.4.6 Properties of the K operator

Let's define M as the dimension of K, as in section 8.4.4.

Since K is symmetric we use a QR symmetric algorithm to compute its eigenvalues.

Also note that if $M \leq N$ and $S_M(\mu) \subset S_N(\mu)$, K entries are formed by the L_2 scalar product of the basis functions of W_M, and if the basis functions of W_M are L_2 normalized then K is the identity matrix.

Proof. This is easily proven by using the fact that the components of the basis functions of W_M are the canonic vectors of $\mathbb{R}^M$. □

It follows that $\mathrm{rank}(K) \leq N$. See for example section 8.4.10.0 and figure 8.8 for numerical evidence.

Proof. First we consider the simple case where $M \leq N$. We make a change of reduced basis(without loss of generality) : we use $S_M(\mu)$ to construct W_M, we orthonormalize its basis functions and use the above property.

Now we turn to $M > N$. Again we make a change of reduced basis : we use $S_M(\mu)$ to construct W_M, we orthonormalize its basis functions — $\int_\Omega \xi_i\, \xi_j = \delta_{ij}$.

We have that

$$K_{mn} = \sum_{i=1}^{N}\sum_{j=1}^{N} u_i(\mu^m)\, u_j(\mu^n) \tag{8.38}$$

We note that for $m, n \leq N$, $u_i(\mu^m) = e_i^m$ and $u_j(\mu^n) = e_j^n$ where $e^k, k \leq M$ is the k-th canonical vector of $\mathbb{R}^M$.

Now K is made of four blocks as follows

$$K = \begin{pmatrix} (1)_{N,N} & (u_m^n)_{m=1,N;n=N+1,M} \\ (u_n^m)_{m=N+1,M;n=1,N} & (\sum_{i=1}^{N} u_i^m\ u_i^n)_{m=N+1,M;n=N+1,M} \end{pmatrix} \tag{8.39}$$

Using for example the Chio pivotal condensation to compute the determinant of K, we have immediately that $\det(K) = 0$ and $\operatorname{rank}(K) = N$. □

8.4.7 Higher Precision Approximations

When applying the reduced basis methodology in a standard way to compute the eigenvalues of K, we do observe very rapid decay of the eigenvalues. However we get only very few eigenvalues to check properly that the decay is indeed exponential.

To address this problem, we used a generic finite element framework [FIXME: CP/LIFEV] that allows to use numerical types with higher precision — more digits — albeit at a significant performance loss. We use coarser finite element approximations to alleviate the cost of using high order precision numerical types. Whatever computation related to approximation — finite element and reduced basis — and the eigenvalue solves must be done using high precision types, otherwise precision will "saturate" at double precision — Mesh node coordinates and parameter in D^μ need not use this particular numerical type. —

We have used the QD library [30] that provides quad-double precision type (approximately 64 decimal bits) particularly suited for use in C++ code.

8.4.8 Numerical Results

Remark 8.4.1. *On most figures that are shown in the following pages, we plot* $\lfloor \log_{10}(\lambda_k) \rfloor$ *where* $\lfloor \cdot \rfloor$ *is the floor function. The rationale is that we are only interested in the general decay behavior of* λ_k.

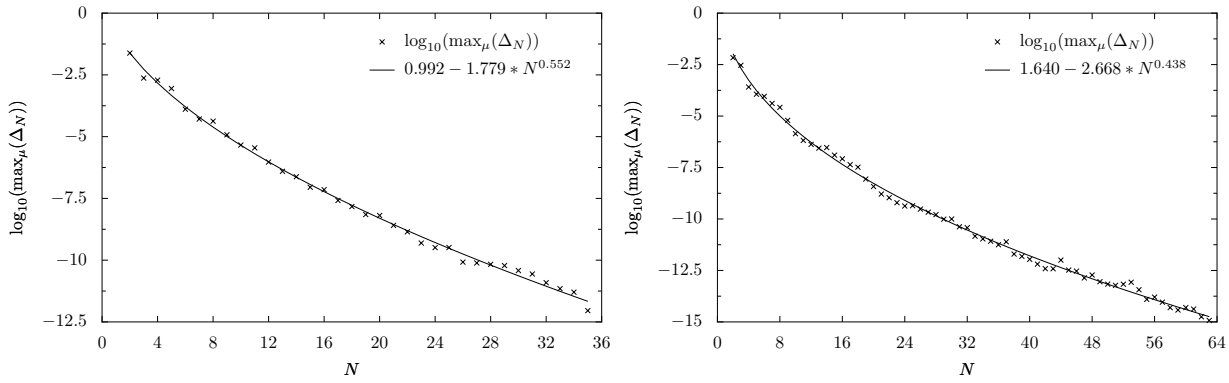
d]-$\Delta u + \mu(x)u = f$ on $[0.1]^d$

We conducted numerical tests for two different instances of the bilinear form a as in equation (8.31) using $g(x;\mu) = \mu_1 + \mu_2 x$ where $\mu = (\mu_1, \mu_2) \in D^\mu \subset [0.01; 100]^2$ (Case $P = 2$) and $g(x;\mu) = \mu_1 + \mu_2 x + \mu_3 xy$ where $\mu = (\mu_1, \mu_2, \mu_3) \in D^\mu \subset [0.01; 100]^3$ (Case $P = 3$). The geometrical domain is the same in all cases, $\Omega = [0; 1]^2$.

A function of the form $y = A - Bk^C$ is fitted to the data (λ_k) using a least square procedure to show the decayrate.

Reduced Basis Approximation

In all cases, a database is built offline thanks to the readily affine decomposition of $a(\cdot,\cdot;\mu)$ that will be used for all the results later on. The figures show the worst error bounds $\Delta_N(\mu)$ over a sample of 200 random parameters in D^μ for $P = 2$ and $P = 3$ respectively, see figure 8.1. We can observe the convergence of the worst error (behaves like $\exp(-N^{1/2})$).



(a) $P = 2$ (b) $P = 3$

Figure 8.1: Worst error over a sample of 200 random parameters for cases $P = 2$ and $P = 3$.

Case $P = 2$

In figure 8.2, we display $\lfloor \log_{10}(\lambda_k) \rfloor$. We have that $C > 1$ which shows that we have indeed exponential decay.

Case $P = 3$

In figure 8.3, we display $\lfloor \log_{10}(\lambda_k) \rfloor$. We have that $C > 1$ which shows that we have indeed exponential decay.

8.4.9 Remarks

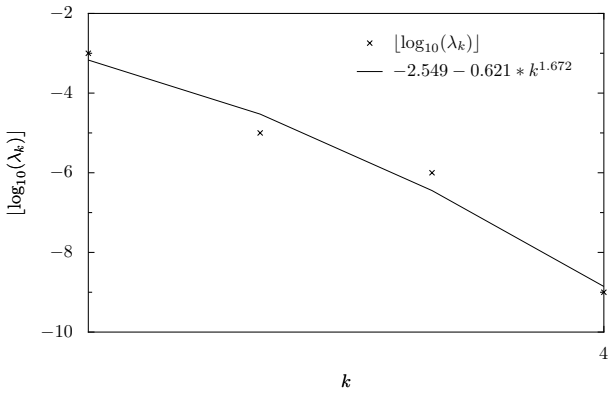
Orthornormalization of W_N

Rounding errors due to ill conditioning have been observed for the computations without orthonormalisation.

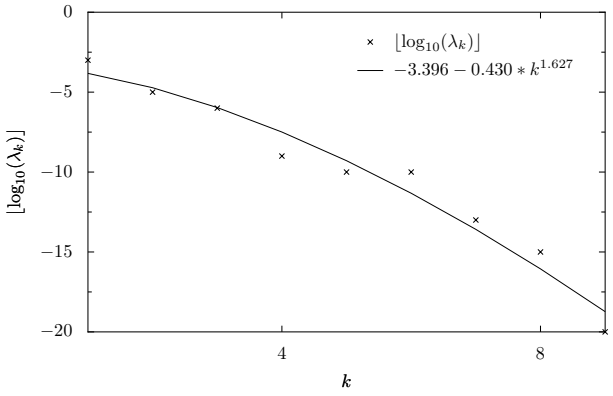
If we orthonormalize the basis functions of W_N using the L_2, we have that the entries of K take the following form

$$K^{L_2}(\mu^m, \mu^n) \;=\; K^{L_2}_{m,n} \;=\; \sum_{i=1}^{N} u_i(\mu^m)\; u_i(\mu^n) \tag{8.40}$$

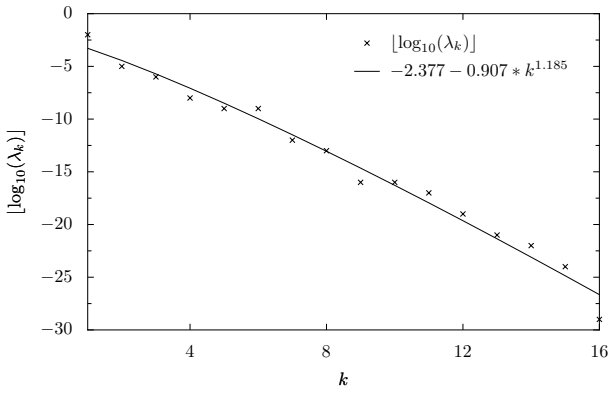
K^{L_2} has the same construction and has similar eigenvalues decay as K, see figure 8.4



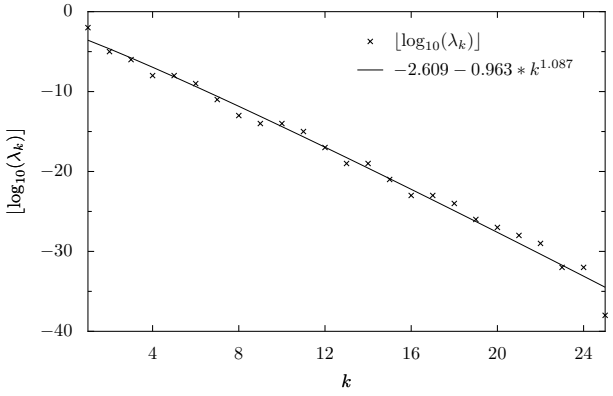
(a) $N = 4$



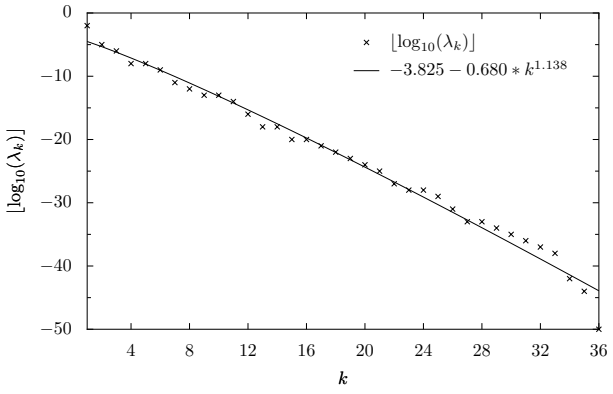
(b) $N = 9$



(c) $N = 16$

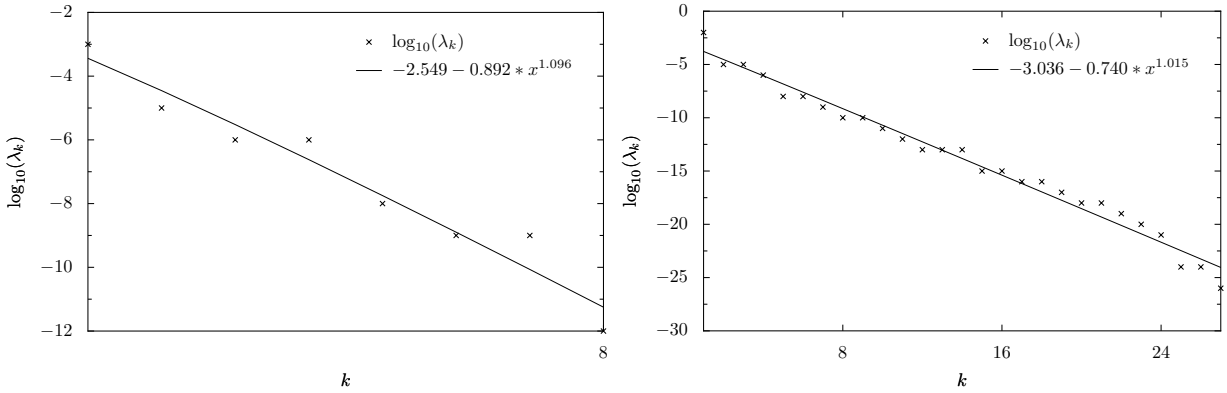


(d) $N = 25$



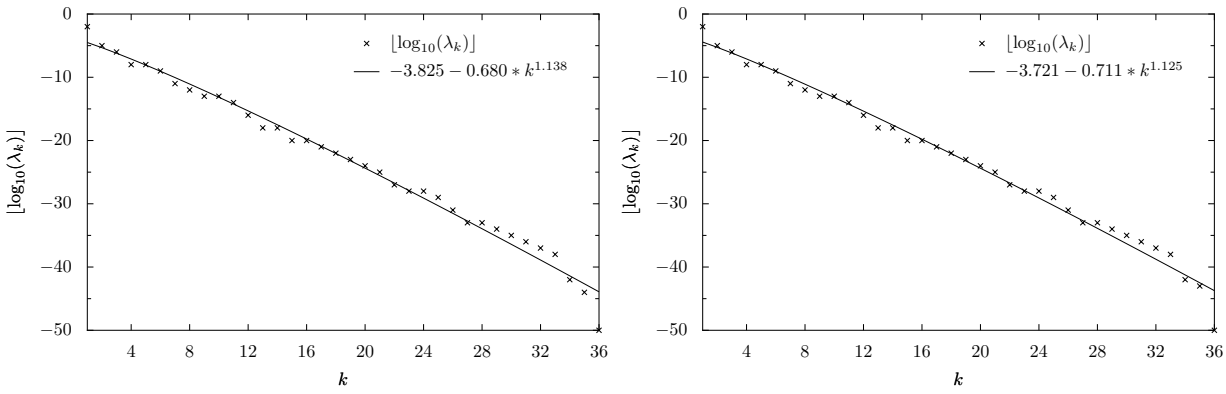
(e) $N = 36$

Figure 8.2: K eigenvalues decay for various reduced basis space dimension for $P = 2$



(a) $N = 8$ (b) $N = 27$

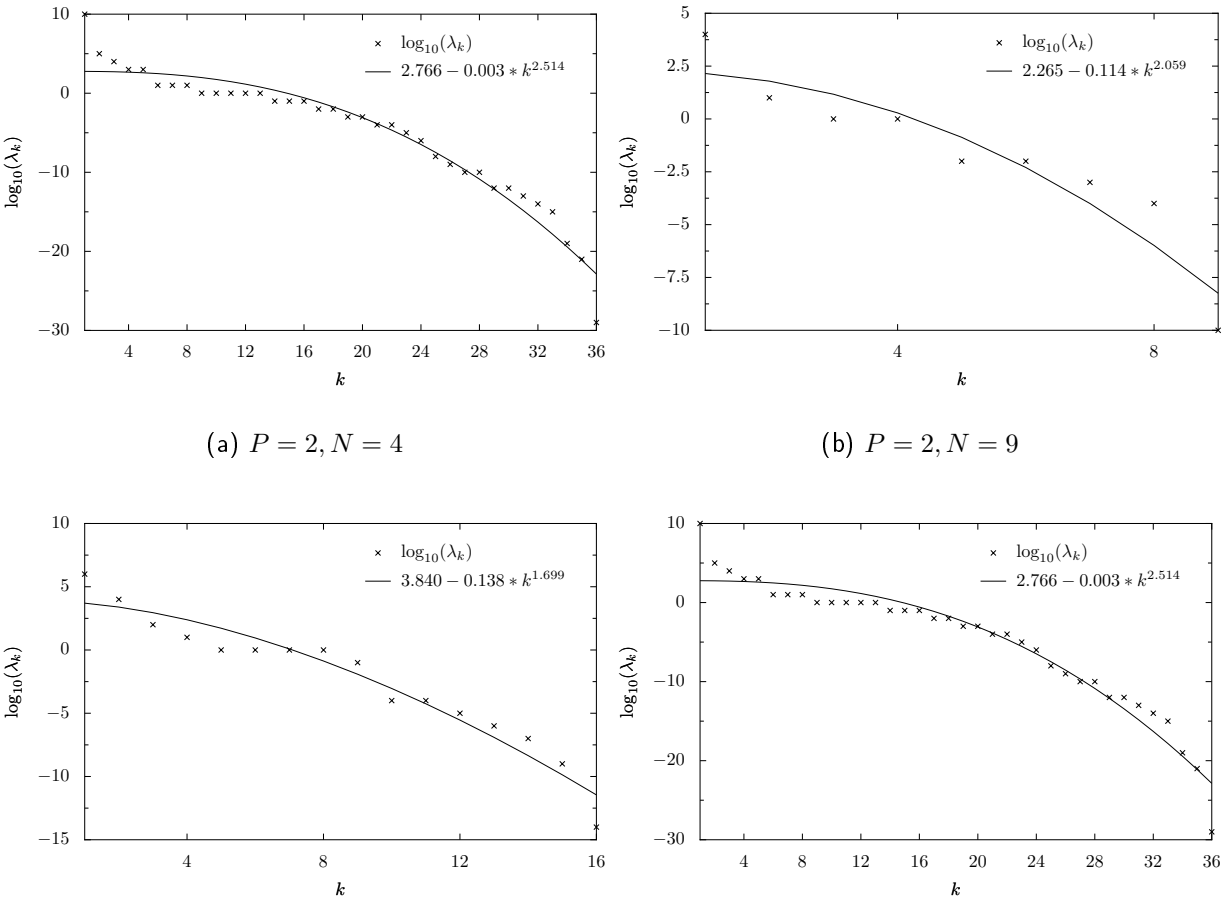
Figure 8.3: K eigenvalues decay for various reduced basis space dimension for $P = 3$



(a) $P = 2, N = 36$, no orthornormalization (b) $P = 2, N = 36$, L_2-orthornormalization

Figure 8.4: K eigenvalues decay with and without L_2 orthonormalization for test case $P = 2$

Without orthonormalization, then the eigenvalues of the matrix $(\sum_{i=1}^{N} u_i(\mu^m)\ u_i(\mu^n))_{m,n=1\ldots N}$ decay is as $\exp(-cN^2)$, see figure 8.5



(a) $P = 2, N = 4$

(b) $P = 2, N = 9$

(c) $P = 2, N = 16$

(d) $P = 2, N = 36$

Figure 8.5: Eigenvalues decay of $(\sum_{i=1}^{N} u_i(\mu^m)\ u_i(\mu^n))_{m,n=1\ldots N}$ with and without L_2 orthonormalization for test case $P = 2$

Some tests were also conducted using $a(\cdot,\cdot;\mu_{\min})$ orthonormalization. Results of the K eigenvalues decay are displayed in figure 8.6.

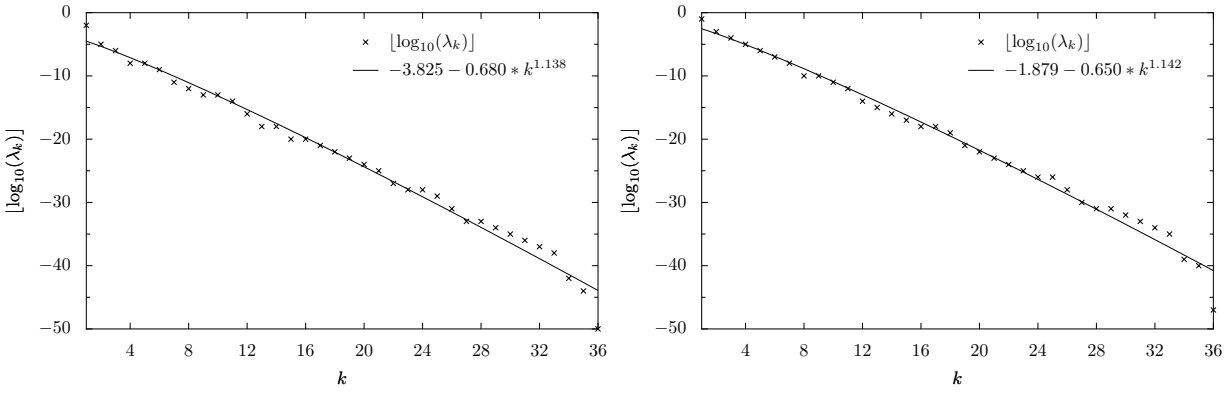
8.4.10 Fin example

We consider now a 2D fin with one stage as shown on figure 8.4.10

$$-\Delta u + \mu(x)u = f$$

First, we conducted numerical tests by defining the bilinear form a as in equation (8.31) using $g(x;\mu) = \mu_1 + \mu_2 x$ where $\mu = (\mu_1, \mu_2) \in D^\mu \subset [0.1; 10]^2$ (Case Fin $P = 2$)

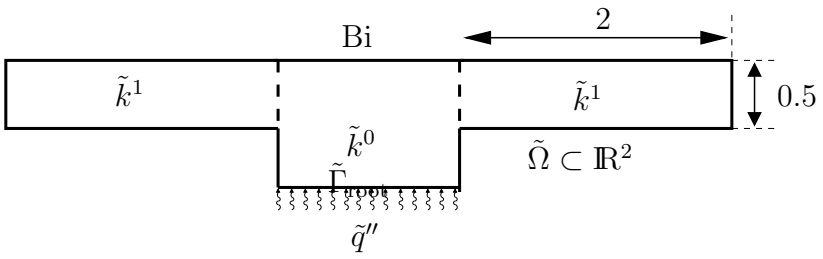
The results are presented on Figure 8.7. Again the assumption is confirmed.

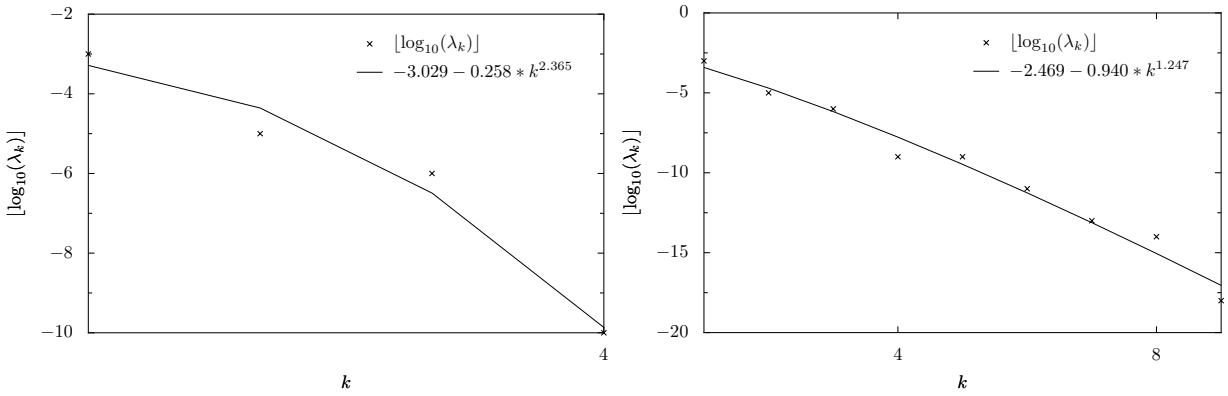


(a) $P = 2, N = 36$, no orthornormalization

(b) $P = 2, N = 36$, $a(\cdot, \cdot; \mu_{\min})$-orthornormalization

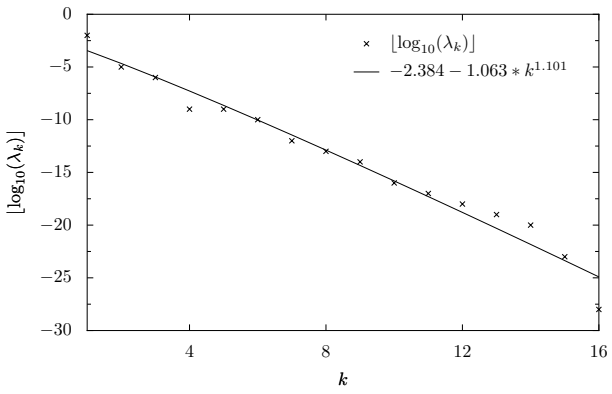
Figure 8.6: K eigenvalues decay with and without $a(\cdot, \cdot; \mu_{\min})$ orthonormalization for test case $P = 2$.





(a) Fin $P = 2, N = 4$, L_2-orthornormalization

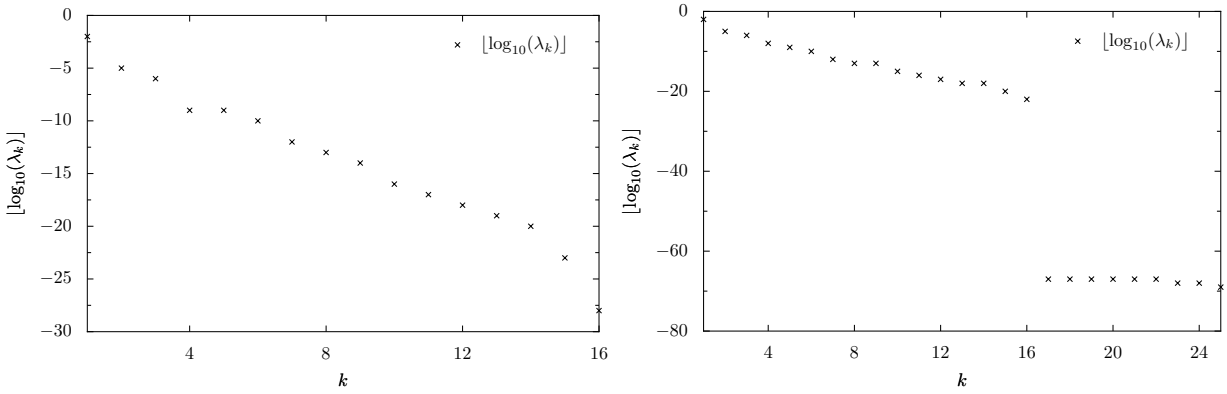
(b) Fin $P = 2, N = 9$, L_2-orthornormalization



(c) Fin $P = 2, N = 16$, L_2-orthornormalization

Figure 8.7: K eigenvalues decay for $-\Delta u + \mu(x)u = f$ on a fin geometry

We also verify that the matrix K has its rank $\leq N$, see figure 8.8.



(a) Fin $P = 2, N = 16$,nrows$(K) = 4$, L_2-orthornormalization

(b) Fin $P = 2, N = 16$,nrows$(K) = 5$, L_2-orthornormalization

Figure 8.8: K spectrum for $-\Delta u + \mu(x)u = f$ on a fin geometry with nrows$(K) \geq N$

Heat Transfer in a 2D Thermal Fin

Case $P = 2$ The fin consists of a vertical central "post" of conductivity $\tilde{k}_0$ and four horizontal "subfins" of conductivity $\tilde{k}^1$. The fins conduct heat from a prescribed uniform flux source $\tilde{q}''$ at the root $\tilde{\Gamma}_{\text{root}}$ through the post and large-surface-area subfins to the surrounding flowing air; the latter is characterized by a sink temperature $\tilde{u}_0$ and prescribed heat transfer coefficient $\tilde{h}$. The physical model is simple conduction: the temperature field in the fin, $\tilde{u}$, satisfies

$$\sum_{i=0}^{1} \int_{\tilde{\Omega}_i} \tilde{k}^i \, \tilde{\nabla}\tilde{u} \cdot \tilde{\nabla}\tilde{v} \; + \int_{\partial\tilde{\Omega}\setminus\tilde{\Gamma}_{\text{root}}} \tilde{h} \, (\tilde{u} - \tilde{u}_0) \, \tilde{v} \; = \int_{\tilde{\Gamma}_{\text{root}}} \tilde{q}'' \, \tilde{v}, \quad \forall \, \tilde{v} \in \tilde{X} \equiv H^1(\tilde{\Omega}), \quad (8.41)$$

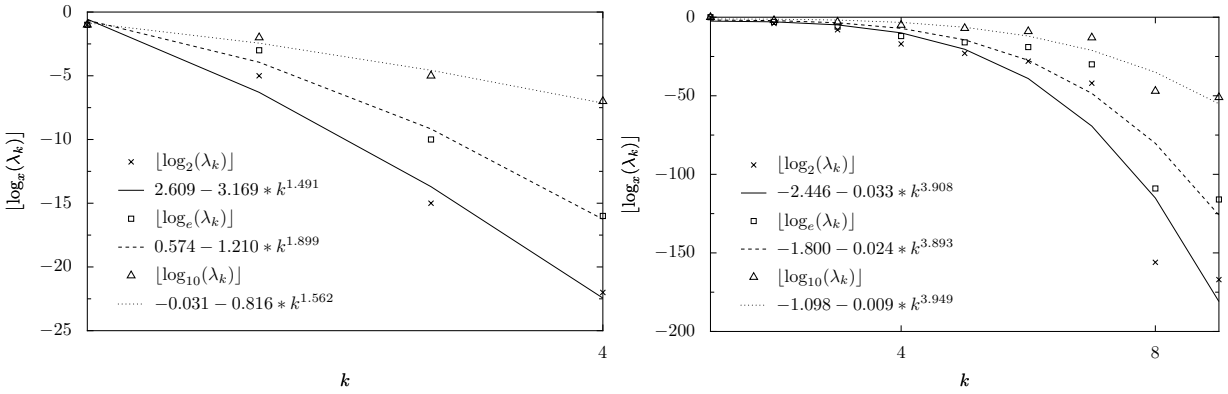
where $\tilde{\Omega}_i$ is that part of the domain with conductivity $\tilde{k}^i$, and $\partial\tilde{\Omega}$ denotes the boundary of $\tilde{\Omega}$.

We now (*i*) nondimensionalize the weak equations (8.41), and (*ii*) apply a continuous piecewise-affine transformation from $\tilde{\Omega}$ to a fixed (μ-independent) reference domain Ω [42]. The abstract problem statement (8.1) is then recovered for $\mu = \{k^1, \text{Bi}\}$, $D^\mu = [0.1, 10.0]^2$, and $P = 2$; here k^1 is the thermal conductivity of the "subfin" (see Figure 8.4.10) relative to the thermal conductivity of the fin base; Bi is a nondimensional form of the heat transfer coefficient; It is readily verified that a is continuous, coercive, and symmetric as required by the preliminaries in section 8.4.1;

The decay of the eigenvalues of K show on

8.4.11 Validation of the hypothesis

The previsous results showed that the hypothesis of the exponential decay with slope $\rho > 2Pln(2)$ is realistic, as in all cases superlinear decay has been observed. Although complete



(a) Fin $P = 2, N = 4$, L_2-orthornormalization

(b) Fin $P = 2, N = 9$, L_2-orthornormalization

Figure 8.9: K eigenvalues decay the 2D thermal fin

theoretical explanation of this property is not currently available, we expect the decay to depend on the nature of the PDE itself.

Part III

Publications: Scientific Computing and Technology

Chapter 9

A Mathematical and Computational Framework for Reliable Real-Time Solution of Parametrized Partial Differential Equations

Authors: C. Prud'homme, D. Rovas, K. Veroy and A.T. Patera

Abstract. We present in this article two components: these components can in fact serve various goals independently, though we consider them here as an ensemble. The first component is a technique for the *rapid and reliable evaluation* prediction of linear functional outputs of elliptic (and parabolic) partial differential equations with affine parameter dependence. The essential features are (*i*) (provably) rapidly convergent global reduced–basis approximations — Galerkin projection onto a space W_N spanned by solutions of the governing partial differential equation at N selected points in parameter space; (*ii*) *a posteriori* error estimation — relaxations of the error–residual equation that provide inexpensive yet sharp and rigorous bounds for the error in the outputs of interest; and (*iii*) off–line/on–line computational procedures — methods which decouple the generation and projection stages of the approximation process. This component is ideally suited — considering the operation count of the online stage — for the repeated and rapid evaluation required in the context of parameter estimation, design, optimization, and real–time control. The second component is a framework for distributed simulations. This framework comprises a library providing the necessary abstractions/concepts for distributed simulations and a small set of tools — namely SimTeX and SimLaB— allowing an easy manipulation of those simulations. While the library is the backbone of the framework and is therefore general, the various interfaces answer specific needs. We shall describe both components and present how they interact.

9.1 Introduction to Reduced Basis Output Bound Methods

The optimization, control, and characterization of an engineering component or system requires the prediction of certain "quantities of interest," or performance metrics, which we shall denote *outputs* — for example deflections, maximum stresses, maximum temperatures, heat transfer rates, flowrates, or lift and drags. These outputs are typically expressed as functionals of field variables associated with a parametrized partial differential equation which describes the physical behavior of the component or system. The parameters, which we shall denote *inputs*, serve to identify a particular "configuration" of the component: these inputs may represent design or decision variables, such as geometry — for example, in optimization studies; control variables, such as actuator power — for example in real–time applications; or characterization variables, such as physical properties — for example in inverse problems. We thus arrive at an implicit *input–output* relationship, evaluation of which demands solution of the underlying partial differential equation.

Our goal is the development of computational methods that permit *rapid* and *reliable* evaluation of this partial-differential-equation-induced input-output relationship *in the limit of many queries* — that is, in the design, optimization, control, and characterization contexts. The "many query" limit has certainly received considerable attention: from "fast loads" or multiple right-hand side notions (e.g., [21, 25]) to matrix perturbation theories (e.g., [3, 85]) to continuation methods (e.g., [4, 71]). Our particular approach is based upon the reduced–basis method, first introduced in the late 1970s for nonlinear structural analysis [5, 51], and subsequently developed more broadly in the 1980s and 1990s [14, 26, 55, 58, 70]. The reduced–basis method recognizes that the field variable is not, in fact, some arbitrary member of the infinite-dimensional space associated with the partial differential equation; rather, it resides, or "evolves," on a much lower–dimensional manifold induced by the parametric dependence.

The reduced–basis approach as earlier articulated is local in parameter space in both practice and theory. To wit, Lagrangian or Taylor approximation spaces for the low–dimensional manifold are typically defined relative to a particular parameter point; and the associated *a priori* convergence theory relies on asymptotic arguments in sufficiently small neighborhoods [26]. As a result, the computational improvements — relative to conventional (say) finite element approximation — are quite modest [58]. Our work differs from these earlier efforts in several important ways: first, we develop (in some cases, provably) *global* approximation spaces; second, we introduce rigorous *a posteriori error estimators*; and third, we exploit *off–line/on–line* computational decompositions. These three ingredients allow us — for a restricted but important class of problems — to reliably decouple the generation and projection stages of reduced–basis approximation, thereby effecting computational economies of several orders of magnitude.

In this expository review paper we focus on these new ingredients. We begin in Section 2 by introducing an abstract problem formulation and several illustrative instantiations. In Section 3 we describe the reduced–basis approximation for coercive symmetric problems and "compliant" outputs; associated *a posteriori* estimators are then developed in Section 4.

9.2 Problem Statement

9.2.1 Abstract Formulation

We consider a suitably regular domain $\Omega \subset \mathbb{R}^d$, $d = 1,\ 2,$ or 3, and associated function space $X \subset H^1(\Omega)$, where $H^1(\Omega) = \{v \in L^2(\Omega),\ \nabla v \in (L^2(\Omega))^d\}$, and $L^2(\Omega)$ is the space of square integrable functions over Ω. The inner product and norm associated with X are given by $(\,\cdot\,,\,\cdot\,)_X$ and $\|\cdot\|_X = (\,\cdot\,,\,\cdot\,)^{1/2}$, respectively. We also define a parameter set $\mathcal{D} \in \mathbb{R}^P$, a particular point in which will be denoted μ. Note that Ω does *not* depend on the parameter.

We then introduce a bilinear form $a\colon\ X \times X \times \mathcal{D} \to \mathbb{R}$, and linear forms $f\colon\ X \to \mathbb{R}$, $\ell\colon\ X \to \mathbb{R}$. We shall assume that a is continuous, $a(w,v;\mu) \le \gamma(\mu)\ \|w\|_X\ \|v\|_X \le \gamma_0\ \|w\|_X\ \|v\|_X,\ \forall \mu \in \mathcal{D}$; furthermore, in Sections 9.3 and 9.4, we assume that a is coercive,

$$0 < \alpha_0 \le\ \alpha(\mu) = \inf_{w \in X} \frac{a(w,w;\mu)}{\|w\|_X^2}, \quad \forall\, \mu \in \mathcal{D}, \tag{9.1}$$

and symmetric, $a(w,v;\mu) = a(v,w;\mu),\ \forall\ w,v \in X,\ \forall\ \mu \in \mathcal{D}$. We also require that our linear forms f and ℓ be bounded; in Sections 9.3 and 9.4 we additionally assume a "compliant" output, $f(v) = \ell(v),\ \forall\ v \in X$.

We shall also make certain assumptions on the parametric dependence of a, f, and ℓ. In particular, we shall suppose that, for some finite (preferably small) integer Q, a may be expressed as

$$a(w,v;\mu)\ =\ \sum_{q=1}^{Q} \sigma^q(\mu)\ a^q(w,v), \quad \forall\ w,v \in X,\ \forall\ \mu \in \mathcal{D}, \tag{9.2}$$

for some $\sigma^q\colon\ \mathcal{D} \to \mathbb{R}$ and $a^q\colon\ X \times X \to \mathbb{R}$, $q = 1,\ldots,Q$. This "separability," or "affine," assumption on the parameter dependence is crucial to computational efficiency; however, certain relaxations are possible — see [62]. For simplicity of exposition, we assume that f and ℓ do not depend on μ; in actual practice, affine dependence is readily admitted.

Our abstract problem statement is then: for any $\mu \in \mathcal{D}$, find $u(\mu) \in X$ such that

$$a(u(\mu),v;\mu)\ =\ f(v), \quad \forall\ v \in X; \tag{9.3}$$

and $s(\mu) \in \mathbb{R}$ given by

$$s(\mu) = \ell(u(\mu)). \tag{9.4}$$

In the language of the introduction, a is our partial differential equation (in weak form), μ is our parameter, $u(\mu)$ is our field variable, and $s(\mu)$ is our output.
For simplicity, we may suppress the μ-dependence along the article when there is no possible confusion.

9.2.2 Particular Instantiations

We indicate here a few instantiations of the abstract formulation; these will serve to illustrate the methods (for coercive, symmetric problems) of Sections 9.3 and 9.4.

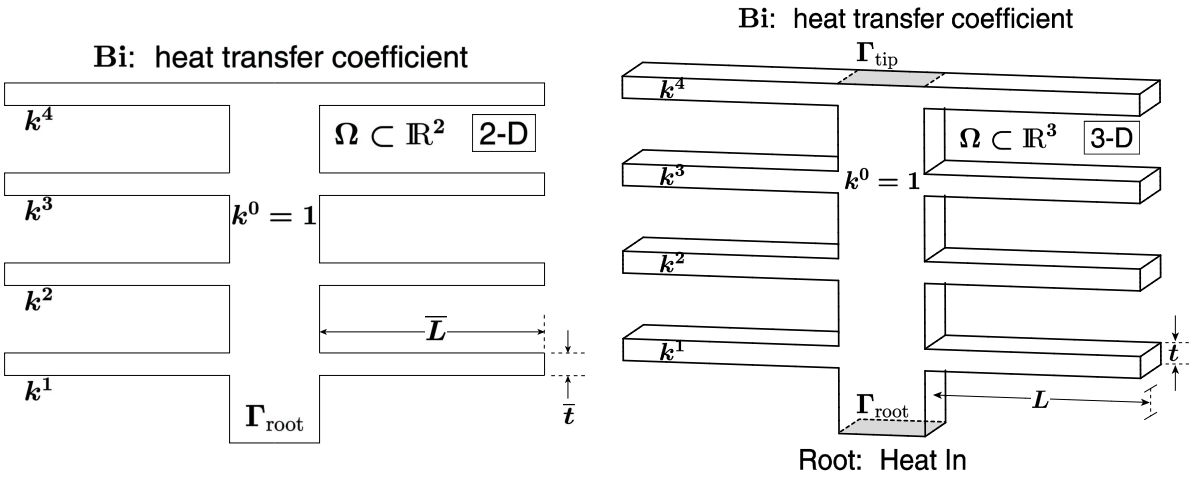


Figure 9.1: Two- and Three-Dimensional Thermal Fins.

A Thermal Fin

In this example we consider the two- and three-dimensional thermal fins shown in Figure 9.1; these examples may be (interactively) accessed on our web site[†]. The fins consist of a vertical central "post" of conductivity $\tilde{k}_0$ and four horizontal "subfins" of conductivity $\tilde{k}^i$, $i = 1, \ldots, 4$; the fins conduct heat from a prescribed uniform flux source, $\tilde{q}''$, at the root, $\tilde{\Gamma}_{\text{root}}$, through the post and large-surface-area subfins to the surrounding flowing air; the latter is characterized by a sink temperature $\tilde{u}_0$, and prescribed heat transfer coefficient $\tilde{h}$. The physical model is simple conduction: the temperature field in the fin, $\tilde{u}$, satisfies

$$\sum_{i=0}^{4} \int_{\tilde{\Omega}_i} \tilde{k}^i \, \tilde{\nabla}\tilde{u} \cdot \tilde{\nabla}\tilde{v} \; + \int_{\partial\tilde{\Omega}\setminus\tilde{\Gamma}_{\text{root}}} \tilde{h} \, (\tilde{u} - \tilde{u}_0) \, \tilde{v} \; = \; \int_{\tilde{\Gamma}_{\text{root}}} \tilde{q}'' \, \tilde{v}, \quad \forall \, \tilde{v} \in \tilde{X} \equiv H^1(\tilde{\Omega}), \tag{9.5}$$

where $\tilde{\Omega}_i$ is that part of the domain with conductivity $\tilde{k}^i$, and $\partial\tilde{\Omega}$ denotes the boundary of $\tilde{\Omega}$.

We now (*i*) nondimensionalize the weak equations (9.5), and (*ii*) apply a continuous piecewise-affine transformation to map $\tilde{\Omega}$ to a fixed reference domain Ω [42]. The abstract problem statement (9.3) is then recovered [72] for $\mu = \{k^1, \; k^2, \; k^3, \; k^4, \; \text{Bi}, \; L, \; t\}$, $\mathcal{D} = [0.1, 10.0]^4 \times [0.01, 1.0] \times [2.0, 3.0] \times [0.1 \times 0.5]$, and $P = 7$; here $k^1, \ldots, k^4$ are the thermal conductivities of the "subfins" (see Figure 9.1) relative to the thermal conductivity of the fin base; Bi is a nondimensional form of the heat transfer coefficient; and, L, t are the length and thickness of each of the "subfins" relative to the length of the fin root $\tilde{\Gamma}_{\text{root}}$. It is readily verified that a is continuous, coercive, and symmetric; and that the "affine" assumption (9.2) obtains for $Q = 16$ (two-dimensional case) and $Q = 25$ (three-dimensional case). Note that the geometric variations are reflected, via the mapping, in the $\sigma^q(\mu)$.

For our output of interest, $s(\mu)$, we consider the average temperature of the root of the fin nondimensionalized relative to $\tilde{q}''$, $\tilde{k}^0$, and the length of the fin root. This output is calculated

[†] FIN3D: http://augustine.mit.edu/fin3d_1/fin3d_1.pdf and FIN2D: http://augustine.mit.edu/fin2d/fin2d.pdf

as $s(\mu) = \ell(u(\mu))$, where $\ell(v) = \int_{\Gamma_{\text{root}}} v$. It is readily shown that this output functional is bounded and also "compliant": $\ell(v) = f(v),\ \forall v \in X$.

A Truss Structure

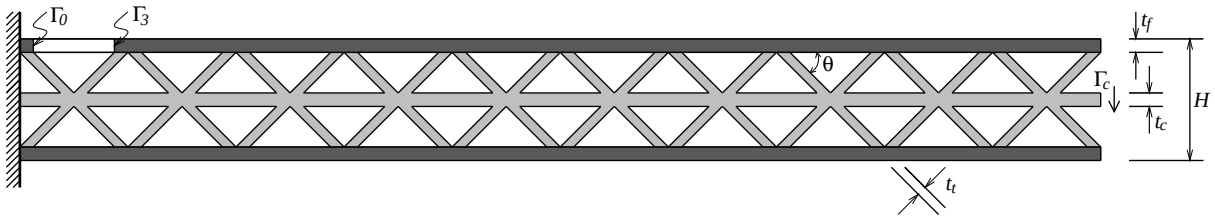


Figure 9.2: A Truss Structure

We consider a prismatic microtruss structure [24, 84] shown in Figure 9.2; this example may be (interactively) accessed on our web site[†]. The truss consists of a frame (upper and lower faces, in dark gray) and a core (trusses and middle sheet, in light gray); the structure transmits a force per unit depth $\tilde{F}$ uniformly distributed over the tip of the middle sheet, $\tilde{\Gamma}_3$, through the truss system to the fixed left wall, $\tilde{\Gamma}_0$. The physical model is simple plane–strain (two-dimensional) linear elasticity: the displacement field $u_i,\ i = 1, 2$, satisfies

$$\int_{\tilde{\Omega}} \frac{\partial \tilde{v}_i}{\partial \tilde{x}_j} \tilde{E}_{ijkl} \frac{\partial \tilde{u}_k}{\partial \tilde{x}_l} = -\left(\frac{\tilde{F}}{\tilde{t}_c}\right) \int_{\tilde{\Gamma}_3} \tilde{v}_2, \quad \forall\ v \in \tilde{X}, \tag{9.6}$$

where $\tilde{\Omega}$ is the truss domain, and $\tilde{X}$ refers to the set of functions in $H^1(\tilde{\Omega})$ which vanish on $\tilde{\Gamma}_0$. We assume summation over repeated indices.

We now (*i*) nondimensionalize the weak equations (9.6), and (*ii*) apply a continuous piecewise-affine transformation to map $\tilde{\Omega}$ to a fixed reference domain Ω. The abstract problem statement (9.3) is then recovered [78] for $\mu = \{t_f,\ t_t,\ H,\ \theta\}, \mathcal{D} = [0.08, 1.0] \times [0.2, 2.0] \times [4.0, 10.0] \times [30.0°, 60.0°]$, and $P = 4$; here t_f and t_t are the thicknesses of the frame and trusses, respectively; H is the total height of the microtruss; and θ is the angle between the trusses and the faces. The Poisson's ratio, $\nu = 0.3$, and the frame and core Young's moduli, $E_f = 75$ GPa and $E_c = 200$ GPa, respectively, are held fixed. It is readily verified that a is continuous, coercive, and symmetric; and that the "affine" assumption (9.2) obtains for $Q = 44$; Q is larger than for the fin examples due to the more complex (sheared) affine geometry mappings.

Our outputs of interest are (*i*) the average downward deflection (compliance) at the core tip, Γ_3, nondimensionalized by $\tilde{F}/\tilde{E}_f$; and (*ii*) the average normal stress across the critical (yield) section denoted Γ_1^s in Figure 9.2. These compliance and noncompliance outputs are written as $s^1(\mu) = \ell^1(u(\mu))$ and $s^2(\mu) = \ell^2(u(\mu))$, respectively, where $\ell^1(v) = -\int_{\Gamma_3} v_2$, and

$$\ell^2(v) = \frac{1}{t_f} \int_{\Omega^s} \frac{\partial \chi_i}{\partial x_j} E_{ijkl} \frac{\partial u_k}{\partial x_l}$$

[†] TRUSS: http://augustine.mit.edu/virg_1/virg_1.pdf

are bounded linear functionals; here χ_i is any suitably smooth function in $H^1(\Omega^s)$ such that $\chi_i \hat{n}_i = 1$ on Γ_1^s and $\chi_i \hat{n}_i = 0$ on Γ_2^s, where $\hat{n}$ is the unit normal.

9.3 Reduced-Basis Approach

We recall that in this section, as well as in Section 9.4, we assume that a is continuous, coercive, symmetric, and affine in μ — see (9.2); and that $\ell(v) = f(v)$, which we denote "compliance."

9.3.1 Reduced–Basis Approximation

We first introduce a sample in parameter space, $S_N = \{\mu_1, \ldots, \mu_N\}$, where $\mu_i \in \mathcal{D}$, $i = 1, \ldots, N$; see Section 3.2 for a brief discussion of point distribution. We then define our Lagrangian [58] reduced–basis approximation space as $W_N = \text{span}\{\zeta_n \equiv u(\mu_n),\ n = 1, \ldots, N\}$, where $u(\mu_n) \in X$ is the solution to (9.3) for $\mu = \mu_n$. In actual practice, $u(\mu_n)$ is replaced by a finite element approximation on a suitably fine truth mesh; we shall discuss the associated computational implications in Section 9.3.3. Our reduced–basis approximation is then: for any $\mu \in \mathcal{D}$, find $u_N(\mu) \in W_N$ such that

$$a(u_N(\mu), v; \mu) \;=\; \ell(v), \quad \forall\ v \in W_N; \tag{9.7}$$

we then evaluate $s_N(\mu) = \ell(u_N(\mu))$. (Non-Galerkin projections are briefly described in [62].)

9.3.2 *A Priori* Convergence Theory

Optimality

We consider here the convergence rate of $u_N(\mu) \to u(\mu)$ and $s_N(\mu) \to s(\mu)$ as $N \to \infty$. To begin, it is standard to demonstrate optimality of $u_N(\mu)$ in the sense that

$$\|u(\mu) - u_N(\mu)\|_X \le \sqrt{\frac{\gamma(\mu)}{\alpha(\mu)}} \inf_{w_N \in W_N} \|u(\mu) - w_N\|_X \ . \tag{9.8}$$

(We note that, in the coercive case, stability of our ("conforming") discrete approximation is not an issue; the noncoercive case is decidedly more delicate (see [62].) Furthermore, for our compliance output,

$$s(\mu) = s_N(\mu) + \ell(u - u_N) = s_N(\mu) + a(u, u - u_N; \mu) = s_N(\mu) + a(u - u_N, u - u_N; \mu) \tag{9.9}$$

from symmetry and Galerkin orthogonality. It follows that $s(\mu) - s_N(\mu)$ converges as the square of the error in the best approximation and, from coercivity, that $s_N(\mu)$ is a lower bound for $s(\mu)$.

Best Approximation

It now remains to bound the dependence of the error in the best approximation as a function of N. At present, the theory is restricted to the case in which $P = 1$, $\mathcal{D} = [0, \mu_{\max}]$, and

$$a(w, v; \mu) = a_0(w, v) + \mu a_1(w, v), \tag{9.10}$$

where a_0 is continuous, coercive, and symmetric, and a_1 is continuous, positive semi-definite ($a_1(w, w) \geq 0$, $\forall\, w \in X$), and symmetric. This model problem (9.23) is rather broadly relevant, for example to variable orthotropic conductivity, rectilinear geometry variations, piecewise-constant conductivity variations, and variable Robin boundary conditions.

We now suppose that the μ_n, $n = 1, \ldots, N$, are logarithmically distributed in the sense that

$$\ln\left(\mu_n + \overline{\lambda}^{-1}\right) = \ln \overline{\lambda}^{-1} + \frac{n-1}{N-1} \ln\left(\frac{\mu_{\max} + \overline{\lambda}^{-1}}{\overline{\lambda}^{-1}}\right), \quad n = 1, \ldots, N, \tag{9.11}$$

where $\overline{\lambda}$ is the maximum eigenvalue of a_0 relative to a_1. (Note $\overline{\lambda}$ is perforce bounded thanks to our assumption of continuity and coercivity; the possibility of a continuous spectrum does not, in practice, pose any problems.) We can then prove [46] that, for $N > N_{\text{crit}} \equiv 2e \ln(\overline{\lambda}\, \mu_{\max} + 1)$,

$$\inf_{w_N \in W_N} \|u(\mu) - w_N(\mu)\|_X \leq \sqrt{\frac{\gamma}{\alpha}}\, \|u(0)\|_X \, \exp\left\{\frac{-N}{2e\, \ln(\overline{\lambda}\, \mu_{\max} + 1)}\right\}, \quad \forall\, \mu \in \mathcal{D}\ . \tag{9.12}$$

We observe exponential convergence, uniformly (globally) for all μ in $\mathcal{D}$, with only very weak (logarithmic) dependence on the range of the parameter ($\mu_{\max}$).

The proof exploits the (parameter–space) interpolant as a surrogate for the Galerkin approximation. As a result, the bound is not "sharp": we observe many cases in which the Galerkin projection is considerably better than the associated interpolant; optimality (9.8) chooses to "illuminate" only certain points μ_n, automatically selecting a best "sub–approximation" amongst all possibilities — we thus see why reduced–basis *state-space* approximation of $s(\mu)$ via $u(\mu)$ is preferred to simple *parameter-space* interpolation of $s(\mu)$ ("connecting the dots") via $(\mu_n, s(\mu_n))$ pairs. Nevertheless, the logarithmic point distribution (9.11) implicated by our interpolant–based arguments is *not* simply an artifact of the proof: in numerous numerical tests, the logarithmic distribution performs considerably better than other obvious candidates, in particular for large ranges of the parameter. Fortunately, the convergence rate is not *too* sensitive to point selection: the theory only requires a log "on the average" distribution [46]; and, in practice, $\overline{\lambda}$ in (9.12) may be replaced with any "reasonable" value.

The result (9.12) is certainly tied to the particular form (9.23) and associated regularity of $u(\mu)$. However, we do observe similar exponental behavior for more general operators; and, most importantly, the exponential convergence rate degrades only very slowly with increasing parameter dimension, P. We present in Table 9.1 the error $|s(\mu) - s_N(\mu)|/s(\mu)$ as a function of N, at a particular representative point μ in $\mathcal{D}$, for the two-dimensional thermal fin problem of Section 9.2.2.0; we present similar data in Table 9.2 for the truss problem of Section 9.2.2.0. In both cases, since tensor-product grids are prohibitively profligate as P increases, the μ_n are

N	$\lvert s(\mu) - s_N(\mu)\rvert / s(\mu)$	$\Delta_N(\mu)/s(\mu)$	$\eta_N(\mu)$
10	1.29×10^{-2}	8.60×10^{-2}	2.85
20	1.29×10^{-3}	9.36×10^{-3}	2.76
30	5.37×10^{-4}	4.25×10^{-3}	2.68
40	8.00×10^{-5}	5.30×10^{-4}	2.86
50	3.97×10^{-5}	2.97×10^{-4}	2.72
60	1.34×10^{-5}	1.27×10^{-4}	2.54
70	8.10×10^{-6}	7.72×10^{-5}	2.53
80	2.56×10^{-6}	2.24×10^{-5}	2.59

Table 9.1: Error, error bound, and effectivity as a function of N, at a particular representative point $\mu \in \mathcal{D}$, for the two-dimensional thermal fin problem (compliant output).

N	$\lvert s(\mu) - s_N(\mu)\rvert / s(\mu)$	$\Delta_N(\mu)/s(\mu)$	$\eta_N(\mu)$
10	3.26×10^{-2}	6.47×10^{-2}	1.98
20	2.56×10^{-4}	4.74×10^{-4}	1.85
30	7.31×10^{-5}	1.38×10^{-4}	1.89
40	1.91×10^{-5}	3.59×10^{-5}	1.88
50	1.09×10^{-5}	2.08×10^{-5}	1.90
60	4.10×10^{-6}	8.19×10^{-6}	2.00
70	2.61×10^{-6}	5.22×10^{-6}	2.00
80	1.19×10^{-6}	2.39×10^{-6}	2.00

Table 9.2: Error, error bound, and effectivity as a function of N, at a particular representative point $\mu \in \mathcal{D}$, for the truss problem (compliant output).

chosen "log-randomly" over $\mathcal{D}$: we sample from a multivariate uniform probability density on $\log(\mu)$. We observe that for both the thermal fin ($P = 7$) and truss ($P = 4$), the error is remarkably small even for very small N; and, in both cases, very rapid convergence obtains as $N \to \infty$. We do not yet have any theory for $P > 1$. But certainly the Galerkin optimality plays a central role, automatically selecting "appropriate" scattered-data subsets of S_N and associated "good" weights so as to mitigate the curse of dimensionality as P increases; and the log–random point distribution is also important — for example, for the truss problem of Table 9.2, a (non–log) uniform random point distribution yields errors which are larger by factors of 20 and 10 for $N = 30$ and 80, respectively.

9.3.3 Computational Procedure

The theoretical and empirical results of Sections 9.3.1 and 9.3.2 suggest that N may, indeed, be chosen very small. We now develop off–line/on–line computational procedures that exploit this dimension reduction.

We first express $u_N(\mu)$ as

$$u_N(\mu) = \sum_{j=1}^{N} u_{N\,j}(\mu)\,\zeta_j = (\underline{u}_N(\mu))^T\,\underline{\zeta}, \tag{9.13}$$

where $\underline{u}_N(\mu) \in \mathbb{R}^N$; we then choose for test functions $v = \zeta_i$, $i = 1, \ldots, N$. Inserting these representations into (9.22) yields the desired algebraic equations for $\underline{u}_N(\mu) \in \mathbb{R}^N$,

$$\underline{A}_N(\mu)\;\underline{u}_N(\mu) \;=\; \underline{F}_N \tag{9.14}$$

in terms of which the output can then be evaluated as $s_N(\mu) = \underline{F}_N^T\,\underline{u}_N(\mu)$. Here $\underline{A}_N(\mu) \in \mathbb{R}^{N\times N}$ is the SPD matrix with entries $A_{N\,i,j}(\mu) \equiv a(\zeta_j, \zeta_i; \mu)$, $1 \le i, j \le N$, and $\underline{F}_N \in \mathbb{R}^N$ is the "load" (and "output") vector with entries $F_{N\,i} \equiv f(\zeta_i)$, $i = 1, \ldots, N$.

We now invoke (9.2) to write

$$A_{N\,i,j}(\mu) = a(\zeta_j, \zeta_i; \mu) = \sum_{q=1}^{Q} \sigma^q(\mu)\,a^q(\zeta_j, \zeta_i)\;, \tag{9.15}$$

or

$$\underline{A}_N(\mu) = \sum_{q=1}^{Q} \sigma^q(\mu)\,\underline{A}_N^q\;,$$

where $A_{N\,i,j}^q = a^q(\zeta_j, \zeta_i)$, $i \le i, j \le N$, $1 \le q \le Q$. The off–line/on–line decomposition is now clear:

> In the *off–line* stage, we compute the $u(\mu_n)$ and form the $\underline{A}_N^q$ and $\underline{F}_N$: this requires N (expensive) "a" finite element solutions and $O(QN^2)$ finite-element-vector inner products.
>
> In the *on–line* stage, for any given new μ, we first form $\underline{A}_N$ from (9.15), then solve (9.14) for $\underline{u}_N(\mu)$, and finally evaluate $s_N(\mu) = \underline{F}_N^T\,\underline{u}_N(\mu)$: this requires $O(QN^2) + O(\frac{2}{3}N^3)$ operations and $O(QN^2)$ storage.

Thus, as required, the incremental, or marginal, cost to evaluate $s_N(\mu)$ for any given new μ — as proposed in a design, optimization, or inverse-problem context — is very small: first, because N is very small, typically $O(10)$ — thanks to the good convergence properties of W_N; and second, because (9.14) can be very rapidly assembled and inverted — thanks to the off–line/on–line decomposition. For the problems discussed in this paper, the resulting computational savings relative to standard (well-designed) finite-element approaches are significant — at least $O(10)$, typically $O(100)$, and often $O(1000)$ or more.

9.4 A Posteriori Error Estimation: Output Bounds

From Section 9.3 we know that, in theory, we can obtain $s_N(\mu)$ very inexpensively: the on–line stage scales as $O(N^3) + O(QN^2)$; and N can, *in theory*, be chosen quite small. However, *in practice*, we do not know *how* small N can be chosen: this will depend on the desired

accuracy, the selected output(s) of interest, and the particular problem in question; in some cases $N = 5$ may suffice, while in other cases, $N = 100$ may still be insufficient. In the face of this uncertainty, either too many or too few basis functions will be retained: the former results in computational inefficiency; the latter in unacceptable uncertainty — particularly egregious in the decision contexts in which reduced–basis methods typically serve. We thus need *a posteriori* error estimators for s_N. Surprisingly, *a posteriori* error estimation has received relatively little attention within the reduced–basis framework [51], even though reduced–basis methods are particularly in need of accuracy assessment: the spaces are *ad hoc* and pre-asymptotic, admitting relatively little intuition, "rules of thumb," or standard approximation notions.

Recall that, in the section, we continue to assume that a is coercive and symmetric, and ℓ is "compliant."

9.4.1 Method I

The approach described in this section is a particular instance of a general "variational" framework for *a posteriori* error estimation of outputs of interest. However, the reduced-basis instantiation described here differs significantly from earlier applications to finite element discretization error [43, 40] and iterative solution error [54, 53] both in the choice of (energy) relaxation and in the associated computational artifice.

Formulation

We assume that we are given a function $g(\mu) : \mathcal{D} \rightarrow \mathbb{R}_+$, and a continuous, coercive, symmetric (μ-independent) bilinear form $\hat{a} : X \times X \rightarrow \mathbb{R}$, such that

$$\underline{\alpha}_0 \|v\|_X^2 \;\leq\; g(\mu)\; \hat{a}(v,v) \;\leq\; a(v,v;\mu), \quad \forall\, v \in X, \forall\, \mu \in \mathcal{D}. \tag{9.16}$$

We then find $\hat{e}(\mu) \in X$ such that

$$g(\mu)\; \hat{a}(\hat{e}(\mu), v) \;=\; R(v; u_N(\mu); \mu), \quad \forall v \in X \tag{9.17}$$

where for a given $w \in X$, $R(v; w; \mu) = \ell(v) - a(w, v; \mu)$ is the weak form of the residual. Our lower and upper output estimators are then evaluated as

$$s_N^-(\mu) \;\equiv\; s_N(\mu), \text{ and } s_N^+(\mu) \equiv s_N(\mu) + \Delta_N(\mu), \tag{9.18}$$

respectively, where

$$\Delta_N(\mu) \;\equiv\; g(\mu)\; \hat{a}(\hat{e}(\mu), \hat{e}(\mu)) \tag{9.19}$$

is the estimator gap.

Computational Procedure

Finally, we turn to the computational artifice by which we can efficiently compute $\Delta_N(\mu)$ in the on–line stage of our procedure. To begin, we rewrite the "modified" error equation, (9.17), as

$$\hat{a}(\hat{e}(\mu), v) \;=\; \frac{1}{g(\mu)} \left(\ell(v) - \sum_{q=1}^{Q} \sum_{j=1}^{N} \sigma^q(\mu) u_{N\,j}(\mu) a^q(\zeta_j, v) \right), \quad \forall\, v \in X$$

where we have appealed to our reduced–basis approximation (9.13) and the affine decomposition (9.2). It is immediately clear from linear superposition that we can express $\hat{e}(\mu)$ as

$$\hat{e}(\mu) = \frac{1}{g(\mu)} \left(\hat{z}_0 + \sum_{q=1}^{Q} \sum_{j=1}^{N} \sigma^q(\mu) u_{N\,j}(\mu) \hat{z}_j^q \right); \tag{9.20}$$

where $\hat{z}_0 \in X$ satisfies $\hat{a}(\hat{z}_0, v) = \ell(v),\ \forall\ v \in X$, and $\hat{z}_j^q \in X, j = 1, \ldots, N,\ q = 1, \ldots, Q$, satisfies $\hat{a}(\hat{z}_j^q, v) = -a^q(\zeta_j, v),\ \forall\ v \in X$. Inserting (9.20) into our expression for the upper bound, $s_N^+(\mu) = s_N(\mu) + g(\mu)\hat{a}(\hat{e}(\mu), \hat{e}(\mu))$, we obtain

$$\begin{aligned} s_N^+(\mu) = s_N(\mu) + \frac{1}{g(\mu)} \Bigg(& c_0 + 2 \sum_{q=1}^{Q} \sum_{j=1}^{N} \sigma^q(\mu) u_{N\,j}(\mu) \Lambda_j^q + \\ & \sum_{q=1}^{Q} \sum_{q'=1}^{Q} \sum_{j=1}^{N} \sum_{j'=1}^{N} \sigma^q(\mu) \sigma^{q'}(\mu) u_{N\,j}(\mu) u_{N\,j'}(\mu) \Gamma_{jj'}^{qq'} \Bigg) \end{aligned} \tag{9.21}$$

where $c_0 = \hat{a}(\hat{z}_0, \hat{z}_0)$, $\Lambda_j^q = \hat{a}(\hat{z}_0, \hat{z}_j^q)$, and $\Gamma_{jj'}^{qq'} = \hat{a}(\hat{z}_j^q, \hat{z}_{j'}^{q'})$. The off–line/on–line decomposition should now be clear.

> In the *off–line* stage we compute $\hat{z}_0$ and $\hat{z}_j^q$, $j = 1, \ldots, N,\ q = 1, \ldots, Q$, and then form c_0, Λ_j^q, and $\Gamma_{jj'}^{qq'}$: this requires $QN + 1$ (expensive) "$\hat{a}$" finite element solutions, and $O(Q^2N^2)$ finite-element-vector inner products.
>
> In the *on–line* stage, for any given new μ, we evaluate s_N^+ as expressed in (9.21): this requires $O(Q^2N^2)$ operations; and $O(Q^2N^2)$ storage (for c_0, Λ_j^q, and $\Gamma_{jj'}^{qq'}$).

As for the computation of $s_N(\mu)$, the marginal cost for the computation of $s_N^{\pm}(\mu)$ for any given new μ is quite small — in particular, independent of the dimension of the truth finite element approximation space X.

There are a variety of ways in which the off–line/on–line decomposition and output error bounds can be exploited. A particularly attractive mode incorporates the error bounds into an on–line adaptive process, in which we successively approximate $s_N(\mu)$ on a sequence of approximation spaces $W_{N_j'} \subset W_N, N_j' = N_0 2^j$ — for example, $W_{N_j'}$ may contain the N_j' sample points of S_N closest to the new μ of interest — until $\Delta_{N_j'}$ is less than a specified error tolerance. This procedure both minimizes the on–line computational effort and reduces conditioning problems — while simultaneously ensuring accuracy and certainty.

9.4.2 Method II

As already indicated, Method I has certain limitations; we discuss here a Method II which addresses these limitations — albeit at the loss of complete certainty.

Formulation

To begin, we set $M > N$, and introduce a parameter sample $S_M = \{\mu_1, \ldots, \mu_M\}$ and associated reduced–basis approximation space $W_M = \text{span}\ \{\zeta_m \equiv u(\mu_m),\ m = 1, \ldots, M\}$;

both for theoretical and practical reasons we require $S_N \subset S_M$ and therefore $W_N \subset W_M$. The procedure is very simple: we first find $u_M(\mu) \in W_M$ such that $a(u_M(\mu), v; \mu) = f(v), \forall\ v \in W_M$; we then evaluate $s_M(\mu) = \ell(u_M(\mu))$; and, finally, we compute our upper and lower output estimators as

$$s^-_{N,M}(\mu) = s_N(\mu), \qquad s^+_{N,M}(\mu) = s_N(\mu) + \Delta_{N,M}(\mu), \tag{9.22}$$

where $\Delta_{N,M}(\mu)$, the estimator bound gap, is given by

$$\Delta_{N,M}(\mu) = \frac{1}{\tau}\left(s_M(\mu) - s_N(\mu)\right) \tag{9.23}$$

for some $\tau \in (0,1)$. The effectivity of the approximation is defined as

$$\eta_{N,M}(\mu) = \frac{\Delta_{N,M}(\mu)}{s(\mu) - s_N(\mu)}. \tag{9.24}$$

For our purposes here, we shall consider $M = 2N$.

Computational Procedure

Since the error bounds are based entirely on evaluation of the output, we can directly adapt the off–line/on–line procedure of Section 9.3.3. Note that the calculation of the output approximation $s_N(\mu)$ and the output bounds are now integrated: $\underline{A}_N(\mu)$ and $\underline{F}_N(\mu)$ (yielding $s_N(\mu)$) are a sub-matrix and sub-vector of $\underline{A}_{2N}(\mu)$ and $\underline{F}_{2N}(\mu)$ (yielding $s_{2N}(\mu)$, $\Delta_{N,2N}(\mu)$ and $s^{\pm}_{N,2N}(\mu)$) respectively.

> In the *off–line* stage, we compute the $u(\mu_n)$ and form the $\underline{A}^q_{2N}$ and $\underline{F}_{2N}$: this requires $2N$ (expensive) "a" finite element solutions, and $O(4QN^2)$ finite-element-vector inner products.
>
> In the *on–line* stage, for any given new μ, we first form $\underline{A}_N(\mu)$ and $\underline{A}_{2N}(\mu)$ then solve for $\underline{u}_N(\mu)$ and $\underline{u}_{2N}(\mu)$, and finally evaluate $s^{\pm}_{N,2N}(\mu)$: this requires $O(4QN^2) + O(\frac{16}{3}N^3)$ operations and $O(4QN^2)$ storage.

The on–line effort for this Method II predictor/error estimator procedure (based on $s_N(\mu)$ and $s_{2N}(\mu)$) will require eightfold more operations than the predictor procedure of Section x.

Method II is in some sense very naive: we simply replace the true output $s(\mu)$ with a finer–approximation surrogate $s_{2N}(\mu)$. (There are more obscure ways to describe the method — in terms of a reduced–basis approximation for the error — however there is little to be gained from these alternative interpretations.) The essential computation enabler is again exponential convergence, which permits us to choose $M = 2N$ — hence controlling the additional computational effort attributable to error estimation — while simultaneously ensuring that $\varepsilon_{N,2N}(\mu)$ tends rapidly to zero. Exponential convergence also ensures that the cost to compute both $s_N(\mu)$ and $s_{2N}(\mu)$ is "negligible". In actual practice, since $s_{2N}(\mu)$ is available, we can of course take $s_{2N}(\mu)$, rather than $s_N(\mu)$, as our output prediction; this greatly improves not only accuracy, but also certainty — $\Delta_{N,2N}(\mu)$ is almost surely a bound for $s(\mu) - s_{2N}(\mu)$, albeit an exponentially conservative bound as N tends to infinity.

9.5 System Architecture

9.5.1 Introduction

The numerical methods proposed are rather unique relative to more standard approaches to partial differential equations. Reduced–basis output bound methods — in particular the global approximation spaces, *a posteriori* error estimators, and off–line/on–line computational decomposition — are intended to render partial–differential-equation solutions truly "useful": essentially real–time as regards operation count; "blackbox" as regards reliability; and directly relevant as regards the (limited) input–output data required. But to be truly useful, the methodology — in particular the inventory of on–line codes — must reside within a special framework. This framework must permit a User to specify — within a native applications context — the problem, output, and input value of interest; and to receive — quasi–instantaneously — the desired prediction and certificate of fidelity (error bound). We describe such a (fully implemented, fully functional) framework here: we focus primarily on the User point of view;

9.5.2 Overview of Framework

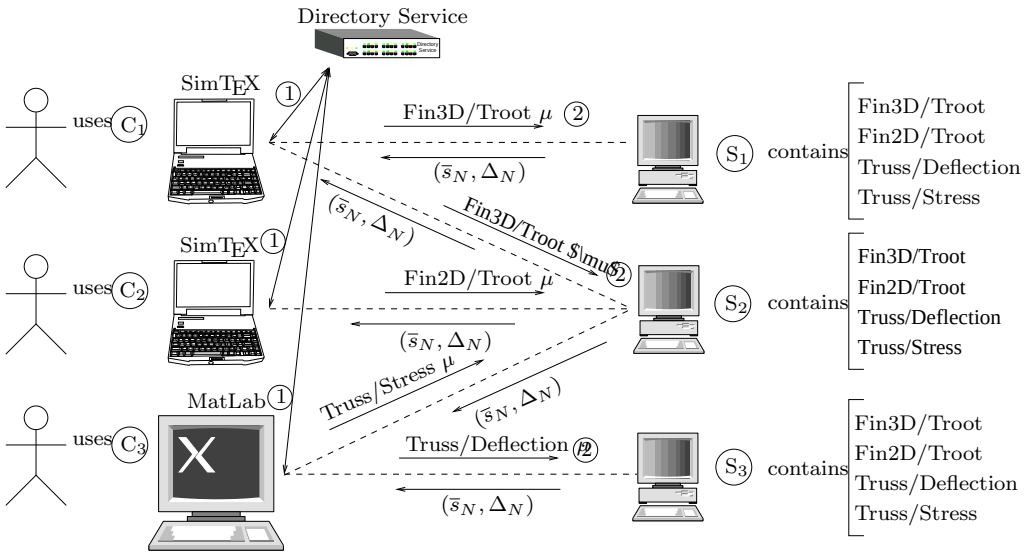


Figure 9.3: A Sample Use Case of the Framework

We show in Figure 9.3 a virtual schematic of the framework. The key components are the User, Computers, Network, Client software, Server software, and Directory Service. Each User interacts with the system through a selected Client (interface) which resides, say, on the User's Computer; we shall describe briefly below two Clients. Based on directives from the User, the Client broadcasts over the Network a Problem Label (e.g., Fin3D), Output Label (e.g., Troot) Pair. This Pair is received by the Directory Service — a White Pages — which informs the Client of the Simulation Resource Locator "SRL" — physical location

on a particular Computer — of a Server which can respond to the request. The Client then sends the Input (μ P–tuple Value) to the designated SRL. The Server — essentially a suite of on–line codes and associated input–output utilities — is awaiting queries at all times; upon receipt of the Input it executes the on–line code for the designated Output Label and Input Value, and responds to the Client with the Output Value ($\overline{s}_N$) and Error Bound Gap (Δ_N). The Client then displays or acts upon this information, and the cycle is complete.

Typically many identical (as well as different) Servers will be available, typically on many different Computers: there are multiple instances of the on–line codes. The Directory Service indicates to the Client the least busy Server so as to provide the fastest response possible. In some cases Clients may issue Input Value which are in fact a vector of input values — that is, L P–tuples. In this case the Directory Service will distribute the calculations over multiple (e.g., as many as L) Servers — in particular Servers on multiple Computers — so as respond more quickly to this multiple–input query. Our framework is clearly an example of "grid" computing, similar to GLOBUS, NetSolve, and Seti@HOME, to name but a few. Indeed, we exploit several generic tools upon which grid and network computing applications may be built; for example, we appeal to CORBA[†] (standardized by OMG[†]) to seamlessly manipulate the Server software *as if* it resided on the Client Computer. We remark that our reduced–basis output bound application is particularly well–suited to grid computing: the computational load on participating Computers (on which the Servers reside) is very light; and the Client–Server input/output load on the Network is very light. The network computing paradigm also serves very well the archival, collaboration, and integration aspects of standardized input–output objects.

9.5.3 Clients

We describe here two Clients: SIMTEX, which is a PDF–based "dynamic text" interface for interrogation, exploration, and display; and SIMLAB, which is a MATLAB–based "mathematical" interface for manipulation and integration. A third client has been developed: WEBLAB; however it is a direct application of the MATLAB client using the MATLAB Web Server Toolbox.

SimTEX

SIMTEX combines several standardized tools so as to provide a very simple interface by which to access the Servers. A particularly nice feature of SIMTEX is the natural context which it provides — in essence, defining the input–output relationship and problem definition in the language of the application. The SIMTEX Client should prove useful in a number of different contexts: textbooks and technical manuscripts; handbooks; and product specification and design sheets.

The SIMTEX Client consists of an authoring component, a display and interface component, and an "intermediary" component. The authoring component is — a standard in scientific typesetting — enhanced (via `hyperref`) with a new `acteq` environment which per-

[†] Common Object Request Broker Architecture — `http://www.corba.org`

[†] Object Management Group — `http://www.omg.org`

mits the inclusion of actionable equations. The acteq environment links an equation to a Problem Label, Output Label(s) and Input Value template. The output is a PDF document: the PDF document serves as the display, graphics, and (rudimentary) interface component of SimTeX. The PDF document contains a form which accepts the Input Values, and an "equal sign" button which initiates the Client–Directory Services Client–Server dialogue described in the previous document. Upon completion of the cycle, the PDF document is updated to display the values of the output and error bound for the Input Values submitted; in cases in which multiple input values or outputs are selected, appropriate graphics are presented using the FIGURE button. Finally, since PDF is not a programming language, and Client–Server intermediary is required: a CGI script serves to parse the PDF form, communicate with the Server, and finally update the Client.

As an example, we include here an actionable equation — the actual SimTeX user interface — for the several outputs (the root temperature, tip temperature, volume) associated with the three–dimensional thermal fin example:

$$\begin{pmatrix} T_{\text{root}} \\ T_{\text{tip}} \\ \text{Volume} \end{pmatrix} = \mathcal{F}\left(\begin{array}{rl} k^1 = & 0.4 \\ k^2 = & 0.6 \\ k^3 = & 0.8 \\ k^4 = & 1.2 \\ \text{Bi} = & 0.1 \\ t = & 0.3 \\ L = & 2.8 \end{array} \right) \quad \text{Submit} \quad \pm$$

FIGURE

The input list corresponds to the μ vector described in Section 9.2.2.0; the input values must lie in the parameter domain $\mathcal{D}$ described in Section 9.2.2.0. The notation $\text{Output} = \mathcal{F}(\text{Input})$ is a description of the input–output relationship $s(\mu)$ implied by $s = \ell(u(\mu))$. The actionable PDF version of this entire paper (in which is embedded the actionable equation) may be found on our web site[†]; readers are encouraged to access this electronic version of the paper and exercise the SimTeX interface, a brief users manual for which may be found again on our web site[†].

SimLaB

The main drawback of SimTeX is the inability to manipulate the on–line codes. SimLaB is a suite of tools that permit Users to incorporate Server on–line codes as MATLAB functions within the standard MATLAB interface; and to generate new Servers and on–line codes from standard MATLAB functions (which themselves may be built upon other on–line codes). In short, SimLaB permits the User to treat the inputs and outputs of our on–line codes as

[†] http://augustine.mit.edu/jfe/jfe.pdf
[†] http://augustine.mit.edu/guided_tour.pdf

mathematical objects that are the result of, or an argument to, other functions — graphics, system design, or optimization — and to archive these higher level operations in new Server objects available to all Clients once registered in the Directory Service.

For example, to incorporate the Fin3D input–output relationship into MATLAB, we first generate the needed MATLAB functions using a MATLAB script called `st2m`. This script, by default, generates automatically a MATLAB function for each Output registered in the Directory Service. It is also possible to ask for a specific Problem and Output using the following command in MATLAB: `st2m -model fin3d -output Troot` — however it implies that one knows the name of the Problem and Output. Then, to set the values for the seven components of the parameter vector, we enter

```
p.values(1).value=0.8;
p.values(1).name='k1';
p.values(2).value=2;
p.values(2).name='k2';
p.values(3).value=14;
p.values(3).name='k3';
p.values(4).value=3;
p.values(4).name='k4';
p.values(5).value=0.2;
p.values(5).name='Bi';
p.values(6).value=0.1;
p.values(6).name='t';
p.values(7).value=2.5;
p.values(7).name='L';
```

within the MATLAB command window. To determine the output value and bound gap for this value of the 7–tuple parameter, we then enter

```
[Troot, Bound_Troot] = fin3d_Troot( p )
```

which returns

```
Troot = 1.06869419906058
Bound_Troot = 1.06869419906058
```

It is also now possible of course to find all values of `Troot` greater than 1.05 for t in the range $[0.1, 0.5]$ and all other parameters fixed as in the list above. To wit, we enter

```
i=1;
while( i < 1000 )
  p.values(6).value=0.1+i*(0.4)/1000;
  t(i)=p.values(6).value;
  [o(i),e(i)] = fin3d_Troot( p );
  i=i+1;
end
plot(x(o<1.05),o(o<1.05),'b'); hold on; grid on;
plot(t(o>=1.05),o(o>=1.05),'r');
line([max(t(o>1.05)) max(t(o>1.05))], [1  1.1]); %% L1
```

which generates the figure 9.4. Where the line L_1 splits the domain $[0.1; 0.5]$ at $t_{\max} = \max(t(o > 1.05)) = 0.$

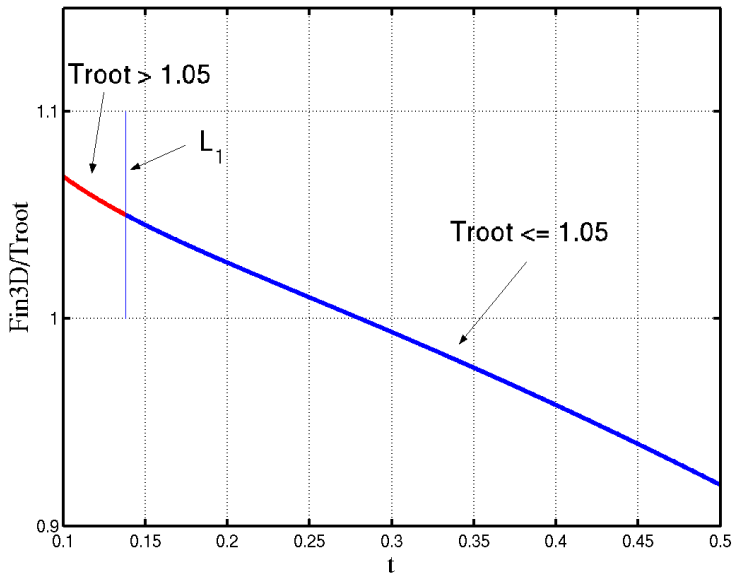


Figure 9.4: Plot

between the values of `Troot` greater than 1.05 ($t < t_{\max}$) and the values of `Troot` less then 1.05 ($t \geq t_{\max}$). To be more certain that `Troot` was *truly* greater than 1.05, we could easily ask for those values of t for which $\overline{s}_N - \Delta_N \geq 1.05$; again readily effected by a simple function call. Obviously, once the on–line code is within the MATLAB environment, we have the full functionality of MATLAB at our disposal; and the rapid response of the reduced–basis output bounds maintains the immediate response expected of an interactive environment, even though we are in fact solving — and solving reliably — three–dimensional partial differential equations.

9.5.4 Overview Of The Framework

Introduction

The main current trend in computing and scientific computing is Distributed Objects — see for example the .NET, Mono, Globus, Netsolve, Seti@Home technologies and as mentioned earlier, we use CORBA as the underlying technology for our Framework.

CORBA is the Distributed Objects technology developed and standardized by the OMG. As envisioned by CORBA, Distributed Objects are the melding of concepts from two paradigms, Client-Server(or more precisely Distributed Computing) and Object Orientation(OO) with some slight differences: (*i*) a Client knows an object by its interface; (*ii*) objects are not always local with respect to their Clients; (*iii*) dynamic composition may compose objects into new application; (*iv*) objects hide many of the underlying differences(between Client and Server) in architecture through encapsulation.

The combination of the Client-Server and OO models gives us the best features: ability

to distribute risk(fault tolerance), rightsizing system development with small composable subtasks and having looser coupling thanks to well-defined integration and interfaces. Another advantage is also that we can have much more complicated topologies — see, for example, the Figure 9.3 — than typically found in the Client-Server paradigm: a Client request computation from a Server which is itself a composition of several other Servers; in this context, our requested object is a Client-Server — a Server for the Client and a Client for the Servers composing it.

It is interesting to note that although there is generally a many-to-one relation for Client to Server, Clients may want to have access to more than one Server for a given purpose. In the overview of our Framework we have seen that it was effectively the case — see Figure 9.3. Indeed, if a single source can be beneficial, it can also be expensive : risk of central outage(single point failure), too little specialization(resource utilization is suboptimal) long queues for services and large distances over which products chip and so on.

Development costs may rise also: distribution introduces more difficult problems such as, for example, the logistics of coordinating multiple sites. Two other particularly serious issues are the *network latency* and the *scalability*. They are difficult to determine beforehand and they can undermine gravely the deployment of the Framework.

Those issues are partly addressed by the numerical methods proposed — see Section 9.5.1. Only partly because the *network latency* is a difficult issue and reducing it is by no means easy — see the Akamai technology[†]. And regarding the *scalability*, the numerical methods proposed are not sufficient — although lots of problems(scheduling, monitoring,...) arising when using more conventional methods are of no concern in the Reduced-Basis Output Bound methods context,— therefore an adequate design for our Framework is also a requirement to ensure *scalability*.

The Framework relies on a library, ST[†], which sits on top of CORBA — see Figure 9.5 — and its associated services. Three Clients — SIMTEX, SIMLAB, WEBLAB— have been developed with ST.

The Main Actors Of St

The design of ST shares similar concepts to the one that can be found in modern Graphical User Interface (GUI) libraries : the `SApplication` class and the `SSimget` class and their respective subclasses. CORBA is a complex middle-ware specification and through simple coarse grain interfaces and high level concepts ST encapsulates all CORBA aspects — standard CORBA calls or CORBA Services — inside its classes. In the following section, we shall describe briefly some aspects of the Framework.

SApplication As shown on Figure 9.5, ST sits on top of CORBA and the CORBA Services : it encapsulates all CORBA and associated components into a small set of classes with well defined behavior. Central to this design is the `St::SApplication` which encapsulates initialization of CORBA, determines the available services and, in Server mode using `St::SApplicationServer` subclass, drives the execution flow of the application through its

[†] http://www.akamai.com
[†] Simulation Toolkit

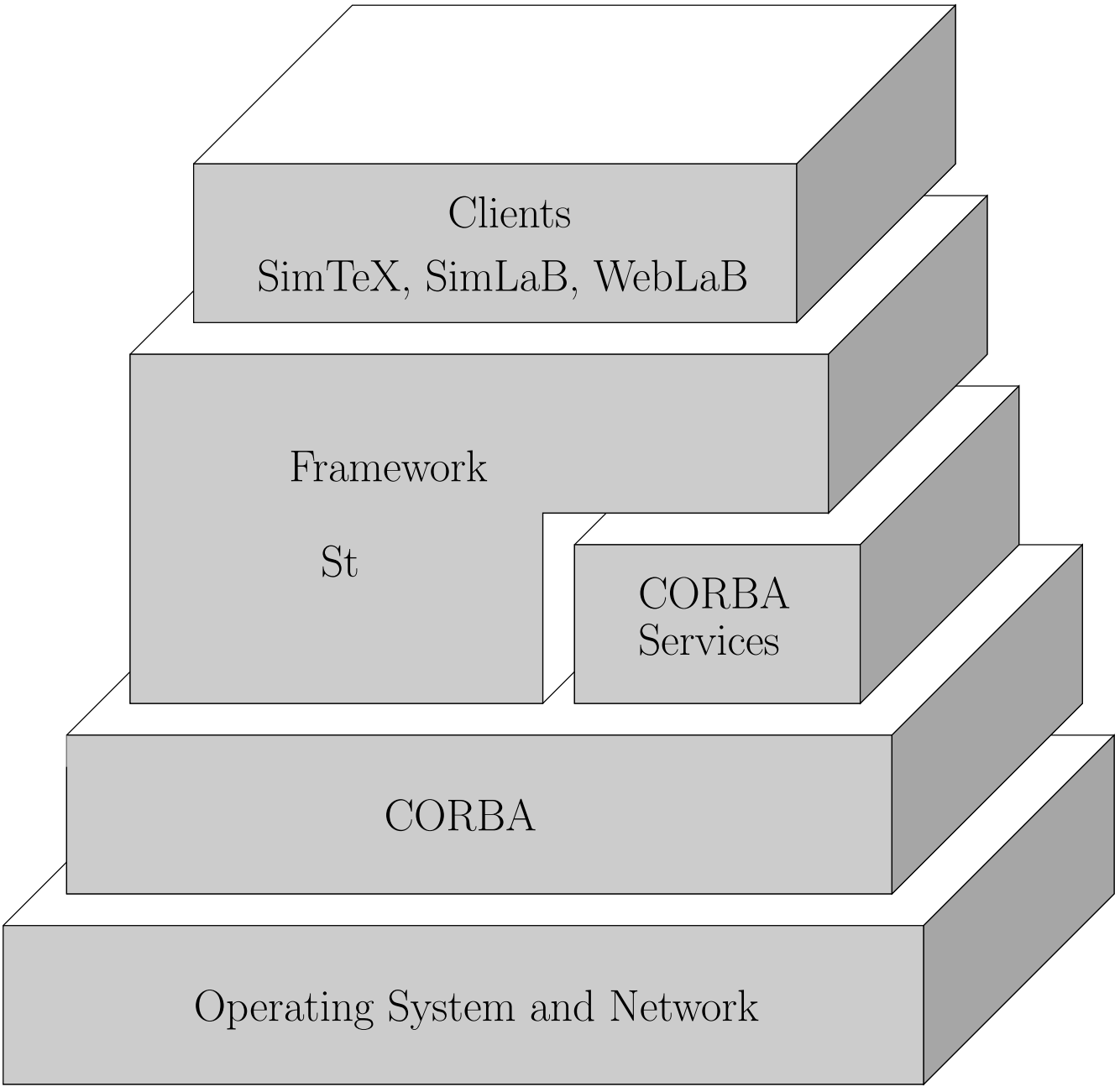


Figure 9.5: ST and CORBA

`St::SApplicationServer::run()` method which is basically an infinite loop waiting for new requests from Clients, see Section 9.5.5.0. A `SApplication`, and subclasses, follows the Singleton pattern to ensure that there exists only one instance of this class per process. It is possible to check for the availability on the Directory Service Server — see Figure 9.3 — of three standard CORBA Services through the member functions `bool hasNamingService()`, `bool hasTradingService()` and `bool hasImplementationRepository()` and have access to each to these Services through there associated class, `SNamingService`, `STradingService` and `SImplementationRepository`.

In practice, accessing the CORBA Services directly from a Client or a Server is neither needed nor recommended, it is usually taken care of by the objects that represent the Simulation objects, the Simgets.

An issue arising inevitably in Distributed Computing is Security. The most basic Security action that can be taken is related to the protection of the computers running the Servers: the processes associated to the Servers must have the lowest permissions on the system. On a Unix system for example, such a process would belong to the `nobody` user and group. By doing so, if someone with ill-intent manages to enter the operating system using those processes, he cannot do any harm. That is the least that has to be done. Unfortunately, ill-intended people can still do harm to the Framework. In order to avoid this, we use an authentication layer on top of the Framework using the Secure Sockets Layer (SSL). It has also the advantage to have some statistics on the Framework usage. The inclusion of the MICOSEC which is an implementation of the CORBA Security Services (CORBASec) Level 2 version 1.7[†] is underway. The current and future security features are/will be built-in `SApplication`.

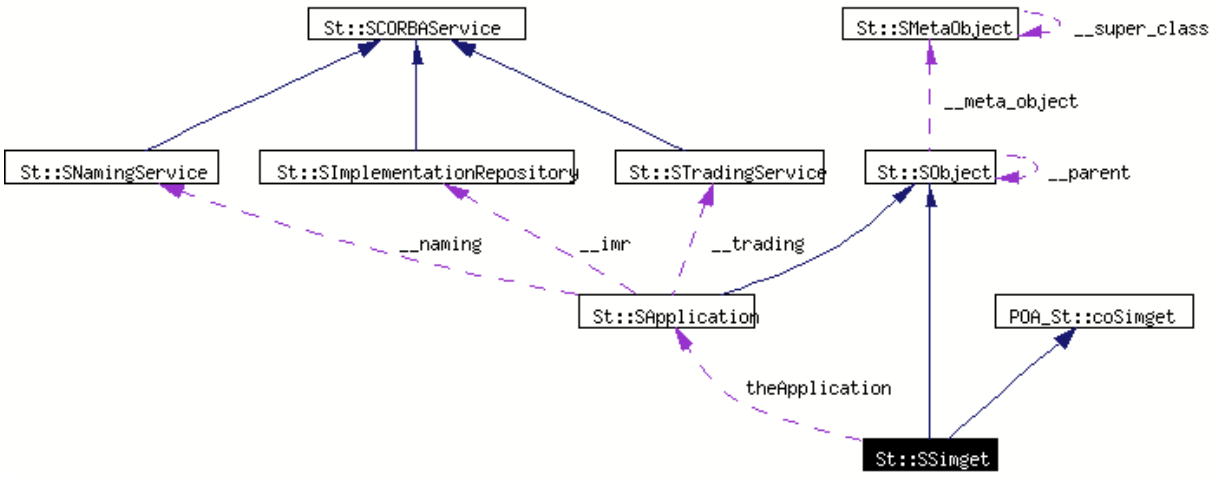


Figure 9.6: The Simget Collaboration Diagram

SSimget As described on the collaboration diagram 9.6, a Simget is a SObject, base class for all ST objects, and also a CORBA object through the interface `POA_St::coSimget`. It contains a reference to the reference of the `SApplication` running in order to have access to

[†] http://www.micosec.org/

the CORBA Services to be able to register itself automatically in each of them, provided that some of the Services are available. A Simget is a rather complex object which is following the Composite pattern : it can be either a standalone computational object or a composite object of Simgets.

The constructor of the SSimget class and subclasses follows the prototype

```
#include <SSimget.hpp>
SSimget( SSimget* parent, const char* name );

class A
 : public St::SSimget
{
  public:
  A( St::SSimget* parent, const char* name )
   : St::SSimget( parent, name )
 {}
};
```

where parent is the parent Simget and name is the name of the object passed to SObject — every SObject has a name. — Then, using this composite or tree, it is possible to define a directory of Simgets in the various CORBA Services. The name of the Simget and its relationship with the other ones will be used to define its location in the Services. For example, if the following code is executed

```
St::SApplicationServer *app = new St::SApplicationServer;
A * a = new A( 0, "a");
A * b = new A( a, "b");
A * c = new A( b, "c");
A * d = new A( a, "d");
app->setMainSimget( a );
```

then the Naming Service will contain the following references

```
console >  nsadmin -ORBNamingAddr \
 inet:<directory service computer>:<port of the service>
> ls
a.a/
> ls a.a
b.b/
d.d/
> ls a.a/b.b
c.c/
```

nsadmin is a tool provided by MICO that allows to browse the Naming Service using the UNIX-like tools like ls or rm.

The SSimget is very general and is the base class for all Simgets. In the context of the Reduced-Basis methods and the Framework we built for them, we added other concepts which could be used to other kind of problems. First, we defined a *model* which is an object describing the model or problem being considered, for example Fin3D. The associated class is St::SModel. Then, we defined a model component which represents a general concept

of model part. The associated class is SModelComponent. Finally, various subclasses of SModelComponent arise like SModelOutput, SModelOutputSet, and SModelField which are associated with the computation respectively of one output and associated error estimation if available, of a set of different outputs and associated error estimations, or a field — say a Finite Element field associated with a Finite Element simulation. — Each of these objects are CORBA objects and follow a relatively simple IDL[†] interface that allows their remote manipulation from a Client program. Here is the IDL interface for a SModelComponent

```
interface coModelComponent :  coSimget
{
  //! get the name of the Model
  string getModelName();

  //! set the new parameter set
  void setParameterSet( in coParameterSeq  pset );

  //! set a new Range
  void setRange( in coRangeSeq range );
};
```

and SModelComponent is defined as a subclass of SSimget which implements the interface coModelComponent. The other classes, SModelOutput, SModelOutput and SModelField, have similar simple CORBA interfaces with further specification. For example a SModelComponent and subclasses have the following constructor

```
SModelComponent( SModel* parent, const char* name );
```

which means that a model component has to have a model as a parent Simget. The relationship model/component is enforced through this explicit constructor. For an implementation example see the Section 9.5.5.0.

SMonitor While monitoring is not an important issue for our Framework since we deal with black-box real-time response simulations, it is interesting however to collect statistics or to provide such a tool for future Simgets that will need monitoring. Monitors are Clients and Servers for the Framework, they are implemented using the Observer and Memento design pattern. The base class for monitors is SMonitor which encapsulates the commonalities for all monitors which are here the interfaces following the above-mentioned design patterns.

The abstraction is that a Simget sends messages stored in a Memento upon special Events to the current Observers/monitors of the Simget. Monitors are Simgets which are Clients to the Simulation Simgets. Here is a short code snippet to describe how it works

```
St::SApplicationServer* app = new St::SApplicationServer;
St::SMonitorOutput* mon =
    new St::SMonitorOutput( 0, "monitor", SYSLOG );
mon->setMonitor( St::MONITOR_TIME |
                 St::MONITOR_RESULT |
                 St::MONITOR_PARAMETERS );
```

[†] Interface Definition Language

```
app->setMainSimget( mon );
app->run();
```

the second line creates a monitor for all Simgets in the Directory Services which will use syslog to log events like computation time, results from the Simgets and parameters passed to the Simgets. In order to create a model specific monitor for the Fin3D model and its outputs, we use the following code for example

```
St::SApplicationServer* app = new St::SApplicationServer;
St::SModel* model = new SModel( 0, "fin3d" );
St::SMonitorOutput* mon =
    new St::SMonitorOutput( fin3d, "monitor", SYSLOG );
mon->setMonitor( St::MONITOR_TIME |
                 St::MONITOR_RESULT |
                 St::MONITOR_PARAMETERS );
app->setMainSimget( model );
app->run();
```

A monitor can just log events but it provides also a CORBA interface that allows Clients like SimTeX for example to access statistics of the Simget being called. Note that this Monitor design pattern is not only used by ST to provide monitoring of the Simget but it is also used for all kind of objects that requires monitoring like solvers, preconditioners, iterative processes in general. The monitor design pattern is a non trivial application of several design patterns and is a reusable component/strategy in scientific computing libraries. We just provided an application in the context of the Framework and Distributed Computing.

XML XML[†] is another trendy technology and it is often associated with Distributed Computing — see SOAP[†] or XML-RPC[†]. — The eXtended Markup Language is used in the Framework as a meta-data language to describe the Simgets. If it exists at the creation of a Simget, an associated XML file is loaded and is available through the CORBA interface of the Simget. In the current implementation, the XML data contain only static information like the name of the Simget, the model name, some parameter set data — like the name of the parameters, their minimum and maximum values, and a description — and a description of the Simget. Here is a simple XML example

```
  <!DOCTYPE St>
<St>
<SModelOutput name="Troot" parent="fin3d" model="fin3d">
 <Author firstname="Christophe" lastname="Prud'homme"/>
 <Basis db="fin3d_Troot.bb" N="20" Nused="20"></Basis>
 <ParameterSet name="D" dimension="7">
    <Parameter ub="10" lb=".1"  name="k1">k1</Parameter>
    <Parameter ub="10" lb=".1"  name="k2">k2</Parameter>
    <Parameter ub="10" lb=".1"  name="k3">k3</Parameter>
    <Parameter ub="10" lb=".1"  name="k4">k4</Parameter>
```

[†] http://www.w3.org/XML/
[†] http://www.w3.org/TR/SOAP/
[†] http://www.xmlrpc.com/spec

```
    <Parameter ub="1"  lb=".01" name="Bi">Bi</Parameter>
    <Parameter ub="3"  lb="2"   name="L">L</Parameter>
    <Parameter ub=".5" lb=".1"  name="t">t</Parameter>
  </ParameterSet>
  <Description>
    Troot computes the temperature at the root
    of the 3D Thermal Fin.

    BlackBox: O(Q^2) in compliant case.

    See http://augustine.mit.edu/simtex/fin3d.pdf
    for more details.
  </Description>
</SModelOutput>
</St>
```

This is particularly helpful for automatic generation of Clients for the Framework. For example SIMLAB uses this feature to automatically generate `.m` files for the Simgets registered in the Directory Service — see Section 9.5.3.0. Without having to communicate with the Server, the Client can for instance provide documentation about the Simget, and checks that the parameter set which will be sent to the Server is contained in $\mathcal{D}$ — see Section 9.3.1.

In the future an automatic *C++* code generator for the Clients will be implemented using the XML meta-data provided by the Simgets registered in the Directory Service. At present only SIMLAB does automatic code generation using the XML meta-data.

9.5.5 A Simple Client/Server Implementation In *C++*

In a Client-Server paradigm, we have a Server side and a Client Side. In the next sections, we are going to present a sample code of a Server running the Simget computing the temperature at the root of the 3D thermal fin and one possible — while very simple — Client code in C++ accessing the code and the equivalent in MATLAB using SIMLAB in order to compare the amount of work needed. Some knowledge of *C++* is required, however some of the general ideas and concepts appear in the code.

Server Side

First look at what the main program looks like from the server point of view:

```
#include <SApplicationServer.hpp>
using namespace St;

int main(int argc, char** argv)
{
  SCommandLineArguments::init( argc, argv );
  SApplicationServer* server = new SApplicationServer();

  server->run();
}
```

The line 6 initializes the command line parsing system. Then, we define the ST Server application using the SApplicationServer class. Unfortunately this example does nothing but running forever. Indeed the server->run() is an infinite loop, where the server is waiting for new requests.

Now we need to feed the server with *Simgets*.

```
SApplicationServer* server = new SApplicationServer();
SModel* fin3d = new SModel( 0, "fin3d");
server->setMainSimget( fin3d );
server->run();
```

With the two new lines 2 and 3, we have created a possibly new Model in the ST Services(Naming and Trading). First we define the "fin3d" model — second argument — which has no parent in the current SApplicationServer — first argument. Again this server is not very helpful, since it does not do any real computation.

We can add for example a Simget which compute the temperature at the root of the three-dimensional thermal fin — see line 2 in the next listing.

```
SModel* fin3d = new SModel( 0, "fin3d");
SModelOutput* troot = new Troot( fin3d, "Troot");
server->setMainSimget( fin3d );
```

The work in the main program is finished and doesn't need further work. when server->run() is executed the server will provide the Troot Simget associated with the model fin3d.

Now let us see what the Troot looks like:

```
#include <SModelOutput.hpp>
using namespace St;
class Troot: public SModelOutput
{
public:
  Troot( SModel* parent, const char* name )
  : SModelOutput( parent, name )
  {<snip> }
  virtual SOutput* run( SParameterSet const& pset )
   {
     SOutput* output = new SOutput;
     // do the computation here for the parameter set pset
     . . .
     output->output = <prediction value>;
     output->error = <associated error estimator value>;
     return output;
    }
};
```

The last step is the execution of the Server :

```
  fin3d_Troot --server augustine.mit.edu --daemon
```

The -server option tells the SApplicationServer to register the Simgets in augustine.mit.edu. Whereas the -daemon tells SApplicationServer to detach from its parent process and run forever in the background.

Under Unix/Linux, a database and the `init.d` system are used to start and stop automatically the Servers used and registered on one computer. The advantage is that if a computer containing Simgets is rebooted or shutdown for some reason then the Simgets will be cleanly shutdown. Indeed, one of the major difficulty with Distributed Computing is the fact that the Object have *external* references and they need to be deleted properly if the corresponding is being shutdown.

Looking back at the example presented above, it would seem that it is possible to have only one Simget per process. Recall the tree structure of the SObject/Simget classes plugged into `SApplicationServer` using its `setMainSimget()` member function, then enabling new Simgets in the same process as the one presented earlier is a one-liner per Simget provided that each Simget has been implemented like `Troot` for example.

```
SModel* fin3d = new SModel( 0, "fin3d");
SModelOutput* troot = new Troot( fin3d, "Troot");
SModelOutput* ttip = new Troot( fin3d, "Ttip");
SModelOutput* volume = new Troot( fin3d, "Volume");
server->setMainSimget( fin3d );
```

The code shown above adds two Simgets to the Framework — in this case the temperature at the tip of the 3D fin and the volume of the 3D fin. — As one can see, in one process we can register one or more components for a given Model in the various services available and the associated code and programming time are reduced.

A Note On Parallelism Finally. let us present a possible extension for the Server part: the various registered Simgets in one process are often independent from each other — eventually a Simget could depend on the computation done by another one. For example a Simget providing a dimensional `Troot` would depend on the non-dimensional `Troot` counterpart : this kind of dependency requires generally simple algebra while the non-dimensional `Troot` would require an Online Code as described in the sections 9.3 and 9.4. — The Simget could be accessed by several users at the same time, and it appears that exploiting parallelism is important in this case to make sure that the Framework is responsive enough with respect to multiple Clients and provide Real-Time response for each Client. We implement parallelism at two levels on the Server side. Firstly, we use a multi-threaded CORBA implementation — MICO/MT[†] which is about to be or already merged with MICO[†] — that enables us to answer "simultaneously" to concurrent requests. Secondly, we implement thread-safe Simgets so that when the Client asks for multiple evaluations — say a thousand randomly chosen parameters, — we use the interface of `SModelOutput` which provides multiple parallel evaluations — see `SModelOutput::run(MultiParameter)` in the next section. — In order to control the level of parallelism for many outputs, the `SApplication` base class provides a way of defining the number of possible multiple threads.

```
SApplicationServer* __app = new SApplicationServer;

// tell thread-aware classes (SModelOutput for example)
```

[†] http://micomt.sf.net
[†] http://www.mico.org

```
// and their subclasses  to use up to 3 threads for
// many outputs evaluations
__app->setNumberOfProcessors( 3 );
```

The Framework provides a simple way to allow multi-threaded classes. The developer has to use the class SThread as the base class for a multi-thread enabled class. Here is an example

```
#include <iostream>
#include <SThread.hpp>

class A : public St::SThread
{
public:
 // a possible default constructor
 A( int __no = 1 ) : St::SThread(), _M_no( __no ) {}

protected:
 // run() method has to be re-implemented
 void run()
 {
   St::SMutex __mutex;
   if ( __mutex.tryLock() )
     {
       std::cerr << "run␣thread␣number␣" << _M_no
                 << std::endl;
     }
   else
     {
       ;//locking did not work
     }
  // __mutex.unlock() is called by destructor automatically
  // so no need to call it here
 }
private:
 int _M_no;
};
```

Note the utilization of the mutex class SMutex — which is provided by SThread.hpp — to ensure that the output will be written sequentially in the console instead of having both threads outputs being mixed up. Then the controller program can do the following

```
int main( int argc, char** argv )
{
  St::SCommandLineArguments::init( argc, argv );
  St::SApplication* __app = new St::SApplication;
  // can either use NPROCS environment variable or
  // SApplication::setNumberOfProcessors( n )
  // to set the number of threads
  int __nprocs = __app->getNumberOfProcessors();
  A** __threads = new A*[ __nprocs ];
```

```
  for( int __i = 0; __i < __nprocs; ++__i )
    {
      // start a new thread and call thread1.run()
      __threads[ __i ] = new A( __i );
      __threads[ __i ]->start();
    }
  // wait for all threads to finish
  for( int __i = 0; __i < __nprocs; ++__i )
    {
      __threads[ __i ]->wait();
    }
}
```

The environment variable NPROCS controls the number of threads than can be used, it is overridden by the SApplication::setNumberOfProcessors(int) member function. When running the above code, we get the following expected console output :

```
console > export NPROCS=4
console > test_thread
run thread number 0
run thread number 1
run thread number 2
run thread number 3
```

The various classes handling the Online Codes computation uses SThread which eases the use of multi-thread programming by encapsulating all the pthread library.

To summarize, parallelism is achieved at two levels on the Server side: at the ORB level using a multi-threaded ORB and at the Simget level. Note that it is transparent to the Framework user except for the part he has to provide the number of threads and that his code has to be thread-safe in order to be used in this multiple threads (> 1) environment. If it is not thread-safe then the Framework user can still fall back to a single thread environment which is the default and conservative behavior. There is a Client parallel counterpart which is discussed in the next section which uses yet another feature of the Framework.

Client Side

On the client side, the Client developer has access to the Naming and Trading services in order to find the object he is looking for. In the next listing, we present a client using the Naming service. The way the Simget in general are stored in the Naming service resembles a directory tree where / is the root of the tree. Under the fin3d model appears the Troot, Ttip and Volume Simgets. In the following example, we retrieve a reference to the Troot Simget and use its interface to do some computations:

```
#include <SApplication.hpp>
#include <SModelOutput.hpp>
using namespace St;


int main(int argc, char** argv)
```

```
{
  SCommandLineArguments::init( argc, argv );
  SApplication* client = new SApplication();

  SModelOutput_var fin3d_troot =
       client->resolve( "fin3d/Troot" );
  fin3d_Troot->setNewParameterSet( pset );
  fin3d_Troot->setRange( range );
  fin3d_troot->run( MultiParameter );

  coOutputArray_var results = fin3d_Troot->getOutputs();
  for( ULong  __i = 0;__i < __output_pl->length();++__i)
    {
      std::cerr << "output(␣" << __i << ")="
                << result( __i ).output << std::endl;
      std::cerr << "error(␣" << __i << ")="
                << result( __i ).error << std::endl;
    }
}
```

The only *non standard* C++ statement is the call to `client->resolve()` which will lookup in the naming service in order to get an handle on the corresponding simulation which was defined earlier. The other statements are classical object manipulation, setting the parameter set, set the range if needed, run the simulation and then retrieve the results.

Writing a client for the framework is relatively easy task, however providing a graphical user interface to the client code certainly comes with some efforts, that one of the reason why we based SimTeX and SimLab Clients upon existing standards(MatLab and PDF) — to alleviate the work in programming user interfaces.

Now it remains to execute the Client program:

```
fin3d_Troot_client --server augustine.mit.edu
```

This command line tells the Client to contact `augustine.mit.edu` for Simget lookups.

A Note On Parallelism Parallelism is/can be also implemented on the Client side. However it is not an automatic builtin feature of the Client. The Framework allows to implement a Client which takes advantage of certain of its properties. The starting point is that, as mentioned in the previous Section, the ORB is multi-threaded and recall that several instances of the same Model/Simget can exist in the CORBA Services — namely the Naming or the Trading Services, — so with those two features one can build a Client which access several instances of the same Simget or several different Simgets at the same time in different threads. The `SThread` class presented in the section 9.5.5.0.0 can be used to implement such a multi-threaded Client. Another important part here is the load balancing system of the Services which ensures that the least used instance will be selected by the Service to answer each new request.

To summarize the parallelism aspects of the Framework, both Server and Client sides, we achieved a three levels parallelism which ensures real-time response — with the certificates of fidelity developed in the first sections of the paper — in the context of many evaluations and many users.

9.6 Conclusion

This framework is likely to be extended in the next few months, and will certainly enjoy some improvements. It has already been tested on many problems already available online on our web site http://augustine.mit.edu. The component-wise design of the overall system is very flexible, scales well as the number of Online Codes increases and is an elegant solution as a platform for engineering design, research and education. Regarding the mathematical framework, it has been extended to several problem categories: non-coercive partial differential equations, parabolic equations, generalized eigenvalue problems, non-symmetric and non-compliant cases, see [62].

Acknowledgements We would like thank Thomas Leurent (formerly of MIT), Shidati Ali of the Singapore-MIT Alliance and Yuri Solodukhov for very helpful discussions. We would also like to acknowledge our longstanding collaborations with Professor Jaime Peraire of MIT and Professor Einar Rønquist of the Norwegian University of Science and Technology. This work was supported by the Singapore-MIT Alliance, by DARPA and ONR under # F49620-01-1-0458, by DARPA and AFOSR under Grant N00014-01-1-0523 (Subcontract # 340-6218-3), and by NASA under Grant # NAG-1-1978.

Chapter 10

A Domain Specific Embedded Language in *C++* for Automatic Differentiation, Projection, Integration and Variational Formulations

Author: C. Prud'homme

10.1 Introduction

Numerical analysis tools such as differentiation, integration, polynomial approximations or finite element approximations are standard and mainstream tools in scientific computing. Many excellent libraries or programs provide a high level programming interface to these methods : *(i)* programs that define a specific language such as the Freefem software family [29, 57], the Fenics project [35, 34], Getdp [23] or Getfem++ [69], or *(ii)* libraries or frameworks that supply some kind of domain specific language embedded in the programming language — hereafter called DSEL — such as LifeV (*C++*) [1, 56], Sundance (*C++*) [36], Analysa (Scheme, which is suited for embedding sub-languages like other Lisp based languages) [10].

These high level interfaces or languages are desirable for several reasons: teaching purposes, solving complex problems with multiple physics and scales or rapid prototyping of new methods, schemes or algorithms. The goal is always to hide (ideally all) technical details behind software layers and provide only the relevant components required by the user or programmer.

The DSEL approach has advantages over generating a specific language like in case *(i)* : compiler construction complexities can be ignored, other libraries can concurrently be used which is often not the case of specific languages which would have to also develop their own libraries and DSELs inherit the capabilities of the language in which they are written. However, DSELs are often defined for one particular task inside a specific domain [76] and implementation or parts of implementation are not shared between different DSELs.

This article proposes a DSEL for automatic differentiation, projection, integration or vari-

ational formulations. The language implementation uses expression templates [76] and other meta-programming techniques [2]. Related works are [56] and [36], but they differ with the proposed DSEL in many aspects: the former was designed only for variational formulations and requires to write the expression object by hand which can become complicated and error prone, while the latter implements the DSEL in an object oriented way without relying on meta-programming or expression templates.

Other objectives of our DSEL implementation are that it should *(i)* be efficient enough to integrate high performance/parallel software, *(ii)* be generic enough to accommodate different numerical types — for example arbitrary precision, see [65] but we won't discuss these aspects here. — A performance benchmark is available in section 10.4.1.

To illustrate further what the DSEL achieves, here is a comparison between a mathematical formulation of a bilinear form (10.1) and its programming counterpart, see listing 10.1.

$$\begin{aligned} a : X_h \times X_h &\to \mathbb{R} \\ (u, v) &\to \int_\Omega \nabla u \cdot \nabla v + uv \end{aligned} \tag{10.1}$$

Listing 10.1: Variational Formulation in *C++*; the `t` extension of `gradt` and `idt` identifies the trial functions

```
// a mesh of Ω ⊂ R^d, d = 1,2,3
Mesh mesh;
// Finite element scalar space P_K, K = 1,2,3,...
Space<Mesh,FEM_PK<d,K> > Xh(mesh);
// two elements of the Space Xh
Space<Mesh,FEM_PK<d,K> >::element_type u(Xh), v(Xh);
// A matrix in CSR format
csr_matrix_type M;
// bilinear form with M as its matrix representation
// with integration over all elements of the mesh
// and a method for exact integration of polynomials of
// degree ≤ K (IM_PK<d,K>)
BilinearForm<Xh,Xh> a(Xh,Xh,M);
a = integrate( elements(mesh),
               dot(gradt(u),grad(v))+idt(u)*id(v),
               IM_PK<N,K>() );
```

We clearly identify in listing 10.1 the variational formulation stated in equation (10.1). We shall describe the various steps to achieve this level of expression with as little overhead as possible. In section 10.2, we present some concepts concerning mainly integration and variational formulations, then in section 10.3 we present the main points about the DSEL. Finally in section 10.4, we present some non-trivial examples to exercise the language.

This article contains many listings written in *C++* however most of them are not correct *C++* in order to simplify the exposition. In particular, *C++* keywords like `typename` or `inline` are often not present. Also many numerical ingredients such as polynomial approximations,

numerical integration methods used in this article are not described or only very roughly, another publication will cover the mathematical kernel used by the DSEL in more details [65].

10.2 Preliminaries on Variational Forms

In what follows, we consider a domain $\Omega \subset \mathbb{R}^d,\ d = 1, 2, 3$ and its associated mesh $\mathcal{T}$ — out of d-simplices and product of simplices.

10.2.1 Mesh

We present first some tools that will be used later, namely how to extract parts of a mesh and the geometric mapping that maps a convex of reference — where polynomial sets and quadratures are constructed — to any convex of the mesh.

Mesh Parts Extraction

While applying integration and projection methods, it is common to be able to extract parts of the mesh. Hereafter we consider only elements of the mesh and elements faces. We wish to extract easily subsets of convexes out the total set consituting $\mathcal{T}$.

To do this out mesh data structure which is by all means fairly standard uses the Boost.Multi_index library[†] to store the elements, elements faces, edges and points. This way the mesh entities are indexed either by their ids, their markers — material properties, boundary ids. . . , — their location — whether the entity is internal or lies on the boundary of the domain. — Other indices could be certainly defined, however those three allow already a wide range of applications[†].

Thanks to Boost.Multi_index, it is trivial to retrieve pairs of iterators over the entities — elements, faces, edges, points — containers depending on the usage context. The pairs of iterators are then turned into a range, see Boost.Range[†], to be manipulated by the integration and projection tools that will be presented later.

A number of free functions are available that hide all details about the mesh class to concentrate only on the relevant parts.

- `elements(<mesh>)` the set of convexes constituting the mesh
- `idedelement(<mesh>, <id>)` the convex with id `<id>`
- `idedelements(<mesh>, <lower bound>, <upper bound>)` iterator range of convexes whose ids are in the range given by the predicates `<lower bound>` and `<upper bound>`, for example `idedelements(mesh,1000<=_1,_1<5000)`[†]
- `markedelements(<mesh>, <marker>)` iterator range over elements marked with `marker`

[†] http://www.boost.org/libs/multi_index/doc/index.html
[†] Another useful type of indexation could be the process id in a parallel framework.
[†] http://www.boost.org/libs/range/index.html
[†] _1 is part of Boost.Lambda.

- `markedelements(<mesh>, <lower bound>, <upper bound>)` iterator range over elements whose markers are in the range given by the predicates `<lower bound>` and `<upper bound>`, for example

  ```
  markedelements(mesh,1<=_1,_1<5)
  ```

- `faces(<mesh>)` iterator range over all `mesh` element faces
- `markedfaces(<mesh>,<marker>)` iterator range over `mesh` element faces marked with `marker`
- `markedfaces(<mesh>,<lower bound>, <upper bound>)` iterator range over `mesh` element faces whose markers are in the range given by the predicates `<lower bound>` and `<upper bound>`, for example

  ```
  markedfaces(mesh, 1<=_1,_1<5)
  ```

- `bdyfaces(<mesh>)` iterator range over all boundary `mesh` element faces
- `internalfaces(<mesh>)` iterator range over all internal `mesh` element faces

Geometric Mapping

Functional spaces and quadrature methods, for example, are derived from polynomial sets or families that have to be constructed over the convexes of $\mathcal{T}$. Instead of doing this, it is common to construct these polynomials over a reference convex $\hat{T}$— segment, triangle, quadrangle, tetrahedron, hexahedron, prism or pyramid — and provide a geometric mapping or transformation from the reference convex $\hat{T}$ to any convex $T \in \mathcal{T} \subset \mathbb{R}^P, P \leq d$. We need to be able also to transform subentities, such as faces or points, of the reference element to the corresponding entities, faces and points respectively, in the real element.

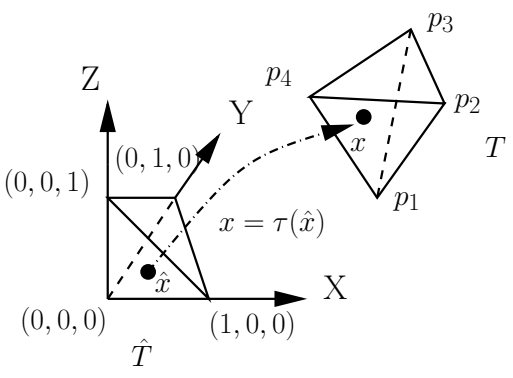


Figure 10.1: Geometric mapping τ from the reference tetrahedron $\hat{T}$ to a real tetrahedron T in 3D.

From now on, we denote with a $\hat{}$ the quantities defined on the reference element. We define $\tau : \mathbb{R}^P \rightarrow \mathbb{R}^N$ that maps $\hat{T}$ to T. We shall denote K_τ its gradient, B_τ its pseudo-inverse and J_τ its jacobian. The geometric mapping is described by *(i)* a n_g components polynomial vector $\{\phi_g(\hat{x})\}_{g=1\ldots n_g}$ and *(ii)* the geometric points $\{p_g\}_{g=1\ldots n_g}$ of T such that

$$x = \tau(\hat{x}) = \sum_{g=1\ldots n_g} \phi_g(\hat{x})\, p_g \qquad (10.2)$$

We denote by $G = (p_1, \ldots, p_{n_g})$ the $N \times n_g$ matrix of geometric nodes. Equation (10.2) and

quantities mentioned above are computed as follows, for any $\hat{x} \in \hat{T}$

$$\begin{aligned} x = \tau(\hat{x}) &= G\ \phi(\hat{x}) \\ K_\tau(\hat{x}) &= G\ \nabla\ \phi(\hat{x}) \\ J_\tau(\hat{x}) &= \begin{cases} \det(K_\tau(\hat{x})) & \text{if } P = N \\ \det(K_\tau^t(\hat{x})K_\tau(\hat{x}))^{1/2} & \text{if } P \neq N \end{cases} \\ B_\tau(\hat{x}) &= \begin{cases} K_\tau^{-t}(\hat{x}) & \text{if } P = N \\ K_\tau(\hat{x})(K_\tau^t(\hat{x})K_\tau(\hat{x}))^{-1} & \text{if } P \neq N \end{cases} \end{aligned} \tag{10.3}$$

where $K_\tau^t(\hat{x})$ denotes the transpose of $K_\tau(\hat{x})$.

Equipped with the geometric mapping concept, we compute an integral on T as an integral on $\hat{T}$: if f is a function defined on T,

$$\int_T f(x)\mathrm{d}x = \int_{\hat{T}} f(\tau(\hat{x}))J_\tau(\hat{x})\mathrm{d}\hat{x} \tag{10.4}$$

and using a quadrature formula:

$$\int_{\hat{T}} f(\tau(\hat{x}))J_\tau(\hat{x})\mathrm{d}\hat{x} \approx \sum_{q=1\ldots Q} \hat{w}_q\ f(\tau(\hat{x}_q))J_\tau(\hat{x}_q) \tag{10.5}$$

where $\{\hat{x}_q, \hat{w}_q\}_{q=1\ldots Q}$ are quadrature nodes and quadrature weights defined in the reference element.

In our framework, the geometric mapping is not used directly by the developer but rather what we call the geometric mapping context which is a subclass of the geometric mapping class. The geometric mapping context is linked to an element T of the mesh such that, given a set of points $\{\hat{x} \in \hat{T}\}$, it provides information for each point in the set $\{x; x \in T \text{ and } x = \tau(\hat{x})\}$ such as the jacobian value $J_\tau(\hat{x})$, the gradient $K_\tau(\hat{x})$ of the mapping, the pseudo-inverse $B_\tau(\hat{x})$ of the gradient; or if the point $\hat{x}$ is on a face of $\hat{x}$, then x is on a face of T and the context provides the normal to the face at this point. A shortened interface of the `Context` class is presented in the listing 10.2.

Listing 10.2: Geometric mapping context class

```
class GeoMap::Context {
public:
  ...
  /**
   * constructor
   * G is column oriented matrix: each column contains
   * the coordinate of a geometric point of T
   * P is column oriented matrix: each column contains
   * the coordinate of the points in reference element
   */
  Context( matrix_node_type G, matrix_node_type P );
  /** return the dimension of the real element */
  int  N ();
```

```
/** return the dimension of the reference element */
int  P ();
/** get the q-th node of P in the reference element */
node_type const&  xRef ( int q ) const;
/** get the q-th node in the real element */
node_type const&  xReal ( int q ) const;
/**  get the value of the jacobian at the q-th node
     in the reference element*/
double J ( int q ) const;
/** get the value of the gradient at q-th node */
matrix_type const&  K ( int q ) const;
/**get the value of the pseudo-inverse at q-th node */
matrix_type const& B ( int q );
/** get the coordinates of the geometric nodes */
matrix_node_type const&  G ();};
```

Another subclass of the geometric mapping class is `Inverse` which as its name state does the inverse of the transformation : given a point x in $T \subset \mathbb{R}^N$ computes its location in the reference element $\hat{x}$ in $\hat{T} \subset \mathbb{R}^P$. `Inverse` is particularly useful for interpolation purposes.

We define a bilinear form $a : X \times Y \to \mathbb{R}$ and a linear form $\ell : X \to \mathbb{R}$, where X and Y are suitable function spaces defined on Ω. The finite element method discretizes X and Y using polynomials spaces defined on $\mathcal{T}$. We denote by $\mathcal{N}_X$ and $\mathcal{N}_Y$ the dimension of the discrete spaces X and Y[†] and by $\{\psi_i\}_{i=1\ldots\mathcal{N}_X}$ and $\{\varphi_i\}_{i=1\ldots\mathcal{N}_Y}$ a basis for X and Y respectively. For any $v \in X$, we have $v = \sum_{i=1\ldots\mathcal{N}} v_i\psi_i$, and similarly for the functions of Y. We can then write the entries $A_{ij}, i = 1\ldots\mathcal{N}_X, j = 1\ldots\mathcal{N}_X$ of the matrix A associated with a and the entries $L_i, i = 1\ldots\mathcal{N}_X$ of the vector L associated with ℓ as follows:

$$\begin{array}{rcll} A_{ij} & = & a(\psi_i, \varphi_j) & i = 1\ldots\mathcal{N}_X, j = 1\ldots\mathcal{N}_Y \\ L_i & = & \ell(\psi_i) & i = 1\ldots\mathcal{N}_X \end{array} \tag{10.6}$$

To construct A and L, we follow a standard assembly process that iterates over the elements T of $\mathcal{T}$ since $a(v,u)$ can be written as $\sum_{T\in\mathcal{T}} a_T(v,u)$ and similarly with ℓ and L. We then introduce *(i)* the restriction of the basis functions to T, $\{\psi_i^T\}_{i=1\ldots N_X}$ and $\{\varphi_i^T\}_{i=1\ldots\mathcal{N}_Y}$ where i is a local numbering over T, and *(ii)* the local to global mappings $\iota_X(\cdot,\cdot)$ and $\iota_Y(\cdot,\cdot)$ between the local numbering of the degrees of freedom and the global one. For example $\iota_X(T,i)$ is the global degree of freedom to which the i-th local degree of freedom of T contributes to. The assembly process is described in the algorithm 3.

In standard finite element software, the assembly is often split into two steps : *(i)* the local matrix $A_T = (a_T(\psi_i^T, \varphi_j^T))_{i=1\ldots N_X, j=1\ldots N_Y}$ and vector $L_T = (\ell_T(\psi_i^T))_{i=1\ldots N_X}$ are first constructed and *(ii)* the local to global mapping is used to add the contribution of the element T to A and L. This splitting is often used to optimize the local to global mappings [12] or optimize the local matrix and vector computation [34]. We will also follow this strategy in the remaining sections.

[†] X and Y will also be named as the test space and the trial space respectively.

Algorithm 3 Standard assembly procedure for A and L.

$A = 0$
for $T \in \mathcal{T}$ **do**
  for $i = 1 \dots N_X$ **do**
    for $j = 1 \dots N_Y$ **do**
      $A_{\iota_X(T,i)\iota_Y(T,j)} = A_{\iota_X(T,i)\iota_Y(T,j)} + a_T(\psi_i^T, \varphi_j^T)$
    end for
    $L_{\iota_X(T,i)} = L_{\iota_X(T,i)} + \ell_T(\psi_i^T)$
  end for
end for

10.2.2 Construction of $a_T(\psi_i^T, \varphi_j^T)$ and $L_T(\psi_i^T)$

We focus now on the construction of the elementary contribution $a_T(\psi_i^T, \varphi_j^T)$ and $\ell_T(\psi_i^T)$ which is the heart of our methodology.

Basis Functions

We turn to the treatment of the basis functions in our framework, and in particular we describe the computation of $f(\tau(\hat{x}))$ for any $\hat{x} \in \hat{T}$ as in equation (10.5). We define the finite element basis functions on the reference element. If f belongs to X, then we have for a given $T \in \mathcal{T}$ and its associated geometric mapping τ :

$$f(\tau(\hat{x})) = \sum_{i=1,\dots,N_X} f_i \ \psi_i(x) \tag{10.7}$$

$$= \sum_{i=1,\dots,N_X} f_i \ \hat{\psi}_i(\hat{x}) \tag{10.8}$$

$$= \underbrace{F^t}_{\text{Expansion coefficients}} \quad \underbrace{\hat{\psi}(\hat{x})}_{\text{Computation on } \hat{T}} \tag{10.9}$$

where $F^t = [f_1, \dots, f_{N_X}]$ and $\hat{\psi}(\hat{x}) = [\hat{\psi}_1(\hat{x}), \dots, \hat{\psi}_{N_X}(\hat{x})]^t$. The gradient reads

$$\nabla \ f(\tau(\hat{x})) = F^t \nabla \psi(x) \tag{10.10}$$

$$= F^t \ B_\tau(\hat{x}) \quad \underbrace{\hat{\nabla} \ \hat{\psi}(\hat{x})}_{\text{Computation on } \hat{T}} \tag{10.11}$$

Similar computations, albeit more involved, might be derived for the second order derivatives.

The basis function concept we developed is similar to the geometric mapping. In our framework, the degrees of freedom are associated with the elements of the mesh. More precisely they are ordered with respect to the geometric subentities of the elements — vertices, edges, faces and volumes — for global continuous functions to ensure a continuous expansion whereas in the case of global discontinuous functions it does not matter how the degress of freedom are ordered or organized within the element. This allows for flexible construction of polynomial sets such as Lagrange, Raviart-Thomas or modal basis with global continuous

expansion or not. The article [65] presents these aspects in details. Essentially the `Basis` base class, see listing 10.3, provides an interface for obtaining the value of the basis function and its derivatives at any given point in the reference element. Similar to the geometric mapping, we also define a `Context` subclass that provides information on the basis functions at a given set of points $\{\hat{x}; \hat{x} \in \hat{T}\}$.

Listing 10.3: Basis functions interface

```
class Basis
{
  public:
  // access to precomputed basis functions values and
  // derivatives at a given set of points
  class Context
  {
    public:
    ...
    // value of the i-th basis function at the q-th node
    // in the reference element : φ^i(x̂_q)
    double phi( int q, int i );
    // gradient of the i-th basis function at the q-th node
    // B^q_τ(x̂_q) ∇̂ ψ^i(x̂_q)
    node_Type const& dphi( int i );
    ...
  }
};
```

Equipped with these tools and concepts and if we consider a function $f \in X$, we have

$$\begin{aligned} \int_T f(x)\,\mathrm{d}x &= \sum_{q=1\ldots Q} \hat{w}_q \sum_{i=1\ldots N_X} f_i\ \hat{\psi}_i(\hat{x}_q)\ J_\tau(\hat{x}_q) \\ \int_T \nabla f(x)\,\mathrm{d}x &= \sum_{q=1\ldots Q} \hat{w}_q \sum_{i=1\ldots N_X} f_i\ B_\tau(\hat{x}_q)\ \hat{\nabla}\hat{\psi}_i(\hat{x}_q)\ J_\tau(\hat{x}_q) \end{aligned} \tag{10.12}$$

Approximation Space

We define the notion of approximation space in *C++* that maps closely the mathematical counterpart. An approximation space is a template class parametrized by a mesh class and the basis functions type — for example the standard Lagrange finite elements. — An approximation space wraps the mesh, the table of degrees of freedom (DoF), the basis function type and provides access to all them. Note that the geometric mapping is provided by the mesh class.

Listing 10.4: Approximation Space Interface

```
template<typename Mesh, typename Basis_t>
class Space
{
  // get the mesh
  Mesh const& mesh() const;
  // get the dof data structure
```

```
  Dof const& dof();
  // get the basis function data structure
  Basis_t const& basis() const;
  // get the geometric mapping
  Mesh::GeoMap const& geomap() const;
};
// ℙ1 finite element space in 2D
Space<Mesh, FEM_PK<2,1,scalar> > Xh( mesh2d );
// ℙ2^3 finite element space in 3D
Space<Mesh, FEM_PK<3,2,vectorial> > Xh( mesh3d );
```

A Space defines its own element type as a subclass: it ensures coherence and consistency when manipulating finite element functions. An Element derives from your preferred numerical vector type: we use uBLAS[†] for our linear algebra data structures and algorithms. The interface is roughly described in the listing 10.5.

Listing 10.5: Approximation Space Interface

```
enum ComponentType { X = 0, Y, Z };
template<typename Mesh, typename Basis_type>
class Space
{
 ...
 // Element subclass
 template<typename T>
 class Element : public ublas::vector<T>
 {
   public:
   ...
   space_type const& space() const;
   component_space_type const& compSpace() const;

   // get the component of the vector
   component_type comp( ComponentType ) const;
   // interpolation of the function at a given point
   T operator()( node_type const& ) const;

   // Sobolev norms
   double normL2() const;
   double normH1() const;
   ...
 };
};
```

An extension of the Space concept, is the MixedSpace which is a product of two spaces. This can actually be extended to a product of several spaces of different types — implemented using the MPL [2]. — This concept is useful for mixed formulations. MixedSpace defines also its own element type with some extra member to retrieve the underlying space elements.

[†] http://www.boost.org/libs/numeric/ublas/doc/index.htm

Listing 10.6: Mixed Space Example

```
// P2 − P1 approximation space in 2D
typedef mpl::vector<FEM_PK<2,2,vectorial>,
                    FEM_PK<2,1,scalar> > SpaceList;
typedef MixedSpace<Mesh, SpaceList > space_t;
space_t Vh( mesh );
space_t::element_type U( Vh );
// FEM_PK<2,2,vectorial>
space_t::element_1_type u = U.element1();
// FEM_PK<2,1,scalar>
space_t::element_2_type u = U.element2();

// extract a view of the X component of u
space_t::element_1_type::component_type ux = u.comp(X);
```

At the moment, `MixedSpace` is concept for a product of two functional spaces. Extending this concept to a product of N functional spaces would be useful.

Linear and Bilinear Forms

One last concept needed to have the language expressive is the notion of forms. They follow closely their mathematical counterparts: they are template classes with arguments being the space or product of spaces they take their input from and the representation we can make out of these forms. In what follows, we consider only the case where the linear and bilinear forms are represented by vectors and matrices respectively. In a future work, we will eventually propose the possibility to have vector-free and matrix-free representations: that would require to *store* the definition of the forms.

Listing 10.7 displays the basic interface and usage of the form classes.

Listing 10.7: Forms

```
Mesh mesh;
// P3(Ω ⊂ R^3)
typedef Space<Mesh,FEM_PK<3,3> > Space_t;
Space_t X_h(mesh);
Space_t::element_type u(X_h), v(X_h);

// Linear forms
template<typename Space, typename Rep>
class LinearForm
{
  LinearForm( Space const&, Rep& rep ) {...}
  LinearForm& operator=( Rep const& ) {...}
  template<typename Expr>
  LinearForm& operator=( Expr const& ) {...}
};
```

```
typedef ublas::vector<T> linearform_rep;
Linearform_rep F;
LinearForm<Space_t,linearform_rep> f(X_h,F);
f=integrate(elements(mesh),id(v));

// BiLinear forms
template<typename Space1, typename Space2,
         typename Rep>
class BiLinearForm
{
  BilinearForm( Space1 const&, Space2 const&,
                Rep& rep ) {...}
  BilinearForm& operator=( Rep const& ) {...}
  template<typename Expr>
  BilinearForm& operator=( Expr const& ) {...}
};


typedef ublas::compressed_matrix<T> bilinearform_rep;
Bilinearform_rep M;
BilinearForm<Space_t,Space_t,bilinearform_rep> a(X_h,X_h,
                                                 M);
a=integrate(elements(mesh),idt(u)*id(v));
```

Note that the linear and bilinear form classes are the glue between their representation and the mathematical expression given by `Expr`, it will

- fill the matrix with non-zero entries depending on the approximation space(s) and the mathematical expression;
- allow a per-component/per-space construction(blockwise);
- check that the numerical types of the expression and the representation are consistent
- when `operator=( Expr const& )` is called, the expression is evaluated and will fill the representation entries

The concepts of `MixedLinearForm` and `MixedBilinearForm` that would correspond to mixed linear and bilinear forms respectively — taking their values in the product of two functional spaces — exist also and follow the same ideas.

With the high level concepts described we can now focus on the language.

10.3 Language

The expression template technique won't be described as it is nowadays a mainstream technique [76, 2, 8, 9, 56]. The construction of the expression template objects in the coming sections is standard.

10.3.1 Expression Evaluation at a Set of Points in a Convex

Let C be a convex in $\mathbb{R}^d, d \leq 1, 2, 3$ — a n-simplex $n \leq d$ like lines, triangles or tetrahedrons or products of simplices like quadrangles, hexahedrons or prisms, — and $\hat{C}$ be a convex of reference in $\mathbb{R}^d, d \leq 1, 2, 3$ associated to C where we define quadrature points for integration or points to construct polynomials for finite elements and other approximation methods, see [22, 32].

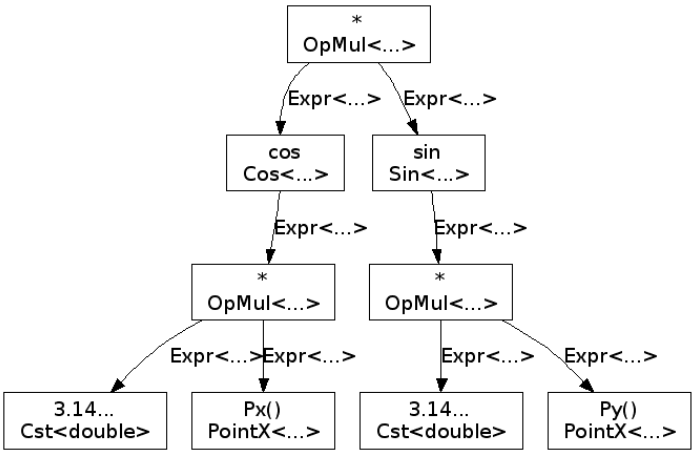


Figure 10.2: Expression template graph for $f : x \rightarrow cos(\pi x)sin(\pi y)$

We wish to evaluate $f(x), \forall x \in \tau(\hat{S}_P) = \{x_1, \ldots, x_P\} \subset C$, $\hat{S}_P = \{\hat{x}_1, \ldots, \hat{x}_P\} \subset \hat{C}$, f is a real-value function $C \rightarrow \mathbb{R}$ and τ is the geometric mapping $\hat{C} \rightarrow C$, see 10.2.1.0.

In our code f is represented by an expression template – and not a standard C++ function or a functor, — see [76]. For example, consider $f : x \in C \rightarrow \cos(\pi x)\sin(\pi y)$, we write it in C++ as `cos(π*Px())*sin(π*Py())`. The expression graph is shown on figure 10.2. Here `Px()` and `Py()` are free functions that construct objects that are evaluated as the `x` and `y` coordinates of the points $x; x \in C$.

Constructing the C++ object that represents the expression is done with standard expression template approach. However evaluating the expression is problematic as some ingredients are not known yet to the expression object such as the geometric mapping. So using a standard expression template approach certainly allow high level expressivity but cannot be applied to evaluate the expression.

To remedy this issue, we propose a very simple but very powerful solution which delegates the evaluation of the expression to another object than the expression object itself. In our case, the evaluation is delegated to a sublass of each object of the expression.

The `Expression` class, which is the glue between the various object types forming the expression template, is roughly sketched in listing 10.8.

Listing 10.8: Evaluation Delegation to Subclass

```
template<typename Expr>
class Expression
{
  typedef Expression<Expr> self_type;
  Expr const& expression() const { return expr; }

  template<typename GeoContext_t>
  class Eval
  {
    typedef Expr::Eval<GeoContext_t> evaluator_type;
```

```
    // construct the evaluator for expression Expr.
    Eval( self_type const& _expr,
          GeoContext_t const& gmc )
    : evaluator(_expr.expression,gmc)  {}

    // evaluatate at the q-th node used to build the
    // the finite element
    double operator()( int q ) const { return eval( q ); }
    evaluator_type eval; }; // Eval
  Expr expr; }; // Expression
```

Expression<Expr>::Eval is a template class parametrized by the geometric mapping context associated with each geometric element of the mesh. The constructor takes the expression Expr and the geometric mapping context as arguments to pass geometric data — coordinates of the current point, normals, measure of the element — down to all objects of the expression so that they can use it as needed. As already mentioned, Px() constructs a *C++* class that returns the x coordinate of the points where the evaluation is effected. Its implementation is presented in2 listing 10.9.

Listing 10.9: Current Evaluation Point Coordinates

```
class PointX
{
  typedef PointX self_type;

  template<typename GeoContext_t>
  class Eval
  {
    // construct the evaluator for expression Expr.
    Eval( self_type const& /*_expr*/,
          GeoContext_t const& gmc )
    : _M_gmc( gmc )  {}

    // returns the x coordinate of the q-th node
    // stored in the geometric mapping context
    double operator()( int q ) const
    { return gmc.xReal(q)[0]; }
    GeoContext_t gmc;
  }; // Eval
}; // Expression
Expression<PointX> Px()
{ return Expression<PointX>( PointX() ); }
```

Py() is implemented in a similar way. Regarding the mathematical functors cos and sin, they also follow the same idea as shown in listing 10.10.

Listing 10.10: Current Evaluation Point Coordinates

```
template<typename Expr>
class Cos
```

```
{
  typedef Cos<Expr> self_type;

  template<typename GeoContext_t>
  class Eval
  {
    typedef Expr::Eval<GeoContext_t> eval_expr_type;
    // construct the evaluator for expression Expr.
    Eval( Expr const& _expr,
          GeoContext_t const& _gmc )
    : eval_expr( _expr,gmc )  {}

    // returns the cosinus of the expression at the q-th
    // node stored in the geometric mapping context
    double operator()( int q ) const
    { return cos( eval_expr( q ) ); }
    eval_expr_type eval_expr;
  }; // Eval
}; // Expression
template<typename Expr>
Expression<Cos<Expr> > cos( Expr const& e )
{ return Expression<Cos<Expr> >( Cos<Expr>( e ) ); }
```

10.3.2 Nodal Projection

We described the mechanism to evaluate an expression at a set of points in a convex, we now turn to nodal projection of a function f onto an approximation space X — for example $X = \{u \in \mathbb{P}_k(\mathcal{T})\}$ where $\mathcal{T}$ is a triangulation of Ω and $\mathbb{P}_k$ is the set spanned by the Lagrange polynomials of degree $\leq k$. — We denote $\pi_X f$ the nodal projection of f onto X.

The nodal projection is an extension of the previous section at a set of convexes and $\hat{S}_P$ being the set of coordinates of the degrees of freedom (DoF) associated with X. The nodal projection is described by algorithm 4.

Algorithm 4 Nodal projection on X

ι_X is the local/global correspondance table
for $T \subset \mathcal{T}$ **do**
  $\hat{p}_i = 1, \ldots, N_X$ points coordinates associated with the DoF in T
  $c \leftarrow \{T, G, (\hat{p}_i)_{i=1\ldots N_X}\}$ geometric mapping context, see 10.2.1.0
  for $i = 1, \ldots, N_X$ points coordinates associated with the DoF **do**
    $c.\hat{x} \leftarrow \hat{p}_i$
    $\pi_X f_{\iota_X(T,i)} = f(c.x)$
  end for
end for

We define a free function `project(<space>,[elements,]<expression>)` that takes two or three arguments : the approximation space onto which we project the function, the expression

representing the function we want to project and optionally a range of elements that restricts the projection to this set of elements, see section 10.2.1.0. `project()` constructs an template class parametrized by the arguments types passed to `project()`, see listing 10.11.

Listing 10.11: Nodal Projection

```
// Space_t type of approximation space
// Erange range of iterators over the elements that
//        restricts the projection
// Expr_t expression to project
template<typename Space_t, typename Erange,
         typename Expr_t>
class Projector
{
  Projector( Space_t const& X, Erange const& erange,
             Expr_t const& E );
  Space_t::element_type operator()() const {...};
};
template<typename Space_t, typename Expr_t>
Space_t::element_type
project( Space_t const& X, Expr_t const& E )
{
  // projection of E over all elements of the mesh
  return project( X, elements(X.mesh()), E );
}
template<typename Space_t, typename ERange,
         typename Expr_t>
Space_t::element_type
project( Space_t const& X, ERange const& erange,
         Expr_t const& E )
{
  Projector<Space_t, Expr_t> P(X, erange, E);
  return P();
}
```

Listing 10.12 shows an example of nodal projection.

Listing 10.12: Nodal Projection Example

```
Mesh mesh; // mesh of Ω ⊂ ℝ^d
// Space of cubic lagrange element
Space<Mesh,FEM_PK<d,3> > X(mesh);
X::element_type u;
// project cos(πx)sin(πy) on ℙ3(Ω)
u = project( X, cos(π*Px())*sin(π*Py());
```

Other types of projection like L_2 or H_1 projections require other ingredients presented in the coming sections.

10.3.3 Numerical Integration

We now turn to numerical integration of $\int_\Omega f(x)\mathrm{d}x$ where f is the function to be integration over Ω. Numerical integration requires the evaluation of the function f at quadrature points in the convexes of the mesh associated to Ω. In our code, we used the quadrature constructions presented in [32] for n-simplices and simplices products.

The integration process is described by algorithm 10.3.3

Algorithm 5 Integration over a mesh $\mathcal{T}$ of a domain $\Omega \subset \mathbb{R}^d$ using a Quadrature Method

$(\hat{w}_q, \hat{x}_q)_{q=1,\dots Q}$ be the set of quadrature nodes and weights
for $T \subset \mathcal{T}$ **do**
  Set $c \leftarrow \{T, G, (\hat{x}_q)_{q=1\dots Q}\}$ geometric mapping context, see section 10.2.1.0
  $\int_\Omega f(x) += \sum_{q=1,\dots Q} \hat{w}_q f(\tau(\hat{x}_q))$ $\{\tau(\hat{x}_q)$ is given by `c.xReal(q)`$\}$
end for

We introduce a new keyword to reflect the integration action, see listing 10.13, which is a free function instantiating an integrator parametrized by *(i)* the set of geometric elements, see section 10.2.1.0, where the integration is done, *(ii)* the expression to integrate and *(iii)* the integration method, see listing 10.14.

Listing 10.13: `integrate` prototype

```
integrate( <elements>, <expr>, <integration method> )
```

Listing 10.14: `Integrator` class and `integrate` free function

```
template<typename EIter, typename Expr_t,typename Im_t>
class Integrator
{
  Integrator( Elist const& elist, Expr_t const& E,
              Im_t const& im );
  double operator()() const {...};
};
template<typename EIter, typename Expr_t,typename Im_t>
double
integrate( EIter const& eiter,
           Expr_t const& e,
           Im_t const & im )
{
  Integrator<EIter, Expr_t, Im_t> I(eiter, e, im );
  return I();
}
```

Listing 10.15 shows an example of the syntactic sugar brought by the language.

Listing 10.15: Integration Syntax

```
// mesh of [0,1]^d, d = 1,2,3 made of simplices
// of dimension d
Mesh mesh;
```

```
// Gauss Legendre integration of Order 10
// over simplex of dimension d
IM_PK<d,10> im;
// ∫_[0,1]^d cos(‖x‖_2) where ‖·‖_2 is the Euclidean norm
// elements(mesh) provides the list of all elements
// in the mesh data structure
v=integrate( elements(mesh), cos(norm2(P())), im );
// ∫_∂[0,1]^d cos(‖x‖_2)
// bdyfaces(mesh) provides the list of the element faces
// on the boundary of the mesh
v=integrate( bdyfaces(mesh), cos(norm2(P())), im );
```

10.3.4 Variational Formulations

The framework, presented in the last sections, can be extended to handle variational formulation with only minor changes to the evaluation class. We consider for now a convex $C \subset \mathbb{R}^d, d = 1, 2, 3$ and its associated reference convex $\hat{C}$, an approximation space X — for example $\mathbb{P}_k(C)$ — a bilinear form $a\ X \times X \to \mathbb{R}$ defined by

$$a(u, v) = \int_C u\ v \quad \forall\ u,\ v \in X \tag{10.13}$$

Listing 10.16 shows the C++ counterpart of equation (10.13).

Listing 10.16: Variational Integration

```
// integrate over element with id 1
integrate( idedelement(mesh,1), idt(u)*id(v) );
```

The `t` in `idt(.)` allows to distinguish trial and test functions: for example, `id(.)` identifies the test function values whereas `idt(.)` identifies the trial function values.

Given $u, v \in X$, we wish to compute the value $a(u, v)$ which can be approximated as follows

$$a(u, v) \approx \sum_{i=1\ldots N_X} \sum_{j=1\ldots N_X} u_i\ v_j \sum_{q=1\ldots Q} \hat{w}_q\ \hat{\psi}_i(\hat{x}_q)\ \hat{\psi}_j(\hat{x}_q) J(\hat{x}_q) \tag{10.14}$$

where $(\hat{w}_q, \hat{x}_q)_{q=1\ldots Q}$ are the quadrature nodes and weights defined in $\hat{C}$, $J(\hat{x}_q)$ is the jacobian of the geometric transformation between $\hat{C}$ and C at the point $\hat{x}_q$ and $(\hat{\psi}_i)_{i=1\ldots N_X}$ is the basis of X.

Recall section 10.2.2.0, we have the basis context subclass that allows to store values and derivatives of basis functions at a set of points. In the case of equation (10.14), the basis context subclass stores and provides an interface to $(\hat{\psi}_i(\hat{x}_q))_{q=1\ldots Q, i=1\ldots N_X}$.

Again the basis functions context is not known to the expression object. In order to accommodate the language with these concepts, it suffices to add new template parameters to the evaluation subclass of each classes allowed in an expression. However these parameters have default values that allows to handle the case of the previous section as well as linear forms and bilinear forms, see listing 10.17. In particular test basis functions context type are defaulted to `boost::none_t`[†] and trial basis functions context type to the test ones. If

[†] See `boost/none_t.hpp`

boost::none_t is used, then no language keyword may be used in the expression that will need basis functions operators such as id(.) or idt(.).

Listing 10.17: Evaluation subclass modifications for Variational Formulations

```
template<typename Expr>
class Expression
{
  typedef Expression<Expr> self_type;
  Expr const& expression() const { return expr; }

  template<typename GeoContext_t,
           typename Basis_test_t = boost::none_t,
           typename Basis_trial_t = Basis_test_t>
  class Eval
  {
    typedef Expr::Eval<GeoContext_t> evaluator_type;


    // construct the evaluator for expression Expr.
    Eval( self_type const& _expr,
          GeoContext_t const& gmc,
          Basis_trial_t const& u
          Basis_test_t const& v )
    : eval(_expr.expression,gmc,u,v)  {}

    // construct the evaluator for expression Expr.
    Eval( self_type const& _expr,
          GeoContext_t const& gmc,
          Basis_test_t const& v )
    : eval(_expr.expression,gmc,v)  {}

    ...

    // evaluatate at the ith-basis function and
    // j-th basis function at the q-th node used
    // to build the finite element
    double operator()( int q, int i, int j ) const
    { return eval( q, i, j ); }

    // evaluatate at the ith-basis function and
    // j-th basis function at the q-th node used
    // to build the finite element
    double operator()( int q, int i, int j ) const
    { return eval( q, i ); }

    ...

    evaluator_type eval; }; // Eval
```

```
  Expr expr; }; // Expression
```

Let's examine for example the implementation of the operator dx(<element:u>) which provides the first component of the first derivative of the basis function associated with u, see listing 10.18.

Listing 10.18: Operator dx(.)

```
template<typename Element>
class OpDx
{
  typedef Element element_type;
  typedef Element::basis_type basis_type;

  template<typename GeoContext_t,
           typename Basis_test_t,
           typename Basis_trial_t = Basis_test_t>
  class Eval
  {
    typedef Expr::Eval<GeoContext_t> evaluator_type;

    // construct the evaluator for expression Expr.
    Eval( self_type const& _expr,
          GeoContext_t const& gmc,
          Basis_trial_t const& u
          Basis_test_t const& v )
    : test_basis(v)  {}
    // construct the evaluator for expression Expr.
    Eval( self_type const& _expr,
          GeoContext_t const& gmc,
          Basis_test_t const& v )
    : test_basis(v)  {}
    ...
    // if the data type for test basis functions
    // are the same then
    //  return the first component of the first
    //  derivative
    // otherwise return 0
    double operator()( int q, int i, int j ) const
    {
      // the if disappears at compile time since
      // the condition is known
      if ( boost::is_same<basis_type,
                          Basis_test_t>::value )
        return test_basis.dphi(q,j)[0];
      else
        return 0;
    }
    double operator()( int q, int i ) const
```

```
    { same as above }
    ...
    Basis_test_t test_basis; }; // Eval
  Expr expr; }; // Expression
```

The types of the test and trial basis functions, `Basis_test_t` and `Basis_trial_t` are tested with respect to the basis function type `basis_type` of the element passed to the operator. If the types are not the same then at *compile time* the evaluation of this operator returns 0. At first glance for standard scalar equations — heat equation say — it allows just to ensure that the evaluation makes sense, however when dealing with mixed formulation — e.g. Stokes equations — this feature becomes very important if not crucial for correctness and performance-wise. Indeed it will disable automatically the terms associated with the basis functions which are not evaluated during local assembly of the mixed formulation : for example when filling the block corresponding to the velocity space, all the others terms — velocity-pressure and pressure terms — are evaluated to 0.

An immediate remark is that we may have lots of computation for nothing, i.e. computing 0. However a simple check with g++ -O2 --save-temps shows that g++ optimizes out expressions containing 0 *at compile time* and removes the corresponding code. The *compile time check* of the basis functions types is then crucial to get the previous g++ optimization[†].

As mentionned in the introduction, DSEL inherits the capabilities of the underlying language; we have shown here a very good illustration of this statement.

Now that we know how to do integrate a variational form on a convex, it is easy to extend it to a set of convexes.

10.3.5 Automatic Differentiation

Automatic differentiation as described in [9, 8] operates differently from the previous algorithms when evaluating an expression template. We are no longer evaluating an expression at a set of points but rather dealing with expressions that manipulate a differentiation numerical type `ADType<P,N>` — P is the number of parameters and N the order of differentiation — holding a value and corresponding derivatives — in pratice up to order $N = 2$ and $P \leq 100$ —

To implement the automatic differentiation engine, we construct the expression object as before. However the evaluation is changed: in our case we create a new subclass for evaluating the differentiation. Each class which could be possibly used for automatic differentiation need to implement this subclass. Listing 10.19 shows an excerpt of the implementation of the expression templates glue class `Expression`.

Listing 10.19: Automatic differentiation subclass

```
template<typename Expr>
class Expression
{
  typedef Expression<Expr> self_type;
  Expr const& expression() const { return expr; }
```

[†] This g++ optimization is a combination of at least two optimization strategies: *(i)* constant folding and *(ii)* algebraic simplifications & Reassociation, see http://www.redhat.com/software/gnupro/technical/gnupro_gcc.html for more details

```
// evaluation of expression at a set of point
template<...> class Eval {};

// differentiate expression
class Diff
{
  typedef Expr::Diff diff_type;

  // construct the differentiator for expression Expr.
  Diff( self_type const& _expr )
  : diff(_expr.expression())  {}

  double value() const
  { return diff.value(); }

  double grad( int __ith ) const
  { return diff.grad( __ith ); }

  double hessian( int __i, int __j ) const
  { return diff.hessian( __i, __j ); }

  diff_type diff; }; // Diff
Expr expr; }; // Expression
```

The automatic differentiation evaluation engine does not need to be much further discussed as it follows ideas already published elsewhere, see [9]. We just demonstrate here that decoupling expression construction from its evaluation allows to share the code constructing the expression template object.

A Remark on mixing automatic differentiation and variational formulations We could actually push the envelop quite a lot and use the automatic differentiation type within a variational formulation to differentiate with respect to some parameters for sensitivity analysis, optimization or control of engineering components. That is to say, mix the two evaluation engines : first the integral expression is evaluated and during the integral evaluation, the automatic differentiation language takes over to finalize it. The enabler of the second stage in the evaluation would be the automatic differentation data type and its `operator=( Expr const& )`. This is truly what meta-programming is about — code that generates code that generates code... — However this has not been tested in the examples shown in section 10.4 and in particular performance could be a concern depending on the quality of the generated code. This is one of the topics for future research as it may open important areas of applications for the language such as the ones mentionned previously — sensitivity analysis, optimization and control.

This leads to another advantage of sharing the expression object construction: it ensures that we have the same set of supported mathematical functions or functors for automatic differentation on the one hand and projection, integration on the other hand which means less development work and less bugs.

10.3.6 Overview of the Language

The implementation of the language itself, the construction of the expression object is done in a very standard way conceptually, see [76]. However from a technical point, it uses state of the art tools such as the Boost.Mpl [2] and Boost.Preprocessor [2, 33].

In particular the Boost.preprocessor library is extremely useful when it comes to generate the objects and functions of the language from a variety of numerical types and operators — using for example the macro

`BOOST_PP_LIST_FOR_EACH_PRODUCT`.

Regarding the grammar of the language, it follows the *C++* one for a specific set of keywords wich are displayed in tables 10.1, 10.2 and 10.3. The keywords choice follows closely the Freefem++ language, see [29].

`H()`	diameter of the current element
`Hface()`	diameter of the current face
`Emarker()`	current element marker
`Eid()`	current element id
`N()`	unit outward normal at the current node of the current face
`Nx(),Ny(),Nz()`	x, y and z component of the unit outward normal
`P()`	coordinates of the current node
`Px(),Py(),Pz()`	x, y and z component of current node

Table 10.1: Tables of geometric/element operators available at the current element, face and node

A note on the supported numerical types The language supports the standard ones available in *C++*— boolean, integral and floating-point types — and some additional ones

- `std::complex<>` complex data type
- `ADtype<.,.>` the automatic differentiation type
- `dd_real` double-double precision data type from the QD library, see [30]
- `qd_real` quad-double precision data type from the QD library
- `mp_real` arbitrary precision data type from the ARPREC library, see [11]

Other data types can be relatively easily supported thanks to Boost.Preprocessor. There are plans for example to support an interval arithmetic data type — the one provided by Boost.Interval, — which would be interesting to measure, for example, the impact of small perturbations or uncertainty on the results.

`*,+,/,-,!,||,&&,...`	standard unary and binary operations
`cos(<expr>)`	trigonometric/hyperbolic functions
`sin(<expr>)`	
`...`	
`exp(<expr>)`	exponential function
`log(<expr>)`	logarithmic function
`abs(<expr>)`	absolute value of expression
`floor(<expr>)`	floor of expression
`ceil(<expr>)`	ceil of expression
`chi(<expr>)`	Heaviside step function
`min(<expr>,<expr>)`	min/max
`max(<expr>,<expr>)`	
`pow(<expr>,<expr>)`	expression to the power
`<expr>^(<expr>)`	
`dot(<expr>,<expr>)`	dot product of two vectorial expressions
`jump(<expr>)`	jump of an expression across a face of an element
`avg(<expr>)`	average of an expression across a face of an element

Table 10.2: Tables of mathematical functions that can be applied to expressions

`id(<element:u>)`	basis functions associated to `u`
`grad(<element:u>)`	gradient of the basis functions associated to `u`
`dx(<element:u>)`	x,y and z components of the gradient of the
`dy(<element:u>)`	basis functions associated to `u`
`dz(<element:u>)`	
`dxx(<element:u>)`	components of the hessian of the basis
`dyy(<element:u>)`	functions associated to `u`
`...`	
`grad(<element:u>)`	gradient operators of basis functions associated to `u`
`div(<element:u>)`	divergence operator of basis functions associated to `u`
`idt(<element:u>)...`	the previous operators with a suffix `t` to specify the trial basis functions

Table 10.3: Tables of Operators for variational formulations

10.4 Test Cases

We present now various examples to illustrate the techniques developed previously. We shall show only the relevant part of each examples. Also, for each example, we shall present different possible formulations.

10.4.1 Performance

In the following section, various results are presented with some timings for the construction of bilinear and linear forms. The calculations have been conducted on an AMD64 opteron [†] running linux `2.6.9` and GNU/Debian/Linux.

The `g++-4.0.1` compiler was used to compile the code with the following options

Listing 10.20: g++ options for optimization

```
# standard options
-march=opteron -fast-math -O2
-funroll-loops
# aliasing
-fstrict-aliasing -fargument-noalias-global
```

These options are relatively standard and give good results all the time. Other optimization options have been tried but did not yield significant performance improvements.

We consider various Advection-Diffusion-Reaction problems to evaluate the performance of our framework proposed in [56].

$$\mu\Delta u = 0 \quad D, \tag{10.15}$$

$$\mu\Delta u + \sigma u = 0 \quad DR, \tag{10.16}$$

$$\mu\Delta u + \beta\Delta\nabla u + \sigma u = 0 \quad DAR \tag{10.17}$$

on a unit square and cube domain.

For each problem we consider both the case of constant and space-dependent coefficients. The following expressions were assumed for the space-dependent coefficients

$$\mu(x,y,z) = \begin{cases} x^3+y^2 & (\mathbf{2D}) \\ x^3+y^2z & (\mathbf{3D}) \end{cases} \tag{10.18}$$

$$\beta(x,y,z) = \begin{cases} (x^3+y^2z, x^3+y^2) & (\mathbf{2D}) \\ (x^3+y^2z, x^3+y^2, x^3) & (\mathbf{3D}) \end{cases} \tag{10.19}$$

$$\sigma(x,y,z) = \begin{cases} x^3+y^2 & (\mathbf{2D}) \\ x^3+y^2z & (\mathbf{3D}) \end{cases} \tag{10.20}$$

The corresponding *C++* code for the 3D is presented in listing 10.21. The *C++* for the 2D cases is very similar but simpler.

[†] AMD Opteron(tm) Processor 248, 2.2Ghz, 1MB cache, 8GB ram

Listing 10.21: Benchmark for elliptic problems

```
#define gradugradv(u,v) (dxt(u)*dx(v) + dyt(u)*dy(v) + \
                         dzt(u)*dz(v))
#define agradu(a,b,c,u) ((a)*dxt(u) + (b)*dyt(u) + \
                         (c)*dzt(u))

// D
D = integrate(elements(mesh), gradugradv( u,v ) );
D = integrate(elements(mesh), (Px()^(3)+Py()^(2)*Pz())*
                               gradugradv(u,v));

// DR
DR = integrate( elements(mesh), gradugradv(u,v )  +
                                idt(u)*id(v));
DR = integrate( elements(mesh), (Px()^(3)+Py()^(2)*Pz())*
                (gradugradv(u,v) + idt(u)*id(v)));

// DAR
DAR = integrate( elements(mesh), gradugradv(u,v) +
                 idt(u)*id(v) + agradu(1,1,1,u)*id(v));
DAR = integrate( elements(mesh),
               (Px()^(3)+Py()^(2)*Pz())*(gradugradv(u,v) +
               idt(u)*id(v))+
               agradu((Px()^(3)+Py()^(2)*Pz()),
                      (Px()^(3)+Py()^(2)),
                      (Px()^(3)),u) * id(v) );
```

Results

The benchmark table, see 10.4, reports the timings for filling the matrix entries associated with the problems D, DR and DAR. Lagrange element of order 1 — 3 dof in 2D, 4 in 3D — and order 2 — 6 dof in 2D, 10 dof in 3D — have been tested. The integration rule changes depending on the polynomial order we wish to integrate exactly. For $\mathbb{P}1$ Lagrange elements 3 quadrature points are used in 2D, 4 in 3D and for $\mathbb{P}2$ Lagrange elements 4 are used in 2D and 15 in 3D.

Analysis

In order to facilitate the study of these timing results, they are displayed on figures 10.3 and 10.4 for the 2D cases and 3D cases respectively with each time the matrix assembly time with respect to the number of elements along with the ratio between the cst timings and xyz timings for $\mathbb{P}1$ and $\mathbb{P}2$ cases. We can observe a few things:

- matrix assembly time versus number of elements is linear (in $\log - \log$) which is no surprise,

Dim	NEls	$\mathbb{P}$k	NDoF	D		DR		DAR	
				const	xyz	const	xyz	const	xyz
2D	20742	$\mathbb{P}1$	10570	0.08	0.13	0.08	0.13	0.13	0.17
		$\mathbb{P}2$	41881	0.2	0.23	0.2	0.23	0.25	0.29
	82460	$\mathbb{P}1$	41 629	0.36	0.48	0.37	0.48	0.51	0.66
		$\mathbb{P}2$	165717	0.81	0.99	0.83	1.02	1.07	1.42
	330102	$\mathbb{P}1$	165850	1.47	2	1.5	2	2.06	2.6
		$\mathbb{P}2$	661801	3.38	4.06	3.42	4.22	4.22	5.45
	517454	$\mathbb{P}1$	259726	2.44	3.25	2.48	3.24	3.29	4.28
		$\mathbb{P}2$	1036905	5.5	6.57	5.86	7.17	6.98	8.78
3D	8701	$\mathbb{P}1$	1723	0.06	0.07	0.06	0.07	0.08	0.11
		$\mathbb{P}2$	12825	0.42	0.46	0.43	0.48	0.49	0.58
	69602	$\mathbb{P}1$	12239	0.49	0.73	0.5	0.75	0.69	1.09
		$\mathbb{P}2$	96582	3.47	3.77	3.6	4.22	4.3	4.9
	554701	$\mathbb{P}1$	92071	4.05	5.88	4.55	6.36	5.96	8.55
		$\mathbb{P}2$	748575	31.27	34.42	32.35	36.16	35.82	42.08
	1085910	$\mathbb{P}1$	178090	8.59	12.32	8.96	12.62	11.74	17.54

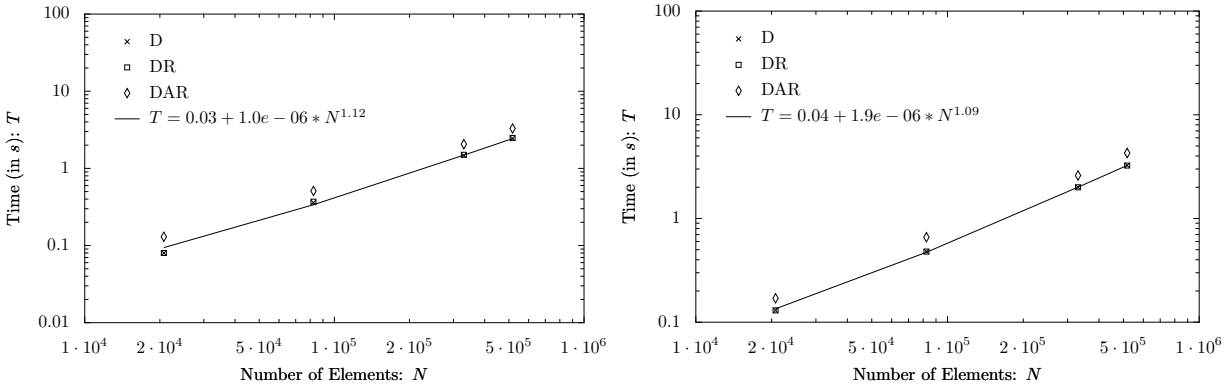
Table 10.4: Language Performances in $[0,1]^2$ and $[0,1]^3$. Timings are in seconds.

- the difference of performances between the different equations D, DR and DAR are quite small even with the DAR and the 3D cases which means that we do a good job at sharing as much computations as possible between the different terms of the equation,

- the overhead due to non-constant coefficients is very small in all cases. which is not the case for example in [56]. In particular, as shown by the ratio figures, the ratios are always less than 2 and in most cases — among them the most expensive ones — they are in $[1.1; 1.3]$. In [56] they used functions or functors to treat the non constant coefficients, they get factors between 2 and 5 (5 in the worst case 3D $\mathbb{P}_2$). This means that the expression templates mechanism and the g++ optimizer do a very good job at optimizing out the expression evaluation.

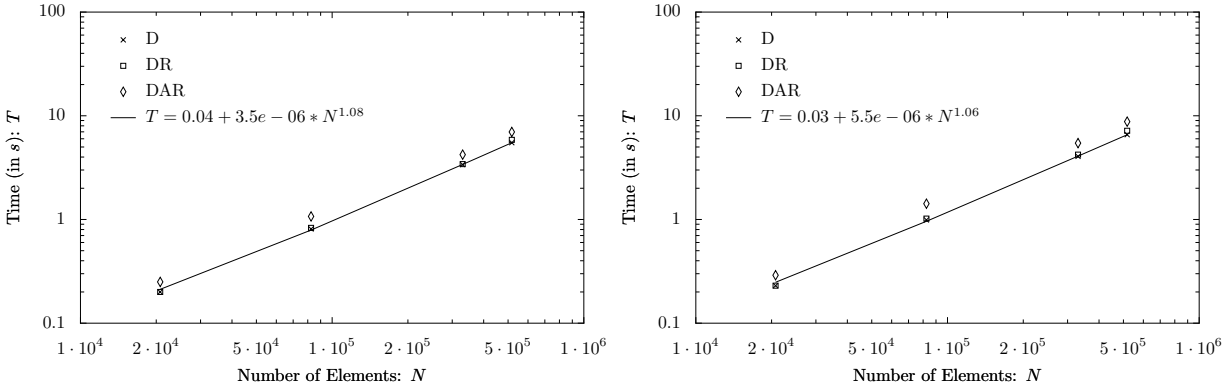
These results illustrate very well the pertinence of using meta-programming in general and expression templates in particular in demanding scientific computing codes. Also note that there was no particular optimization based on some knowledge of the underlying equation terms with respect to the local matrix assembly, so there is room for improvements — for example the symmetry of the bilinear form or the precomputation of stiffness, mass and convection matrices on the reference element, see [34].

Regarding compilation time, it is certainly true that using expressions template and meta-programming has an impact on the compile time. However if most of the library is templatized then the compile time cost is usually paid when compiling the end-user application and no more when compiling the library itself. To give an idea, the performance benchmark code was compiled with all cases : $\mathbb{P}1$ and $\mathbb{P}2$ in 2D and 3D for all problems D, DR and DAR. In other words, a single executable was generated to run all the cases presented earlier. On an AMD64



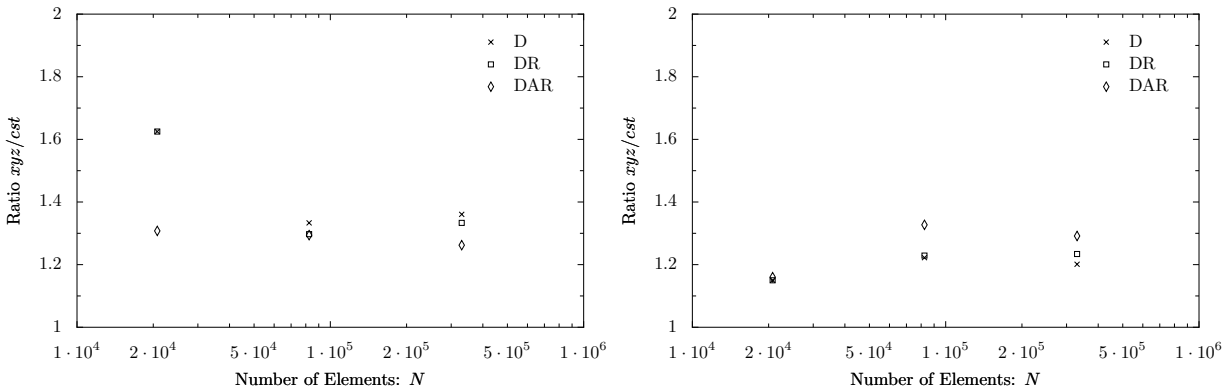
(a) $\mathbb{P}_1$: `cst`

(b) $\mathbb{P}_1$: `xyz`



(c) $\mathbb{P}_2$: `cst`

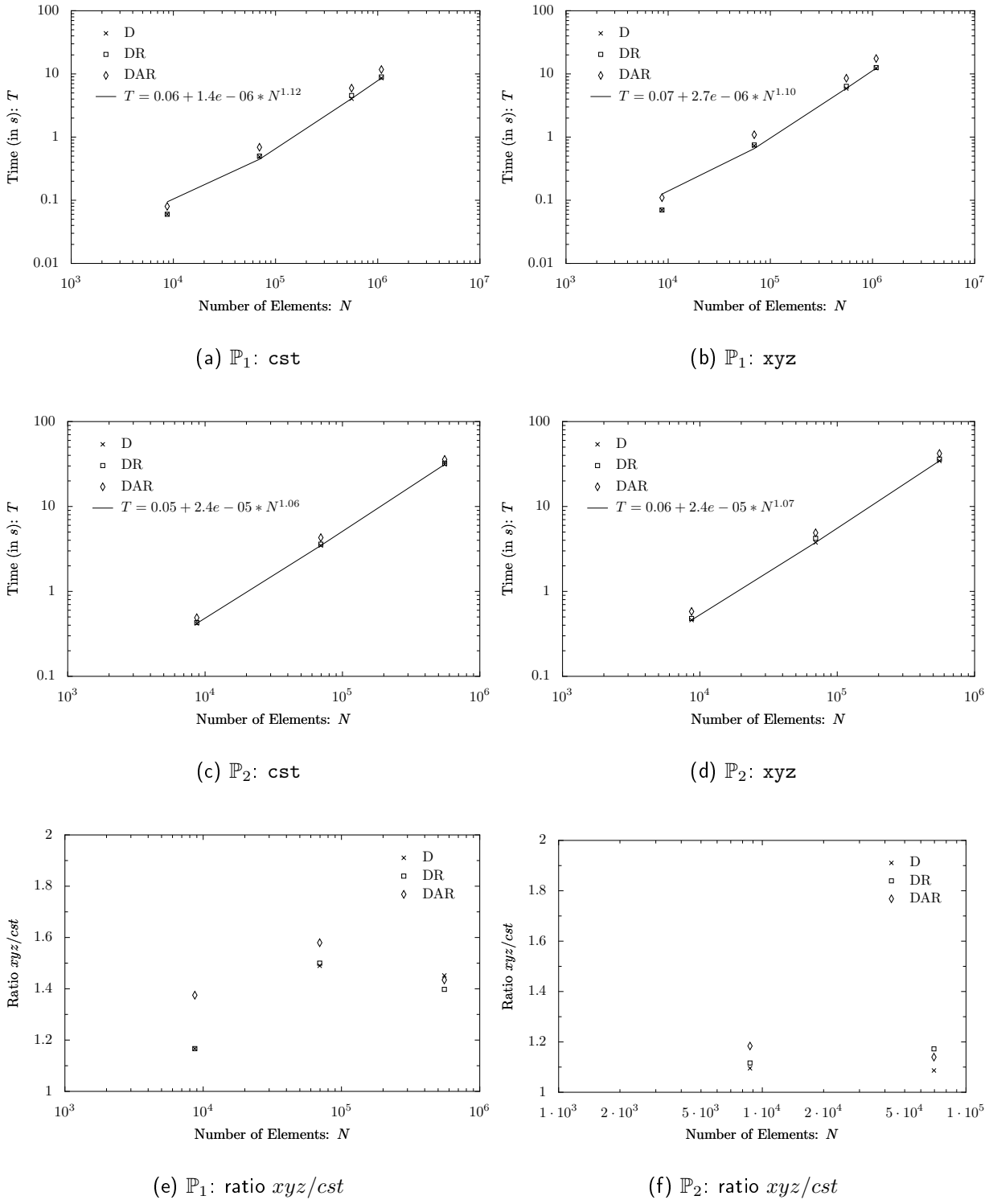
(d) $\mathbb{P}_2$: `xyz`



(e) $\mathbb{P}_1$: ratio xyz/cst

(f) $\mathbb{P}_2$: ratio xyz/cst

Figure 10.3: Cases $\mathbb{P}_1$ and $\mathbb{P}_2$ in 2D. Matrix assembly time versus number of elements and ratios between non-constant coefficients assembly and constant coefficients assembly



(a) $\mathbb{P}_1$: `cst` (b) $\mathbb{P}_1$: `xyz`

(c) $\mathbb{P}_2$: `cst` (d) $\mathbb{P}_2$: `xyz`

(e) $\mathbb{P}_1$: ratio xyz/cst (f) $\mathbb{P}_2$: ratio xyz/cst

Figure 10.4: Cases $\mathbb{P}_1$ and $\mathbb{P}_2$ in 3D. Matrix assembly time versus number of elements and ratios between non-constant coefficients assembly and constant coefficients assembly

opteron [†], the compilation takes between 2 and 3 minutes using the compiler options from listing 10.20. The book by David Abrahams and Aleksey Gurtovoy [2] provides a complete discussion on meta-programming and its impact on compile time.

10.4.2 A Variationnal Inequality

This test case is described in [20]. We consider a rectangular tank of length $a = 0.1\text{m}$, and height $b = 0.05\text{m}$. A cylindrical tube crosses it with a diameter of 0.015m. The tank is filled with ice and there is a thin layer of solid/liquid polymer on top of it. For symmetry reason, we consider only half of the tank — $a = 0.05\text{m}$. — The problem is formulated as follows: The temperature θ is solution of the parabolic equation

$$\int_\Omega \beta(\theta)\,\frac{\partial \theta}{\partial t}\,v \;-\; \nabla\cdot(\mu(\theta)\nabla\theta) \;=\; 0 \tag{10.21}$$

where

$$\mu(\theta) \;=\; \chi_{y<b-3p}[\mu_1\chi_{\theta>\epsilon} + \mu_2\chi_{\theta<0}] + \mu_3\chi_{y>b-3p}, \tag{10.22}$$

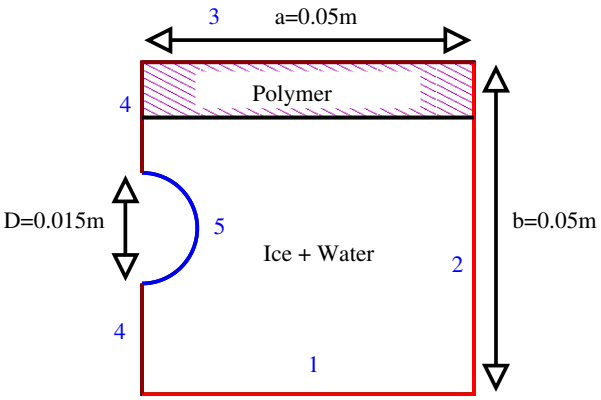


Figure 10.5: Variational Inequality

$$\beta(\theta) \;=\; \chi_{y<b-3p}[c_1\chi_{\theta>\epsilon} + c_2\chi_{\theta<0} + \frac{1}{\epsilon}L_1\chi_{0<\theta<\epsilon}] + \chi_{y>b-3p}[c_3\chi_{\theta>\epsilon} + c_4(\chi_{\theta>\epsilon} + \chi_{\theta<0}) + \frac{1}{\epsilon}L_2\chi_{0<\theta<\epsilon}] \tag{10.23}$$

and $\epsilon = 0.05$. $\beta(\theta)$ is the volumetric heat capacity and $\mu(\theta)$ is the thermal conductivity. Finally, χ represent the characteristic function.

We impose the following conditions:

- $\theta = 0$ on boundaries 1 and 2
- $\frac{\partial\theta}{\partial n} = 0$ on boundary 3 and 4
- $\theta = -0.1 + 0.05t$ on boundary 5 where t represents the time

The actual code reads as follows :

Listing 10.22: Variational Inequality

```
Mesh mesh;
fespace_type P1;
P1::element_type u,v;
LinearForm<fespace_type> f( P1, F );

f = integrate( elements(mesh),
```

[†] AMD Opteron(tm) Processor 248, 2.2Ghz, 1MB cache, 8GB ram

```
// term: β(θ) θ v
          id(P1,θ)*id(v)*
// χ_{y<b-3p}[c1χ_{θ>ε} + c2χ_{θ<0} + (L1/ε)χ_{0<θ<ε}]
           (chi(Py()<b-3*p)*
          (c1*chi(id(P1,θ)>epsilon)+
           c2*chi(id(P1,θ)<0)+
           (L1/epsilon)*(chi(id(P1,θ)>0)*
                         chi(id(P1,θ)<epsilon
// χ_{y>b-3p}[c3χ_{θ>ε} + c4(χ_{θ>ε} + χ_{θ<0}) + (L2/ε)χ_{0<θ<ε}]
           chi(Py()>=b-3*p)*(c3*chi(id(P1,θ)>
           c4*chi(id(P1,θ)<0.5)+
          (L2/epsilon)*chi(id(P1,θ)>0.5)*
                       chi(id(P1,θ)<0.5+epsil

BilinearForm<fespace_type> a( u, v, A );
a = integrate( elements(mesh),
// term: β(θ) θ v
          id(θ)*id(v)*
// χ_{y<b-3p}[c1χ_{θ>ε} + c2χ_{θ<0} + (L1/ε)χ_{0<θ<ε}]
           (chi(Py()<b-3*p)*
          (c1*chi(id(P1,θ)>epsilon)+
           c2*chi(id(P1,θ)<0)+
           (L1/epsilon)*(chi(id(P1,θ)>0)*
                         chi(id(P1,θ)<epsilon
// χ_{y>b-3p}[c3χ_{θ>ε} + c4(χ_{θ>ε} + χ_{θ<0}) + (L2/ε)χ_{0<θ<ε}]
           chi(Py()>=b-3*p)*(c3*chi(id(P1,θ)>
           c4*chi(id(P1,θ)<0.5)+
          (L2/epsilon)*chi(id(P1,θ)>0.5)*
                       chi(id(P1,θ)<0.5+epsil

// term: dt μ(θ) ∇ θ · ∇ v
          dt*dot(grad(θ),grad(v))*(
// χ_{y<b-3p}[μ1χ_{θ>ε} + μ2χ_{θ<0}] + μ3χ_{y>b-3p}
           (chi(Py()<b-3*p)*(\mu_1*chi(id(P1,
           mu2*chi(id(P1,θ)<0))+
           mu3*chi(Py()>b-3*p)));
// boundary conditions
a += on(1,θ,F,-0.1)+
     on(2,θ,F,-0.1)+
     on(5,θ,F,-0.1+0.05*t);
```

Figure 10.6 shows the contour lines of the temperature at various time steps and the mesh colored by the thermal conductivity over the entire domain: we see that the ice is melting and transformed into water — in red the ice and light blue the water.

10.4.3 Stokes and Navier-Stokes

We consider here the standard driven cavity test case with the following setting described by the figure 10.7.
First we use a stable mixed approximation for the velocity and pressure spaces — say the Taylor-Hood element ($\mathbb{P}_2 - \mathbb{P}_1$). — The variational formulation reads as follows:
Find $(u, p) \in X_h \times M_h$ such that for all $(v, q) \in X_h \times M_h$

$$\int_\Omega \nabla u \cdot \nabla v - \int_\Omega \nabla \cdot v \; p = 0 \tag{10.24}$$

$$\int_\Omega \nabla \cdot u \; q = 0 \tag{10.25}$$

$$u_{|Y=1} = (1,0)^T \quad \text{in 2D} \tag{10.26}$$

$$u_{|Z=1} = (1,0,0)^T \quad \text{in 3D} \tag{10.27}$$

Listing 10.23: Mixed Variational Formulation

```
// mixed finite element space (P2 velocity, P1 pressure) in 3D
typedef MixedSpace<Mesh_t,
                   FEM_PK<3,2,vectorial>,
                   FEM_PK<3,1,scalar> > space_type;
space_type V_h;
V_h U,V;
// views for U and V
typedef space_type::space_1_type X_h;
typedef space_type::element_2_type M_h;
space_type::element_1_type u = U.element1();
space_type::element_2_type p = U.element2();
space_type::element_1_type v = V.element1();
space_type::element_2_type q = V.element2();
//
csr_matrix_type A;
MixedBilinearForm<space_type> a( V_h, V_h, A );
// block wise construction
a( u, v ) = integrate( elements(mesh), dot(grad(u), grad(v)) );
a( p, v ) = integrate( elements(mesh), -id(p)*div(v) );
a( u, q ) = integrate( elements(mesh), id(q)*div(u) );
a( p, q ) = integrate( elements(mesh), 1e-6*id(p)*id(q) )+
            // 10 identifies the lid
            on( 10, u, F, oneX() )+
            // 20 = ∂Ω\10
            on( 20, u, F, 0 );
// or could be done this way. less expensive:
// only one loop over the elements
a = integrate( elements(mesh),
               dot(gradt(u), grad(v)) - idt(p)*div(v) +
               id(q)*divt(u) + 1e-6*idt(p)*id(q) )+
```

```
    // 10 identifies the lid
    on( 10, u, F, oneX() )+
    // 20 = ∂Ω\10
    on( 20, u, F, 0 );
```

We can also use an equal order approximation — say $\mathbb{P}1 - \mathbb{P}1$ — and add a stabilization term like the jump of the pressure gradient over the internal faces as proposed in [18]. The formulation reads then as follows: Find $(u, p) \in X_h \times M_h$ such that $\forall\, (v, q) \in X_h \times M_h$

$$\int_\Omega \nabla u \cdot \nabla v - \int_\Omega \nabla \cdot v \; p = 0 \tag{10.28}$$

$$\int_\Omega \nabla \cdot uq + \sum_{\mathcal{F} \in \Omega_h} \int_{\mathcal{F}} [\frac{\partial p}{\partial n}][\frac{\partial q}{\partial n}] = 0 \tag{10.29}$$

$$u_{|Y=1} = (1,0)^T \quad \text{in 2D} \tag{10.30}$$

$$u_{|Z=1} = (1,0,0)^T \quad \text{in 3D} \tag{10.31}$$

where $[\cdot]$ denotes the jump of the quantity across a face. Codewise we operate just a slight modification of listing 10.23, see the listing 10.24.

Listing 10.24: Mixed Variational Formulation

```
// can be of equal order for velocity and pressure spaces
MixedBilinearForm<space_type> a( V_h, V_h, A, false );
a = integrate( elements(mesh),
               dot(gradt(u), grad(v)) - idt(p)*div(v) +
               id(q)*divt(u) + 1e-6*idt(p)*id(q) )+
  // add the jump of the normal derivatives of the pressure
  // where Γ_p is a constant. One can take Γ_p = 2.5e^-2
  integrate(internalfaces(mesh),Γ_p*(Hface()^3)*jump(dnt(p))*
                                jump(dn(q)) )+
  // 10 identifies the lid
  on( 10, u, F, oneX() )+
  // 20 = ∂Ω\10
  on( 20, u, F, 0 );
```

10.4.4 Particule in a Shear Flow

We consider now a rigid particule in a shear Stokes flow treated using a penalization method, see [31].

Denote $\Omega = [-1, 1]^2$, we consider a cirular particule positionned at $(x_p = 0, y_p = 0) \in \Omega$ of radius $r_p = 0.1$ and a penalization parameter ϵ. Denote χ_p the characteristic function of the particule defined as $\chi_p = \chi((x - xp)^2 + (y - yp)^2 > r_p)$. We seek $(u, p) \in X_h \times M_h$ such

that $\forall (v,q) \in X_h \times M_h$

$$\int_\Omega (1+\chi_p/\epsilon)(\nabla u + \nabla u^T)(\nabla v + \nabla v^T) - \int_\Omega \nabla \cdot v\, p = 0 \tag{10.32}$$

$$\int_\Omega \nabla \cdot u\, q = 0 \tag{10.33}$$

$$u_{|\partial\Omega} = (0, x) \tag{10.34}$$

where X and M are sub-spaces of $H_1(\Omega)$ and $L_2(\Omega)$ respectively. Listing 10.25 shows the corresponding *C++* code.

Listing 10.25: Particular in a shear Stokes flow

```
// (xp,yp) are the coordinates of the center of the particule
// rp is the radius of the particule
//  ∫_[-1,1]^2 (1+χ_{(x-xp)^2+(y-yp)^2>r}/ε)(∇u+∇u^T)(∇v+∇v^T)
MixedBilinearForm<Space_t> a(P2P1,P2P1,A);
a = integrate( elements(mesh),
               (1+chi(rp^2>(Px()-xp)^2+(Py()-yp)^2)/eps)*
               dot(gradt(u)+gradTt(v),grad(v)+gradT(v))-
               idt(p)*div(v) + idt(q)*divt(u),
               IM_PK<2,2>() )+
// Dirichlet boundary conditions for the shear flow
    on( 7, uy, F, Px() )+
    on( 7, ux, F, 0 );
```

Figure 10.8 shows the velocity vector field and identifies the particule in the mesh using its characteristic function χ_p.

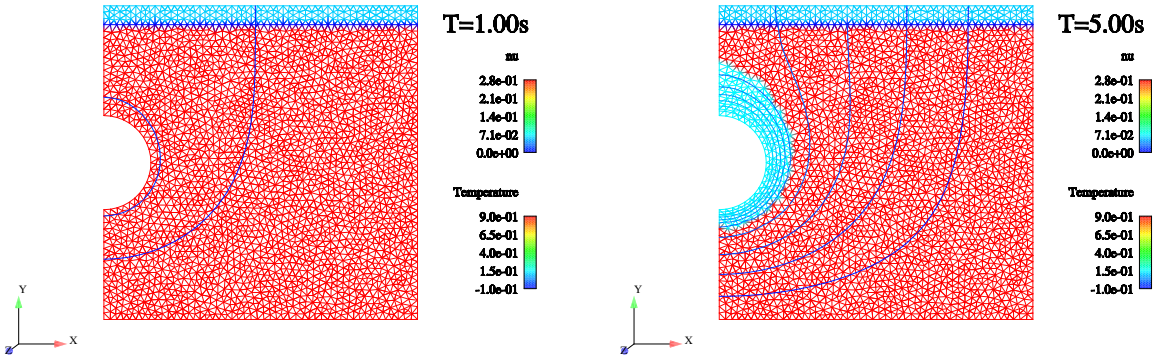
Conclusion

We have developed a unified DSEL for different aspects of numerical analysis, namely differentiation, integration, projection and variational formulations. The main results of this article are that *(i)* such a DSEL is feasible: an implementation has been done and exercised with some non-trivial test cases and *(ii)* decoupling the expression object construction from its evaluation using a delegate subclass allows for very powerful notations in *C++*: one unique engine is used for the expression construction and as many engines as needed for the expression evaluation — we have seen two different engines: one for projection, integration and variational formulations and one for automatic differentiation. — This technique can certainly be successfully used in other contexts.

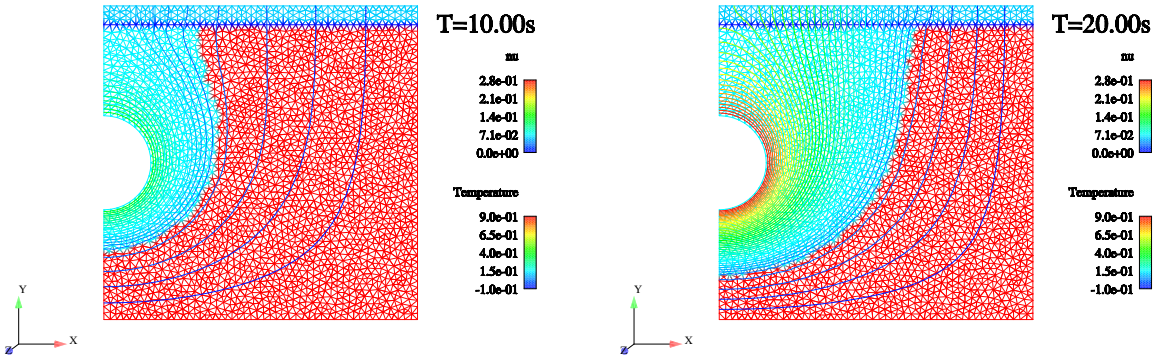
Future developments will include a study whether the work done in [34] can be applied to accelerate the assembly steps during integration. Also, although vectorial notations in the language can be used, they need to be formalized within the language to avoid ambiguities that would yield wrong results.

Acknowledgments

The DSEL was developed within a development branch of the LifeV project, see [65, 1].

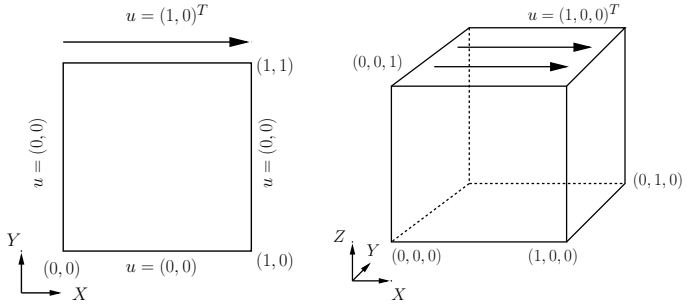


(a) $T = 1s$ (b) $T = 5s$



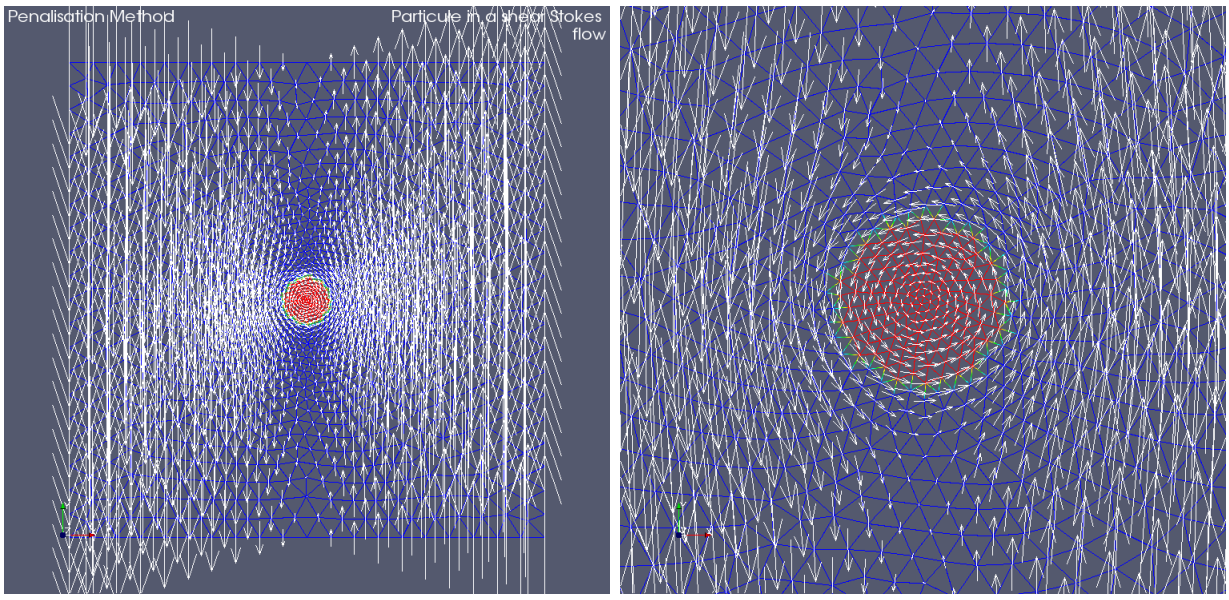
(c) $T = 10s$ (d) $T = 20s$

Figure 10.6: Temperature and thermal conductivity in the tank at various time steps



(a) 2D (b) 3D

Figure 10.7: Driven Cavity



(a) Velocity vector field (b) Close-up

Figure 10.8: Particule in shear Stokes flow

Part IV

Annex

Chapter 1

References

Bibliography

[1] Lifev: a finite element library. http://www.lifev.org.

[2] David Abrahams and Aleksey Gurtovoy. *C++ Template Metaprogramming : Concepts, Tools, and Techniques from Boost and Beyond*. C++ in Depth Series. Addison-Wesley Professional, 2004.

[3] M. A. Akgun, J. H. Garcelon, and R. T. Haftka. Fast exact linear and non-linear structural reanalysis and the Sherman-Morrison-Woodbury formulas. *International Journal for Numerical Methods in Engineering*, 50(7):1587–1606, March 2001.

[4] E. Allgower and K. Georg. Simplicial and continuation methods for approximating fixed-points and solutions to systems of equations. *SIAM Review*, 22(1):28–85, 1980.

[5] B. O. Almroth, P. Stern, and F. A. Brogan. Automatic choice of global shape functions in structural analysis. *AIAA Journal*, 16:525–528, May 1978.

[6] A.C. Antoulas and D.C. Sorensen. Approximation of large-scale dynamical systems: An overview. Technical report, Rice University, 2001.

[7] S. Sugata A.T. Patera. Reduced basis approximation and a posteriori error estimation for many-parameter problems. *MIT Disertation*, 2004.

[8] Pierre Aubert and Nicolas Di Césaré. Expression templates and forward mode automatic differentiation. In George Corliss, Christèle Faure, Andreas Griewank, Laurent Hascoët, and Uwe Naumann, editors, *Automatic Differentiation of Algorithms: From Simulation to Optimization*, Computer and Information Science, chapter 37, pages 311–315. Springer, New York, NY, 2001.

[9] Pierre Aubert, Nicolas Di Césaré, and Olivier Pironneau. Automatic differentiation in C++ using expression templates and application to a flow control problem. *Computing and Visualisation in Sciences*, 2000. Accepted.

[10] Babak Bagheri and Ridgway Scott. Analysa. http://people.cs.uchicago.edu/ ridg/al/aa.ps, 2003.

[11] David H. Bailey, Yozo Hida, Karthik Jeyabalan, Xiaoye S. Li, and Brandon Thompson. C++/fortran-90 arbitrary precision package. http://crd.lbl.gov/~dhbailey/mpdist/.

[12] Satish Balay, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2004.

[13] E. Balmes. Parametric families of reduced finite element models: Theory and applications. *Mechanical Systems and Signal Processing*, 10(4):381–394, 1996.

[14] A. Barrett and G. Reddien. On the reduced basis method. *Z. Angew. Math. Mech.*, 75(7):543–549, 1995.

[15] F. Brezzi, J. Rappaz, and P.A. Raviart. Finite dimensional approximation of nonlinear problems. Part I: Branches of nonsingular solutions. *Numerische Mathematik*, 36:1–25, 1980.

[16] A. Buffa, Y.Maday, A.T. Patera, C. Prud'homme, and G. Turinici. A priori convergence of multi-dimensional parametrized reduced basis, 2005. In preparation, To be submitted in October or November.

[17] E. Burman. A unified analysis for conforming and nonconforming stabilized finite element methods using interior penalty. *SIAM J. Numer. Anal.*, 2005. in press.

[18] Erik Burman and Peter Hansbo. Edge stabilization for the generalized Stokes problem: a continuous interior penalty method. *Comput. Methods Appl. Mech. Engrg.*, 2005. in press.

[19] G. Caloz and J. Rappaz. Numerical analysis for nonlinear and bifurcation problems. In P.G. Ciarlet and J.L. Lions, editors, *Handbook of Numerical Analysis, Vol. V*, Techniques of Scientific Computing (Part 2), pages 487–637. Elsevier Science B.V., 1997.

[20] Nicolas Di Césaré and Oliver Pironneau. Hatfem, une bibliothèque de manipulation des fonctions chapeaux. http://nicolas.dicesare.free.fr/Fac/R97033.ps.gz and http://www.ann.jussieu.fr/~pironneau/.

[21] T. F. Chan and W. L. Wan. Analysis of projection methods for solving linear systems with multiple right-hand sides. *SIAM Journal on Scientific Computing*, 18(6):1698, 1721 1997.

[22] Philippe G. Ciarlet. *The finite element method for elliptic problems*, volume 40 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2002. Reprint of the 1978 original [North-Holland, Amsterdam; MR0520174 (58 #25001)].

[23] Patrick Dular and Christophe Geuzaine. Getdp: a general environment for the treatment of discrete problems. http://www.geuz.org/getdp.

[24] A. G. Evans, J. W. Hutchinson, N.A. Fleck, M. F. Ashby, and H. N. G. Wadley. The topological design of multifunctional cellular metals. *Progress in Materials Science*, 46(3-4):309–327, 2001.

[25] C. Farhat, L. Crivelli, and F.X. Roux. Extending substructure based iterative solvers to multiple load and repeated analyses. *Computer Methods in Applied Mechanics and Engineering*, 117(1-2):195–209, July 1994.

[26] J. P. Fink and W. C. Rheinboldt. On the error behavior of the reduced basis technique for nonlinear finite element approximations. *Z. Angew. Math. Mech.*, 63:21–28, 1983.

[27] L. Formaggia, J.F. Gerbeau, and C. Prud'homme. *LifeV Developer Manual*. The LifeV Project. http://www.lifev.org/documentation/lifev-dev.pdf.

[28] M. B. Giles and N. A. Pierce. Superconvergent lift estimates through adjoint error analysis. Technical report, Oxford University Computing Laboratory, 1998.

[29] Frédéric Hecht and Olivier Pironneau. *FreeFEM++ Manual*. Laboratoire Jacques Louis Lions, 2005.

[30] Yozo Hida, Xiaoye S. Li, and David H. Bailey. Quad-double arithmetic: Algorithms, implementation, and application. Technical Report LBNL-46996, Lawrence Berkeley National Laboratory, Berkeley, CA 9472, Oct. 2000. http://crd.lbl.gov/~dhbailey/mpdist/.

[31] Joao Janela, Aline Lefebvre, and Bertrand Maury. A penalty method for the simulation of fluid - rigid body interaction. In *ESAIM proceedings*, 2005. submitted.

[32] George Em Karniadakis and Spencer J. Sherwin. *Spectral/hp element methods for CFD*. Numerical Mathematics and Scientific Computation. Oxford University Press, New York, 1999.

[33] Vesa Karvonen and Paul Mensonides. The boost library preprocessor subset for c/c++. http://www.boost.org/libs/preprocessor/doc/.

[34] Robert C. Kirby and Anders Logg. A compiler for variational forms. Technical report, Chalmers Finite Element Center, 05 2005. www.phi.chalmers.se/preprints.

[35] A. Logg, J. Hoffman, R.C. Kirby, and J. Jansson. Fenics. http://www.fenics.org/, 2005.

[36] Kevin Long. Sundance: Rapid development of high-performance parallel finite-element solutions of partial differential equations. http://software.sandia.gov/sundance/.

[37] L. Machiels, Y. Maday, I. B. Oliveira, A. T. Patera, and D. V. Rovas. Output bounds for reduced-basis approximations of symmetric positive definite eigenvalue problems. *C. R. Acad. Sci. Paris, Série I*, 331(2):153–158, July 2000.

[38] L. Machiels, Y. Maday, and A. T. Patera. A "flux-free" nodal Neumann subproblem approach to output bounds for partial differential equations. *C. R. Acad. Sci. Paris, Série I*, 330(3):249–254, February 2000.

[39] L. Machiels, A. T. Patera, and D. V. Rovas. Reduced basis output bound methods for parabolic problems. *Computer Methods in Applied Mechanics and Engineering*, 2001. Submitted.

[40] L. Machiels, J. Peraire, and A. T. Patera. *A posteriori* finite element output bounds for the incompressible Navier-Stokes equations: Application to a natural convection problem. *Journal of Computational Physics*, 172:401–425, 2001.

[41] Y. Maday. Private communication.

[42] Y. Maday, L. Machiels, A. T. Patera, and D. V. Rovas. Blackbox reduced-basis output bound methods for shape optimization. In *Proceedings* 12^{th} *International Domain Decomposition Conference*, pages 429–436, Chiba, Japan, 2000.

[43] Y. Maday, A. T. Patera, and J. Peraire. A general formulation for a posteriori bounds for output functionals of partial differential equations; Application to the eigenvalue problem. *C. R. Acad. Sci. Paris, Série I*, 328:823–828, 1999.

[44] Y. Maday, A. T. Patera, and D. V. Rovas. A blackbox reduced-basis output bound method for noncoercive linear problems. In D. Cioranescu and J.-L. Lions, editors, *Nonlinear Partial Differential Equations and Their Applications, Collége de France Seminar Volume XIV*, pages 533–569. Elsevier Science B.V., 2002.

[45] Y. Maday, A. T. Patera, and D.V. Rovas. Petrov-Galerkin reduced-basis approximations to noncoercive linear partial differential equations. In progress.

[46] Y. Maday, A. T. Patera, and G. Turinici. Global *a priori* convergence theory for reduced-basis approximation of single-parameter symmetric coercive elliptic partial differential equations. *C. R. Acad. Sci. Paris, Série I*, 335(3):289–294, 2002.

[47] Y. Maday, A. T. Patera, and G. Turinici. A priori convergence theory for reduced–basis approximations of single-parameter elliptic partial differential equations. *Journal of Scientific Computing*, 17(1–4):437–446, December 2002.

[48] Y. Maday, A.T. Patera, and G. Turinici. A priori convergence theory for reduced-basis approximations of single-parametric elliptic partial differential equations. *Journal of Scientific Computing*, 17(1-4):437–446, 2002.

[49] Yvon Maday, T. Patera, Anthony, and Gabriel Turinici. Global a priori convergence theory for reduced-basis approximations of single-parameter symmetric coercive elliptic partial differential equations. *C. R. Acad. Sci., Paris, Sér. I, Math.*, 335:289–294, 2002.

[50] N. C. Nguyen. *Reduced-Basis Approximation and* A Posteriori *Error Bounds for Nonaffine and Nonlinear Partial Differential Equations: Application to Inverse Analysis*. PhD thesis, Singapore-MIT Alliance, National University of Singapore., 2005. In progress.

[51] A. K. Noor and J. M. Peters. Reduced basis technique for nonlinear analysis of structures. *AIAA Journal*, 18(4):455–462, April 1980.

[52] I. B. Oliveira and A. T. Patera. Reliable real-time optimization of nonconvex systems described by parametrized partial differential equations. In *Proceedings Singapore-MIT Alliance Symposium*, January 2003.

[53] A. T. Patera and E. M. Rønquist. A general output bound result: Application to discretization and iteration error estimation and control. *Mathematical Models and Methods in Applied Science*, 2000. MIT FML Report 98-12-1.

[54] A. T. Patera and E. M. Rønquist. A general output bound result: Application to discretization and iteration error estimation and control. *Math. Models Methods Appl. Sci.*, 11(4):685–712, 2001.

[55] J. S. Peterson. The reduced basis method for incompressible viscous flow calculations. *SIAM J. Sci. Stat. Comput.*, 10(4):777–786, July 1989.

[56] Daniele A. Di Pietro and Alessandro Veneziani. Expression templates implementation of continuous and discontinuous galerkin methods. *Submitted to Computing and Visualization in Science*, 2005.

[57] Stéphane Del Pino and Olivier Pironneau. *FreeFEM3D Manual*. Laboratoire Jacques Louis Lions, 2005.

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

[59] C. Prud'homme. Adaptative reduced basis space generation and approximation. *In preparation*, 2005.

[60] C. Prud'homme. Automatic differentiation for real-time optimization in the context of the reduced-basis output bound methods for parametrized partial differential equations. *In preparation*, 2005.

[61] C. Prud'homme and A. T. Patera. Reduced-basis output bounds for approximately parameterized elliptic coercive partial differential equations. *Comput. Vis. Sci.*, 6(2-3):147–162, 2004.

[62] C. Prud'homme, D. V. Rovas, K. Veroy, L. Machiels, Y. Maday, A. T. Patera, and G. Turinici. Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods. *Journal of Fluids Engineering*, 124(1):70–80, 2002.

[63] C. Prud'homme, D.V. Rovas, K. Veroy, L. Machiels, Y. Maday, A.T. Patera, and G. Turinici. Reduced–basis output bound methods for parametrized partial differential equations. In *Proceedings SMA Symposium*, January 2002.

[64] C. Prud'homme, D.V. Rovas, K. Veroy, L. Machiels, Y. Maday, A.T. Patera, and G. Turinici. Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods. *Journal of Fluids Engineering - Transactions of the ASME*, 124(1):70–80, March 2002.

[65] Christophe Prud'homme. A generic library for variational methods. In preparation.

[66] Christophe Prud'homme. A domain specific embedded language in c++ for automatic differentiation, projection, integration and variational formulations. *Scientific Programming*, 2005. Submitted.

[67] Christophe Prud'homme, Dimitrios V. Rovas, Karen Veroy, and Anthony T. Patera. A mathematical and computational framework for reliable real-time solution of parametrized partial differential equations. *M2AN (Math. Model. Numer. Anal.)*, 36(5):747–771, 2002.

[68] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations*. Springer, 2nd edition, 1997.

[69] Yves Renard and Julien Pommier. Getfem++: Generic and efficient c++ library for finite element methods elementary computations. http://www-gmm.insa-toulouse.fr/getfem/.

[70] W. C. Rheinboldt. On the theory and error estimation of the reduced basis method for multi-parameter problems. *Nonlinear Analysis, Theory, Methods and Applications*, 21(11):849–858, 1993.

[71] W.C. Rheinboldt. Numerical analysis of continuation methods for nonlinear structural problems. *Computers and Structures*, 13(1-3):103–113, 1981.

[72] D.V. Rovas. *Reduced-Basis Output Bound Methods for Parametrized Partial Differential Equations*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, October 2002.

[73] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A*, 4(3):519–524, March 1987.

[74] Y. Solodukhov. *Reduced-Basis Methods Applied to Locally Non-Affine Problems*. PhD thesis, Massachusetts Institute of Technology, 2004. In progress.

[75] G. Strang and G.J. Fix. *An Analysis of the Finite Element Method*. Prentice-Hall, 1973.

[76] Todd Veldhuizen. Using C++ template metaprograms. *C++ Report*, 7(4):36–43, May 1995. Reprinted in C++ Gems, ed. Stanley Lippman.

[77] Todd L. VELDHUIZEN. Expression templates. *C++ Report*, 7(5):26–31, June 1995. Reprinted in C++ Gems, ed. Stanley Lippman.

[78] K. Veroy. *Reduced-Basis Methods Applied to Problems in Elasticity: Analysis and Applications*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2003. June.

[79] K. Veroy, T. Leurent, C. Prud'homme, D. Rovas, and A. T. Patera. Reliable real–time solution of parametrized elliptic partial differential equations: Application to elasticity. In *Proceedings Singapore-MIT Alliance Symposium*, January 2002.

[80] K. Veroy, C. Prud'homme, and A. T. Patera. Reduced-basis approximation of the viscous Burgers equation: Rigorous *a posteriori* error bounds. *C. R. Acad. Sci. Paris, Série I*, 337(9):619–624, November 2003.

[81] K. Veroy, C. Prud'homme, D. V. Rovas, and A. T. Patera. *A posteriori* error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations (AIAA Paper 2003-3847). In *Proceedings of the 16th AIAA Computational Fluid Dynamics Conference*, June 2003.

[82] K. Veroy, D. Rovas, and A. T. Patera. *A Posteriori* error estimation for reduced-basis approximation of parametrized elliptic coercive partial differential equations: "Convex inverse" bound conditioners. *Control, Optimisation and Calculus of Variations*, 8:1007–1028, June 2002. Special Volume: A tribute to J.-L. Lions.

[83] W.L. Wendland. *Elliptic Systems in the Plane*. Pitman, 1979.

[84] N. Wicks and J. W. Hutchinson. Optimal truss plates. *International Journal of Solids and Structures*, 38(30-31):5165–5183, July-August 2001.

[85] E. L. Yip. A note on the stability of solving a rank-p modification of a linear system by the Sherman-Morrison-Woodbury formula. *SIAM Journal on Scientific and Statistical Computating*, 7(2):507–513, April 1986.

Chapter 2

Curriculum Vitae

CURRICULUM VITAE

Christophe PRUD'HOMME

Addresse	Chemin du Chasseron 5 CH-1004 Lausanne Suisse
Téléphone	+41 21 647 0645
Mobile	+41 76 525 6414
Email	prudhomm@gmail.com
Nationalité	Française
Date/Lieu de Naissance	20 Août 1972, Versailles (78), France
Statut Marital	Célibataire

Expériences Professionnelles

09/2006 –	Professeur des Universités en mathématiques appliquées à l'université Joseph Fourier Grenoble I dans le laboratoire de Modélisation et Calcul Scientifique
09/2004 – 08/2006	Membre des comités de pilotage et technique pour le calcul scientifique hautes performances à l'EPFL — http://hpc.epfl.ch
10/2003 – 08/2006	Chercheur dans la chaire de modélisation et calcul scientifique à l'EPFL (Suisse) sous la direction d'A. Quarteroni
11/2001 – 09/2003	Chercheur dans le Département d'Ingénierie Mécanique au MIT (USA) dans le groupe de A.T. Patera
02/2003 – 08/2003	Chercheur invité au Laboratoire Jacques Louis Lions, Univ. Paris VI (France) dans le groupe de recherche d'Y. Maday
12/1999 – 10/2001	Postdoctorant dans le Département d'Ingénierie Mécanique au MIT (USA) dans le groupe de A.T. Patera
1998 – 1999	Assistant de cours en calcul scientifique (Maîtrise et DESS)
1995/11 – 1996/10	Scientifique du contingent lors du service militaire au LIMSI, CNRS (France)
1995/04 – 1995/10	Stage de DEA à l'INRIA en France

Éducation

09/12/2005	Habilitation à diriger les recherches en mathématiques appliquées de l'Université Pierre et Marie Curie
11/1996 – 11/1999	Doctorant en mathématiques appliquées aux Laboratoires d´Analyse Numérique (Université Pierre et Marie Curie)
10/1990 – 10/1995	Scolarité à l'Université Pierre et Marie Curie Paris VI en mathématiques et mathématiques appliquées

Prix et Nominations

ANVAR	Gagnant de la 5ème édition du concours du ministère de la Recherche en 2003 pour aider la création d'entreprises développant des technologies innovantes

FIRTECH	3 ans de financement FIRTECH entre Octobre 1996 et Septembre 1999 — FIRTECH est un consortium d'entreprises françaises, de centres de recherche et d'universités

Développement de Logiciels

LIFE V	Librairie Élément Fini développée conjointement par l'EPFL, le Politecnico di Milano et l'INRIA/Rocquencourt.
GST	Un toolkit pour le calcul scientifique : Algèbre linéaire, Méthodes éléments Finis, élments joints et bases réduites avec bornes d'erreur, Environnement distribué pour simulations temps-réel
DEBIAN	Développeur pour le calcul scientifique et mainteneur de paquets dans Debian/Gnu/Linux BOOST, FREEFEM, FREEFEM3D, ARPACK, ARPACK++, GMSH, SUPERLU, UMFPACK, CORELINUX, QD, ARPREC
FREEFEM	Un langage et logiciel pour la méthode élément fini — Plus de 13000 télé-chargements depuis Juillet 2000

Affiliations

ACM	Membre depuis 2005
DEBIAN	Membre officiel et développeur de Debian/GNU/Linux depuis 2001
IEEE	Membre depuis 2005

Compétences

Design	**UML**	Très bonne connaissance de l'UML [+4years]
	Patterns	Très bonne connaissance des design patterns et leur application au calcul scientifique [+8years]
	Refactoring	Très bonne connaissance des techniques de refactoring [+4years]
	XP	Très bonne connaissance des techniques d'extreme programming [+3years]
Calcul	**Parallèle**	Très bonne connaissance des librairies de calcul parallèle — MPI, PVM — et des environnements — clusters Linux, calculateurs parallèles en général [+6years]
	Distribué	Très bonne connaissance des environnements distribués tels que CORBA(MICO,Qedo) et (XML-)RPC [+6years]
Langages Info.	**C/C++**	Connaissance parfaite [+12 years]
	Fortran	Tres bonne connaissance[+12years]
	Matlab	Très bonne connaissance [+6years]
	Octave	Très bonne connaissance [+6years]
	Maple	Très bonne connaissance [+6years]
	ML	Très bonne connaissance des langages à balise tels que HTML, XML et des outils de publications associés [+6years]
Langues Étrangères	**Anglais**	Lu, écrit, parlé couramment
	Allemand	Compétences de bases

LISTE DES PUBLICATIONS ET SÉMINAIRES

Christophe PRUD'HOMME

Publications : Méthodes des Bases Réduites avec Bornes d'Erreur

[1] C. Prud'homme and A. T. Patera. Reduced-basis output bounds for approximately parameterized elliptic coercive partial differential equations. *Comput. Vis. Sci.*, 6(2-3) :147–162, 2004.

[2] C. Prud'homme, D. Rovas, K. Veroy, Y. Maday, A. T. Patera, and G. Turinici. Reliable real-time solution of parametrized partial differential equations : Reduced-basis output bound methods. *Journal of Fluids Engineering*, 124(1) :70–80, March 2002.

[3] C. Prud'homme, D. V. Rovas, K. Veroy, L. Machiels, Y. Maday, A. T. Patera, and G. Turinici. Reduced–basis output bound methods for parametrized partial differential equations. In *Proceedings Singapore-MIT Alliance Symposium*, January 2002.

[4] Christophe Prud'homme, Dimitrios V. Rovas, Karen Veroy, and Anthony T. Patera. A mathematical and computational framework for reliable real-time solution of parametrized partial differential equations. *M2AN (Math. Model. Numer. Anal.)*, 36(5) :747–771, 2002.

[5] K. Veroy, T. Leurent, C. Prud'homme, D. Rovas, and A. T. Patera. Reliable real–time solution of parametrized elliptic partial differential equations : Application to elasticity. In *Proceedings Singapore-MIT Alliance Symposium*, January 2002.

[6] K. Veroy, C. Prud'homme, and A. T. Patera. Reduced-basis approximation of the viscous Burgers equation : Rigorous *a posteriori* error bounds. *C. R. Acad. Sci. Paris, Série I*, 337(9) :619–624, November 2003.

[7] K. Veroy, C. Prud'homme, D. V. Rovas, and A. T. Patera. *A posteriori* error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations (AIAA Paper 2003-3847). In *Proceedings of the 16th AIAA Computational Fluid Dynamics Conference*, June 2003.

Publications : Méthodes de Décomposition de Domaine

[1] Gassan Abdoulaev, Yves Achdou, Jean-Claude Hontand, Yuri Kuznetsov, Olivier Pironneau, and Christophe Prud'homme. Domain decomposition for Navier-Stokes equations. In *ICIAM 99 (Edinburgh)*, pages 191–204. Oxford Univ. Press, Oxford, 2000.

[2] Gassav S. Abdoulaev, Yves Achdou, Yuri A. Kuznetsov, and Christophe Prud'homme. On a parallel implementation of the mortar element method. *M2AN Math. Model. Numer. Anal.*, 33(2) :245–259, 1999.

[3] Yves Achdou, Gassan Abdoulaev, Jean-Claude Hontand, Yuri A. Kuznetsov, Olivier Pironneau, and Christophe Prud'homme. Nonmatching grids for fluids. In *Domain decomposition methods, 10 (Boulder, CO, 1997)*, volume 218 of *Contemp. Math.*, pages 3–22. Amer. Math. Soc., Providence, RI, 1998.

[4] C. Prud'homme. A strategy for the resolution of the tridimensionnal incompressible navier-stokes equations. In Hermes, editor, *Méthodes itératives de décomposition de domaines et communications en calcul parallèle*, volume 10 of *Calculateurs Parallèles Réseaux et Systèmes répartis*, pages 333–468. Hermes, 1998.

[5] C. Prud'homme. *Décomposition de domaines, application aux équations de Navier-Stokes tridimensionnelles incompressibles*. PhD thesis, University Pierre et Marie Curie, Paris VI, 2000.

Publications : Calcul Scientifique

[1] D. Bernardi, F. Hecht, O. Pironneau, and C. Prud'homme. *The GFEM Documentation Manual : A Finite Element Method Language*. Laboratoire d'analyse numérique de Paris VI, 1996.

[2] I. Mallabiabarrena, C. Prud'homme, A. Radaelli, V. Rigamonti, and D. Sage. From medical images to numerical blood flow simulations in human vessels. In *Proceedings of the 2005 Annual Meeting of the Swiss Society for Biomedical Engineering (SSBE'05)*, page F21, Lausanne VD, Switzerland, September 1-2, 2005.

[3] Christophe Prud'homme. A domain specific embedded language in c++ for automatic differentiation, projection, integration and variational formulations. *Scientific Programming*, 2005. Accepted.

Publications : Soumises

[1] C. Prud'homme, G. Fourestey, N. Parolini, A. Quarteroni, and G.Rozza. Matematica in volo. *Matematica e Cultura*, 2006. Submitted.

Publications : En Préparation

[1] A. Buffa, Y.Maday, A.T. Patera, C. Prud'homme, and G. Turinici. A priori convergence of multi-dimensional parametrized reduced basis, 2005. In preparation.

[2] C. Prud'homme. Adaptative reduced basis space generation and approximation. *In preparation*, 2005.

[3] C. Prud'homme. Automatic differentiation for real-time optimization in the context of the reduced-basis output bound methods for parametrized partial differential equations. *In preparation*, 2005.

[4] C. Prud'homme. A benchmark for reduced basis output bounds methods. *In preparation*, 2005.

[5] C. Prud'homme. A comparison between reduced basis output bounds methods and some response surface methods. *In preparation*, 2005.

[6] Christophe Prud'homme. A modern and unified c++ implementation of finite element and spectral element methods in 1, 2 and 3d. In preparation.

Rapporteur pour des Journaux

2003	Journal of Computing and Visualisation in Sciences – Springer
2004	Comptes Rendus de l'Académie des Sciences
2005	Journal of Scientific Computing – Springer

Séminaires Invités 2003-2006

[1] C. Prud'homme. Méthodes de bases réduites : Aspects mathématiques et informatiques. LMC/IMAG, Grenoble, July 2003.

[2] C. Prud'homme. Méthodes de bases réduites pour quelques problèmes en mécanique : Aspects mathématiques et informatiques. EPFL, Lausanne, Switzerland, June 2003.

[3] C. Prud'homme. Méthodes de bases réduites pour quelques problèmes non linéaires : Aspects mathématiques et informatiques. Laboratoire Jacques Louis Lions, Univ. Paris VI, June 2003.

[4] C. Prud'homme. Reduced basis output bounds methods : Theory. IMATI, CNR/Pavia, Italy, December 2003.

[5] C. Prud'homme. Reduced basis output bounds methods : Theory and applications. MOX, Politecnico di Milano, Italy, December 2003.

[6] C. Prud'homme. Simulations temps réel avec certificat de fiabilité : des systèmes embarqués à la grille de calcul. Laboratoire Jacques Louis Lions, Univ. Paris VI, July 23 2003.

[7] C. Prud'homme. Solar impulse, optimization framework and reduced basis. ModeFrontier User-Meeting '04, Trieste, Italy, September 2004.

[8] C. Prud'homme. Evolution of the lifev project in the context of haemodynamics. Keynote lecturer MODELLING OF PHYSIOLOGICAL FLOWS – MPF 2005 – http://www.math.ist.utl.pt/~mpf2005/, April 2005.

[9] C. Prud'homme. Une implémentation moderne et unifiée des méthodes d'éléments finis et des méthodes spectrales en dimension 1, 2 et 3. Seminaire du Laboratoire Jacques-Louis Lions Universite Pierre et Marie Curie (Paris VI), January 2006.

[10] C. Prud'homme. Une implémentation moderne et unifiée des méthodes d'éléments finis et des méthodes spectrales en dimension 1, 2 et 3. Seminaire du Laboratoire J.A. Dieudonné, Février 2006.

Seminaires et Conférences 2003-2006

[1] C. Prud'homme. A mathematical and computational framework for *Real-Time Reliable* simulations. A Day with Ivo Babuska, Laboratoire Jacques Louis Lions, Univ. Paris VI, February 2003.

[2] C. Prud'homme. Real-time reliable distributed simulations : Framework, repository and interfaces. Advanced Environments and Tools for High Performance Computing/ Abstracts, ESF, Albufeira (Algarve), Portugal, June 2003.

[3] C. Prud'homme. The life project. Talk given at the Heamodel European Project Meeting in Graz, April 2004.

[4] C. Prud'homme. A variational formulation language embedded in c++. Talk given at POOSC'05 a conference on parallel/high performance object oriented scientific computing in Scottland, July 2005.

LISTE DE RÉFÉRENCES

Christophe PRUD'HOMME

Références

Y. Maday — maday@ann.jussieu.fr — Université Pierre et Marie Curie, Paris France — Laboratoire Jacques-Louis Lions

A.T. Patera — patera@mit.edu — Massachusetts Institute of Technology, Cambridge USA — Mechanical Engineering Department

O. Pironneau — pironneau@ann.jussieu.fr — Université Pierre et Marie Curie, Paris France — Laboratoire Jacques-Louis Lions

A. Quarteroni — alfio.quarteroni@epfl.ch — École Polytechnique Fédérale de Lausanne — SB-IACS-CMCS