ESAIM: PROCEEDINGS, December 2013, Vol. 43, p. 225-254
S. Descombes, B. Dussoubs, S. Faure, L. Gouarin, V. Louvet, M. Massot, V. Miele, Editors

A REDUCED BASIS FRAMEWORK : APPLICATION TO LARGE SCALE NON-LINEAR MULTI-PHYSICS PROBLEMS

C. Daversin[1], S. Veys[2], C. Trophime[3] et C. Prud'homme[4]

Abstract. In this paper we present applications of the reduced basis method (RBM) to large-scale non-linear multi-physics problems. We first describe the mathematical framework in place and in particular the Empirical Interpolation Method (EIM) to recover an affine decomposition and then we propose an implementation using the open-source library Feel++ which provides both the reduced basis and finite element layers. Large scale numerical examples are shown and are connected to real industrial applications arising from the High Field Resistive Magnets development at the Laboratoire National des Champs Magnétiques Intenses.

Introduction

Nowadays, in many application fields, engineering problems require accurate, reliable, and efficient evaluation of quantities of interest. Often, these quantities of interest depend on the solution of a parametrized partial differential equation where the — *e.g.* physical or geometrical — parameters are inputs of the model and the evaluation of quantities of interest — *e.g.* average values — are outputs. In a real-time or many-query context, the reduced basis method (RBM) offers a rapid and reliable evaluation of the input-output relationship (see [Prud'homme et al., 2002, Veroy et al., 2003a, Veroy et al., 2003b, Prud'homme and Patera, 2004, Quarteroni et al., 2011, Rozza et al., 2007] for the methodology) for a large class of problems.

In this paper, we are interested in studying the RBM applied to large scale non-linear multi-physics parametrized partial differential equations requiring not only a robust mathematical framework but also a HPC-enabled computational framework. We propose an implementation of the reduced basis method and the extensions to non-linear and non-affinely parametrized problems. Other implementations are available, such as [Patera and Rozza, 2007, Knezevic and Peterson, 2010]. The reduced basis methodology is suited to develop efficient strategies to tackle design and optimization in industrial context however, to our knowledge, it has not (yet) been used effectively in this context. Typical activities of a design department require the ability to efficiently perform parametric studies and sensitivity analysis to improve and guide engineers in their daily work. In particular, safety is an essential ingredient in industrial investigations which involves to take into account all eventual uncertainties on input parameters. The reduced basis framework provides a valuable tool for design only if it may be applied seamlessly from small to large scale applications and may take care of non-linear quantities. We will show how Feel++ framework - which includes these features - may be used to help engineers in their design process. Numerical examples are given in the context of the High Fields Resistives magnets development

[1] Laboratoire National des Champs Magnétiques Intenses, 25 Av des Martyrs, 38042 Grenoble Cedex, France - cecile.daversin@lncmi.cnrs.fr
[2] Laboratoire Jean Kuntzmann, Université Joseph Fourier Grenoble 1, BP53 38041 Grenoble Cedex 9, France - stephane.veys@imag.fr
[3] Laboratoire National des Champs Magnétiques Intenses, 25 Av des Martyrs, 38042 Grenoble Cedex, France - christophe.trophime@lncmi.cnrs.fr
[4] Université de Strasbourg, IRMA UMR 7501, 7 rue René-Descartes, 67084 Strasbourg Cedex, France - prudhomme@unistra.fr

Article published online by EDP Sciences and available at http://www.esaim-proc.org or http://dx.doi.org/10.1051/proc/201343015

at the french Laboratoire National des Champs Magnétiques Intenses. To our knowledge these examples are among the first to show applications of the reduced basis methodology on industrial problems that lead to actual realizations.

In order to solve Finite Elements (FE) or Reduced Basis (RB) problems, we use an open-source library called Feel++ for *Finite Element Embedded Library and Language in C++* ([Prud'Homme et al., 2012a, Prud'homme, 2006]). Feel++ is a library to solve problems arising from partial differential equations (PDEs) with Galerkin methods, standard or generalized, continuous or discontinuous, from 1D to 3D, for low to high order approximations (including geometry). Among the many other Feel++ features, it provides a seamless programming environment with respect to parallel computing using MPI, see section 2.1. Feel++ enjoys an implementation of the RBM which can deal with a wide range of problems: elliptic or parabolic models, coercive or non-coercive models, linear or non-linear models. As mentioned earlier, it is important that such an environment hides as many implementation details as possible and let the user worry only about his/her model and the high level aspects of the FEM and RBM.

The organisation of the paper is as follows: in section 1 we describe the ingredients of the RB mathematical framework on which we build the RB computational framework described in section 2. Especially, we introduce the Empirical Interpolation Method which is actually an integral part of our RB framework. Its main feature is that it enables the RB method on model which do not enjoy the so-called affine decomposition, such as non-linear multi-physics applications of LNCMI described in section 3.

1. Mathematical framework for reduced basis

This section describes the mathematical framework of the reduced basis method (RBM) and ingredients we need. We first introduce the outline of RBM on elliptic linear problems with affine dependence in parameters. The non-linearity that comes accross the problems we focus on can lead to a non-affine dependence on input parameters. The Empirical Interpolation Method (EIM) is a good way to manage this by recovering affine dependence in parameters. Furthermore the rapid evaluation of the output offered by RBM, the framework gives a reliability guarantee for the results. This uses Successive Constraints Method (SCM) as an ingredient to compute efficiently the lower bound of the coercivity constant of a bilinear form, in order to have an a posteriori error estimation. Finally, the application of RBM on elliptic non-linear problems by means of the tools previously introduced is more precisely described.

1.1. Elliptic linear problems with affine dependence in parameters

1.1.1. *Preliminaries*

Let Ω be a suitably regular bounded spatial domain in $\mathbb{R}^d$ (for d=1,..,3). Denoting $L^2(\Omega)$ the space of square integrable functions over Ω, we have $H^1(\Omega) = \{u | u \in L^2(\Omega), \nabla u \in L^2(\Omega)^d\}$ and $H^1_0(\Omega) = \{u \in H^1(\Omega) | u_{\partial\Omega} = 0\}$. From an Hilbert space $X \equiv H^1_0(\Omega)^\nu$ - or more generally $H^1_0(\Omega)^\nu \subset X \subset H^1(\Omega)^\nu$, where $\nu = 1$ (respectively d) for a scalar (respectively vector) field, we define $X^{\mathcal{N}}$ a finite element approximation space of (typically very large) dimension $\mathcal{N}$.

1.1.2. *General problem settings*

Let $u(\boldsymbol{\mu})$ be the solution of a parametrized Partial Differential Equation (PDE) with respect to the input parameter p-vector $\boldsymbol{\mu} \in \mathcal{D}$, where $\mathcal{D} \subset \mathbb{R}^p$ is the parameter space. We are interested in the evaluation of an output of interest $s(\boldsymbol{\mu}) \in \mathbb{R}$ which can be expressed as a functional of a field variable $u(\boldsymbol{\mu})$:

$$s(\boldsymbol{\mu}) = \ell(u(\boldsymbol{\mu}); \boldsymbol{\mu}) \ , \tag{1}$$

for a suitable linear operator $\ell(.; \boldsymbol{\mu})$. The variational formulation of the PDE consists in finding $u(\boldsymbol{\mu}) \in X$ such that

$$a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f(v; \boldsymbol{\mu}) \quad \forall v \in X \ , \tag{2}$$

where $a(.,.;\boldsymbol{\mu})$ and $f(.,\boldsymbol{\mu})$ are respectively bilinear and linear forms associated with the PDE. An important ingredient of the RBM is the development of an efficient offline/online strategy. To this end, a and f and ℓ must depend affinely in $\boldsymbol{\mu}$ that is to say that there exists positive integers Q_a, Q_f and Q_ℓ such that $a(.,.;\boldsymbol{\mu})$, $f(.;\boldsymbol{\mu})$ and $\ell(.;\boldsymbol{\mu})$ can be expressed as

$$\begin{cases} a(u,v;\boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}) a^q(u,v) \quad \forall u,v \in X,\ \forall \boldsymbol{\mu} \in \mathcal{D}\ , \\ f(v;\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}) f^q(v) \quad \forall v \in X,\ \forall \boldsymbol{\mu} \in \mathcal{D}\ , \\ \ell(v;\boldsymbol{\mu}) = \sum_{q=1}^{Q_\ell} \theta_\ell^q(\boldsymbol{\mu}) \ell^q(v) \quad \forall v \in X,\ \forall \boldsymbol{\mu} \in \mathcal{D}\ , \end{cases} \tag{3}$$

where $\theta_a^q : \mathcal{D} \to \mathbb{R}$, $1 \le q \le Q_a$, $\theta_f^q : \mathcal{D} \to \mathbb{R}$, $1 \le q \le Q_f$ and $\theta_\ell^q : \mathcal{D} \to \mathbb{R}$, $1 \le q \le Q_\ell$ are $\boldsymbol{\mu}$-dependent functions. Sections 1.3 and 1.4 propose solutions when the affine decomposition are not readily available.

1.1.3. *Reduced basis method*

We now turn to the construction of the reduced basis approximation. For given a $\boldsymbol{\mu} \in \mathcal{D}$, we start with the finite element (FE) discretization of problem (1)-(2) which consists in evaluating

$$s_{\mathcal{N}}\left(u_{\mathcal{N}}(\boldsymbol{\mu})\right) = \ell\left(u_{\mathcal{N}}(\boldsymbol{\mu});\boldsymbol{\mu}\right)\ , \tag{4}$$

where $u_{\mathcal{N}}(\boldsymbol{\mu}) \in X^{\mathcal{N}}$ satisfies

$$a(u_{\mathcal{N}}(\boldsymbol{\mu}),v;\boldsymbol{\mu}) = f(v;\boldsymbol{\mu}) \quad \forall v \in X^{\mathcal{N}}. \tag{5}$$

For a given positive integer N_{max} we introduce a nested sequence of reduced basis approximation spaces $W_{N_{pr}}$, $1 \le N \le N_{max}$. Note that $W_{N_{pr}}$ is a N-dimensional subspace of $X^{\mathcal{N}}$, and that N_{max} is very small in comparison to $\mathcal{N}$.

Let $S_N = \{\boldsymbol{\mu}^1, ..., \boldsymbol{\mu}^N\}$ be a set of parameters log-randomly picked in $\mathcal{D}$. Usually the Greedy algorithm is used to perform this parameters selection but error estimation for non-linear problems is an ongoing work in the `Feel++` framework.

We then define the set of solutions S_N^u as

$$S_N^u = \{u_{\mathcal{N}}(\boldsymbol{\mu}^i)\ ,\ \forall \boldsymbol{\mu}^i \in S_N\}\ . \tag{6}$$

The application of the Gram-Schmidt process with respect to the $(.,.)_X$ inner product to elements of the set S_N^u gives mutually $(.,.)_X$-orthonormal basis functions ξ_n^{pr} , $1 \le n \le N$. The reduced basis space $W_{N_{pr}}$ is then defined as

$$W_{N_{pr}} = span\{\xi_n^{pr}\ ,\ 1 \le n \le N\}\ . \tag{7}$$

The reduced basis solution $u_N(\boldsymbol{\mu})$ can be expressed as

$$u_N(\boldsymbol{\mu}) = \sum_{j=1}^{N} u_{Nj}(\boldsymbol{\mu})\ \xi_j^{pr}\ . \tag{8}$$

Now we use the affine parameter dependence to construct an efficient offline/online strategy. Choosing test functions as $v = \xi_i^{pr},\ i = 1, \dots N$, $u_N(\boldsymbol{\mu})$ then satisfies

$$\sum_{j=1}^{N} \left(\sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}) a^q\left(\xi_j^{pr}, \xi_i^{pr}\right) \right)\ u_{N_j}(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}) f^q(\xi_i^{pr}) \right)\ , \tag{9}$$

which can also be written in a matrix form as

$$\left(\sum_{q=1}^{Q_a}\theta_a^q(\boldsymbol{\mu})A_N^q\right)u_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_f}\theta_f^q(\boldsymbol{\mu})F_N^q\right) \ , \tag{10}$$

where $(u_N(\boldsymbol{\mu}))_j = u_{N_j}(\boldsymbol{\mu})$,

$$(A_N^q)_{ij} = a^q\left(\xi_j^{pr},\xi_i^{pr}\right) \quad \text{and} \quad (F_N^q)_i = f^q(\xi_i^{pr}) \ . \tag{11}$$

The output can be expressed as

$$s_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_\ell}\theta_\ell^q(\boldsymbol{\mu})\ell^q(u_N(\boldsymbol{\mu}))\right) \ , \tag{12}$$

or in a vector form

$$s_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_\ell}\theta_\ell^q(\boldsymbol{\mu}){L_N^q}^T\right)u_N(\boldsymbol{\mu}) \ , \tag{13}$$

where $(L_N^q)_i = \ell^q(\xi_i^{pr})$.
The offline/online decomposition is clear. During the offline step we compute basis functions $u_{\mathcal{N}}(\boldsymbol{\mu})$, then form matrices A_N^q, vectors F_N^q and vectors L_N^q. During the online step, for a given parameter $\boldsymbol{\mu}$, we assemble the matrix $A_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a}\theta_a^q(\boldsymbol{\mu})A_N^q$, and the vectors $F_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f}\theta_f^q(\boldsymbol{\mu})F_N^q$ and $L_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_\ell}\theta_\ell^q(\boldsymbol{\mu})L_N^q$. We solve the system

$$A_N(\boldsymbol{\mu})\ u_N(\boldsymbol{\mu}) = F_N(\boldsymbol{\mu}) \ , \tag{14}$$

and finally we can evaluate the output as

$$s_N(\boldsymbol{\mu}) = L_N^T(\boldsymbol{\mu})\ u_N(\boldsymbol{\mu}) \ . \tag{15}$$

Up to now, we develop a *primal-only* approach to evaluate the output of interest, but we have not the quadratic convergence effect for outputs, except for compliant cases — a is symmetric and $\ell = f$. — We introduce now the *primal-dual* approach for non-compliant cases. The dual problem associated with the output of interest consists in finding $\Psi(\boldsymbol{\mu}) \in X$ such that

$$a\left(v,\Psi(\boldsymbol{\mu});\boldsymbol{\mu}\right) = -\ell(v;\boldsymbol{\mu}) \ \ \forall v \in X \ . \tag{16}$$

Ψ is denoted adjoint or dual field — note that in the compliant case $\Psi = -u$. — As in the primal case, we introduce the set S_N^Ψ that contains evaluation of adjoint for each $\boldsymbol{\mu}$ in the set S_N

$$S_N^\Psi = \{\Psi_{\mathcal{N}}(\boldsymbol{\mu}^i) \ , \ \forall \boldsymbol{\mu}^i \in S_N\} \ . \tag{17}$$

We apply the Gram-Schmidt process with respect to the $(.,.)_X$ inner product to elements of the set S_N^Ψ, the result are mutually $(.,.)_X$-orthonormal basis functions ξ_n^{du} , $1 \le n \le N$. The reduced basis space $W_{N_{du}}$ is then defined as

$$W_{N_{du}} = span\{\xi_n^{du} \ , \ 1 \le n \le N\} \ . \tag{18}$$

Note that we have chosen the segregated approach — we could have chosen the integrated one where we have a unique reduced basis space holding both primal and dual basis functions. — In that case, the evaluation of the output is given by

$$s_N(\boldsymbol{\mu}) = \ell\left(u_N(\boldsymbol{\mu});\boldsymbol{\mu}\right) - r_{pr}(\Psi_N(\boldsymbol{\mu});\boldsymbol{\mu}) \ . \tag{19}$$

with

$$r_{pr}(v;\boldsymbol{\mu}) = f(v;\boldsymbol{\mu}) - a(u_N(\boldsymbol{\mu}),v;\boldsymbol{\mu}) \ \text{ and } \ r_{du}(v;\boldsymbol{\mu}) = -\ell(v) - a(v,\Psi_N(\boldsymbol{\mu});\boldsymbol{\mu}) \ . \tag{20}$$

So now to have the reduced basis approximation, we need to solve the primal and a dual prolem :

$$a(u_N(\boldsymbol{\mu}),v;\boldsymbol{\mu}) = f(v;\boldsymbol{\mu}) \quad \forall v \in W_{N_{pr}} \ \text{ and } \ a(v,\Psi_N(\boldsymbol{\mu});\boldsymbol{\mu}) = -\ell(v;\boldsymbol{\mu}) \quad \forall v \in W_{N_{du}} \ . \tag{21}$$

1.2. A posteriori error estimation for elliptic linear problems with affine dependence in parameters

Thanks to (19) we can rapidly compute an estimation for the output of interest $s(\boldsymbol{\mu})$. In this section we introduce an a posteriori error bound that allow us to know if this output estimate is a good enough approximation of the output of interest (see for example [Prud'homme et al., 2002, Nguyen et al., 2009]).

1.2.1. *Ingredients*

We first introduce a positive lower bound $\alpha_{LB}^{\mathcal{N}}(\boldsymbol{\mu})$ for $\alpha^{\mathcal{N}}(\boldsymbol{\mu})$ for all $\boldsymbol{\mu}$ in $\mathcal{D}$, where $\alpha^{\mathcal{N}}(\boldsymbol{\mu})$ is the finite elements coercivity constant defined as

$$\alpha^{\mathcal{N}}(\boldsymbol{\mu}) = \inf_{w \in X^{\mathcal{N}}} \frac{a(w, w; \boldsymbol{\mu})}{||w||_X^2} \ \forall \boldsymbol{\mu} \in \mathcal{D}. \tag{22}$$

So we can write

$$0 \leq \alpha_{LB}^{\mathcal{N}}(\boldsymbol{\mu}) \leq \alpha^{\mathcal{N}}(\boldsymbol{\mu}) \ \ \forall \boldsymbol{\mu} \in \mathcal{D}, \tag{23}$$

where the online computational time to evaluate $\boldsymbol{\mu} \to \alpha_{LB}^{\mathcal{N}}(\boldsymbol{\mu})$ has to be independent of $\mathcal{N}$ to compute efficiency the error bound that we introduce here. The successive constraints method (see [Huynh et al., 2007]) determine efficiency the lower bound $\alpha_{LB}^{\mathcal{N}}(\boldsymbol{\mu})$. Note that in this paper we introduce the SCM because it is implemented in `Feel++`. There exists other ways to determine $\alpha_{LB}^{\mathcal{N}}(\boldsymbol{\mu})$, for example by inspection.

1.2.2. *Successive Constraints Method*

First we recall that from (3) we assume that the parametrized bilinear form $a(.,.;\boldsymbol{\mu})$ depend affinely in $\boldsymbol{\mu}$ and can be expressed as

$$a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \theta_a^q(\boldsymbol{\mu}) a^q(u(\boldsymbol{\mu}), v) \quad \forall u(\boldsymbol{\mu}), v \in X, \ \forall \boldsymbol{\mu} \in \mathcal{D} \ . \tag{24}$$

In order to have an offline/online strategy, we will reformulate the expression of the coercivity constant (22) by introducing the objective function $\mathcal{J}^{obj} : \mathcal{D} \times \mathbb{R}^{Q_a} \to \mathbb{R}$ given by $\mathcal{J}^{obj}(\boldsymbol{\mu}; y) = \sum_{q=1}^{Q_a} \theta^q(\boldsymbol{\mu}) \ y^q$ where $y \in \mathbb{R}^{Q_a}$. Then the FE coercivity constant can be defined by

$$\alpha^{\mathcal{N}}(\boldsymbol{\mu}) = \inf_{y \in \mathcal{Y}} \mathcal{J}^{obj}(\boldsymbol{\mu}; y) \ , \tag{25}$$

where the set $\mathcal{Y} \in \mathbb{R}^{Q_a}$ is defined by

$$\mathcal{Y} = \left\{ y \in \mathbb{R}^{Q_a} \mid \exists w_y \in X^{\mathcal{N}} s.t. \ y^q = \frac{a^q(w_y, w_y)}{||w_y||_X^2} \ , \ 1 \leq q \leq Q_a \right\} \ . \tag{26}$$

We also introduce C_J as the "coercivity constraint" sample

$$C_J = \left\{ \boldsymbol{\mu}^1 \in \mathcal{D}, ..., \boldsymbol{\mu}^J \in \mathcal{D} \right\} \ . \tag{27}$$

Lower bound. To determine the lower bound $\alpha_{LB}^{\mathcal{N}}(\boldsymbol{\mu})$ we need to define the "continuity constraint" box

$$\mathcal{B} = \prod_{q=1}^{Q_a} \left[\inf_{w \in X^{\mathcal{N}}} \frac{a^q(w, w)}{||w||_X^2} \ , \ \sup_{w \in X^{\mathcal{N}}} \frac{a^q(w, w)}{||w||_X^2} \right] \ . \tag{28}$$

Let $C_J^{M,\boldsymbol{\mu}}$ the set of $M(\geq 1)$ points in C_J closest in the euclidean norm to a given $\boldsymbol{\mu} \in \mathcal{D}$. The lower bound of the coercivity constant $\alpha^{\mathcal{N}}(\boldsymbol{\mu})$ is defined as

$$\alpha_{LB}^{\mathcal{N}}(\boldsymbol{\mu}; C_J, M) = \min_{y \in \mathcal{Y}_{LB}(\boldsymbol{\mu}, C_J, M)} \mathcal{J}^{obj}(\boldsymbol{\mu}; y) \ , \tag{29}$$

where the "lower bound" set $\mathcal{Y}_{LB}(\boldsymbol{\mu}; C_J, M) \in \mathbb{R}^{Q_a}$ is defined as

$$\mathcal{Y}_{LB}(\boldsymbol{\mu}; C_J, M) \equiv \left\{ y \in \mathbb{R}^{Q_a} \mid y \in \mathcal{B}, \sum_{q=1}^{Q_a} \theta^q(\boldsymbol{\mu}') y^q \geq \alpha^{\mathcal{N}}(\boldsymbol{\mu}') \ , \ \forall \boldsymbol{\mu}' \in C_J^{M,\boldsymbol{\mu}} \right\} \ . \tag{30}$$

The lower bound defined in (29) is a linear optimisation problem, or Linear Program (LP). We observe that the LP (29) contains Q design variables and 2 $Q + M$ inequality constraints. It is important to note that for a given $\mathcal{B}$ and the set $\{\alpha^{\mathcal{N}}(\boldsymbol{\mu}') \mid \boldsymbol{\mu}' \in C_J\}$, the operation count to evaluate $\boldsymbol{\mu} \to \alpha_{LB}^{\mathcal{N}}(\boldsymbol{\mu})$ is independent of $\mathcal{N}$. In actual practice, we have developed a more effcient SCM strategy to build our lower bound, see [Vallaghé et al., 2011].

Upper bound. We have now a lower bound for the coercivity constraint, but to build the sample C_J we also need an upper bound $\alpha_{UB}^{\mathcal{N}}$ of this constant. The lower bound of the coercivity constant $\alpha^{\mathcal{N}}(\boldsymbol{\mu})$ is defined as

$$\alpha_{UB}^{\mathcal{N}}(\boldsymbol{\mu}; C_J, M) = \min_{y \in \mathcal{Y}_{UB}(\boldsymbol{\mu}, C_J, M)} \mathcal{J}^{obj}(\boldsymbol{\mu}; y) \ . \tag{31}$$

where for given C_J , M $\in \mathbb{N}$ and any $\boldsymbol{\mu} \in \mathcal{D}$ we introduce an " upper bound " set $\mathcal{Y}_{UB}(\boldsymbol{\mu}; C_J, M) \in \mathbb{R}^{Q_a}$ as

$$\mathcal{Y}_{UB}(\boldsymbol{\mu}; C_J, M) \equiv \left\{ arg \inf_{y \in \mathcal{Y}} \mathcal{J}^{obj}(\boldsymbol{\mu}; y) \mid \boldsymbol{\mu}' \in C_J^{M,\boldsymbol{\mu}} \right\} \ . \tag{32}$$

We note that to evaluate $\boldsymbol{\mu} \to \alpha_{UB}^{\mathcal{N}}(\boldsymbol{\mu})$ is independent of $\mathcal{N}$.

Construction of the set C_J. Thanks to the previous ingredients, we construct now the set C_J using an offline greedy algorithm. First we require a sample $\Xi_{train} = \{\boldsymbol{\mu}^1, ..., \boldsymbol{\mu}^{n_{train}}\} \subset \mathcal{D}$ of n_{train} parameters. A tolerance $\epsilon \in (0, 1)$ is also required to control the error in the lower bound prediction. Start by taking $J = 1$ and choosing $C_1 = \boldsymbol{\mu}^1$ arbitrarily. Then from $J = 2$, the J^{th} parameter $\boldsymbol{\mu}^J$ selected maximizes the gap between the lower bound and the upper bound of the coercivity constant. That is to say that we perform the algorithm 1.

Algorithm 1 Offline greedy algorithm

while $\max_{\boldsymbol{\mu} \in \Xi_{train}} \left[\dfrac{\alpha_{UB}^{\mathcal{N}}(\boldsymbol{\mu}; C_J, M) - \alpha_{LB}^{\mathcal{N}}(\boldsymbol{\mu}; C_J, M)}{\alpha_{UB}^{\mathcal{N}}(\boldsymbol{\mu}; C_J, M)} \right] > \epsilon$ **do**
  $\boldsymbol{\mu}^{J+1} = arg \max_{\boldsymbol{\mu} \in \Xi_{train}} \left[\dfrac{\alpha_{UB}^{\mathcal{N}}(\boldsymbol{\mu}; C_J, M) - \alpha_{LB}^{\mathcal{N}}(\boldsymbol{\mu}; C_J, M)}{\alpha_{UB}^{\mathcal{N}}(\boldsymbol{\mu}; C_J, M)} \right]$
  $C_{J+1} \leftarrow C_J \cup \boldsymbol{\mu}^{J+1}$
  $J \leftarrow J + 1$
end while

At each iteration we add the parameter $\boldsymbol{\mu} \in \mathcal{D}$ that have the worse lower bound approximation to the "coercivity constraint" sample. As for each $\boldsymbol{\mu} \in C_J$ we have $\alpha_{UB}^{\mathcal{N}}(\boldsymbol{\mu}; C_J, M) = \alpha_{LB}^{\mathcal{N}}(\boldsymbol{\mu}; C_J, M)$ it follows from continuity considerations that, for sufficiently large number of iterations, the error tolerance ϵ is reached.

Offline/Online strategy of the SCM. This method has obviously an offline/online strategy. During the offline step are computed eigenvalues which serve as bounds of the "continuity constraint" box (28). To determine the upper bound $\alpha_{UB}^{\mathcal{N}}(\boldsymbol{\mu})$, the vector $y \in \mathbb{R}^{Q_a}$, element of the set $\mathcal{Y}$ (see 26), is built during the offline step. Then for a given $\boldsymbol{\mu} \in \mathcal{D}$ we determine $\alpha_{UB}^{\mathcal{N}}(\boldsymbol{\mu})$ via (31). To apply constraints needed fo the construction of the set $\mathcal{Y}_{LB}$ (see 30) we compute eigenvalues associated to parameters $\boldsymbol{\mu}'$ during the offline step.

1.2.3. *A posteriori error estimators*

Now that we have a lower bound for the coercivity constant, we introduce the dual norm of the primal (resp. dual) residual $\epsilon_{N_{pr}}(\boldsymbol{\mu})$ (resp. $\epsilon_{N_{du}}(\boldsymbol{\mu})$), defined as

$$\epsilon_{N_{pr}}(\boldsymbol{\mu}) \equiv \sup_{v \in X^{\mathcal{N}}} \frac{r_{pr}(v;\boldsymbol{\mu})}{||v||_X} = ||\hat{e}_{pr}(\boldsymbol{\mu})||_X \quad \text{and} \quad \epsilon_{N_{du}}(\boldsymbol{\mu}) \equiv \sup_{v \in X^{\mathcal{N}}} \frac{r_{du}(v;\boldsymbol{\mu})}{||v||_X} = ||\hat{e}_{du}(\boldsymbol{\mu})||_X \ . \tag{33}$$

In addition to the dual norm of residuals, equation (33) also introduced the Riesz representation of primal (resp. dual) residual : $\hat{e}_{pr}(\boldsymbol{\mu})$ (resp. $\hat{e}_{du}(\boldsymbol{\mu})$). Now we define the a posteriori error estimation in terms of the dual norm of residuals and the lower bound for the coercivity constant. In particular for all $\boldsymbol{\mu} \in \mathcal{D}$ and all N we have

$$|s_{\mathcal{N}}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})| \leq \Delta_N^s(\boldsymbol{\mu}) \ , \tag{34}$$

where the a posteriori error estimator on the output $\Delta_N^s(\boldsymbol{\mu})$ is given by

$$\Delta_N^s(\boldsymbol{\mu}) = \sqrt{\frac{\epsilon_{N_{pr}}(\boldsymbol{\mu})^2}{\alpha_{LB}^{\mathcal{N}}(\boldsymbol{\mu})}} \sqrt{\frac{\epsilon_{N_{du}}(\boldsymbol{\mu})^2}{\alpha_{LB}^{\mathcal{N}}(\boldsymbol{\mu})}} \ . \tag{35}$$

Note that (35) can be seen as the product of error estimators on primal and dual solutions.

1.2.4. *Offline / online strategy*

We now turn to the description of the offline/online strategy for the a posteriori error estimator introduced by (35). We start with the dual norm of the residuals, starting by recalling the expression of the primal residual

$$r_{pr}(v;\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}) f^q(v) - \sum_{q=1}^{Q_a} \sum_{j=1}^{N} \theta_a^q(\boldsymbol{\mu}) u_{Nj}(\boldsymbol{\mu}) a^q(\xi_j^{pr}, v), \quad \forall v \in X^{\mathcal{N}}. \tag{36}$$

The Riesz representation $\hat{e}_{pr}(\boldsymbol{\mu})$ verifies

$$(\hat{e}_{pr}(\boldsymbol{\mu}), v)_X = \sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}) f^q(v) - \sum_{q=1}^{Q_a} \sum_{j=1}^{N} \theta_a^q(\boldsymbol{\mu}) u_{Nj}(\boldsymbol{\mu}) a^q(\xi_j^{pr}, v), \quad \forall v \in X^{\mathcal{N}} \tag{37}$$

and thus, using linear superposition, it reads

$$\hat{e}_{pr}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \theta_f^q(\boldsymbol{\mu}) \Gamma_{N_{pr}}^q + \sum_{q=1}^{Q_a} \sum_{j=1}^{N} \theta_a^q(\boldsymbol{\mu}) u_{Nj}(\boldsymbol{\mu}) \Upsilon_{N_{pr}}^{qj} \ , \tag{38}$$

where

$$\begin{array}{rcll} (\Gamma_{N_{pr}}^q, v)_X & = & f^q(v) & \forall v \in X^{\mathcal{N}} \ , \ 1 \leq q \leq Q_f \ , \\ (\Upsilon_{N_{pr}}^{qj}, v)_X & = & -a^q(\xi_j^{pr}, v) & \forall v \in X^{\mathcal{N}} \ , \ 1 \leq q \leq Q_a \ , \ 1 \leq j \leq N \ . \end{array} \tag{39}$$

Consequently we have

$$\epsilon_{N_{pr}}(\boldsymbol{\mu})^2 = C^{ff}_{N_{pr}}(\boldsymbol{\mu}) + 2\sum_{j=1}^{N} u_{Nj} C^{fa}_{N_{pr}j}(\boldsymbol{\mu}) + \sum_{j=1}^{N}\sum_{j'=1}^{N} u_{Nj}u_{Nj'} C^{aa}_{N_{pr}jj'}(\boldsymbol{\mu}) \ , \tag{40}$$

where for $1 \le j, j' \le N$:

$$\begin{aligned} C^{ff}_{N_{pr}}(\boldsymbol{\mu}) &= \sum_{q=1}^{Q_f}\sum_{q'=1}^{Q_f} \theta_f^q(\boldsymbol{\mu})\ \theta_f^{q'}(\boldsymbol{\mu})\ \Phi^{qq'}_{N_{pr}} \ , \\ C^{fa}_{N_{pr}j}(\boldsymbol{\mu}) &= \sum_{q=1}^{Q_a}\sum_{q'=1}^{Q_f} \theta_a^q(\boldsymbol{\mu})\ \theta_f^{q'}(\boldsymbol{\mu})\ \Phi^{qjq'}_{N_{pr}} \ , \\ C^{aa}_{N_{pr}jj'}(\boldsymbol{\mu}) &= \sum_{q=1}^{Q_a}\sum_{q'=1}^{Q_a} \theta_a^q(\boldsymbol{\mu})\ \theta_a^{q'}(\boldsymbol{\mu})\ \Phi^{qjq'j'}_{N_{pr}} \ . \end{aligned} \tag{41}$$

Here $\Phi^{qq'}_{N_{pr}}$ ($1 \le q, q' \le Q_f$) , $\Phi^{qjq'}_{N_{pr}}$ ($1 \le q \le Q_a$,$1 \le j \le N$, $1 \le q' \le Q_f$) and $\Phi^{qjq'j'}_{N_{pr}}$ ($1 \le q, q' \le Q_a$,$1 \le j, j' \le N$) do not depend on parameter $\boldsymbol{\mu}$ and are defined as

$$\Phi^{qq'}_{N_{pr}} = (\Gamma^q_{N_{pr}}, \Gamma^{q'}_{N_{pr}})_X \quad , \quad \Phi^{qjq'}_{N_{pr}} = (\Upsilon^{qj}_{N_{pr}}, \Gamma^{q'}_{N_{pr}})_X \quad \text{and} \quad \Phi^{qjq'j'}_{N_{pr}} = (\Upsilon^{qj}_{N_{pr}}, \Upsilon^{q'j'}_{N_{pr}})_X \quad . \tag{42}$$

Now let us consider the expression of the dual residual

$$r_{du}(v;\boldsymbol{\mu}) = -\sum_{q=1}^{Q_l} \theta_l^q(\boldsymbol{\mu}) l^q(v) - \sum_{q=1}^{Q_a}\sum_{j=1}^{N} \theta_a^q(\boldsymbol{\mu})\Psi_{Nj}(\boldsymbol{\mu}) a^q(v, \xi_j^{du}), \quad \forall v \in X^{\mathcal{N}}. \tag{43}$$

The Riesz representation $\hat{e}_{du}(\boldsymbol{\mu})$ reads

$$\hat{e}_{du}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_l} \theta_l^q(\boldsymbol{\mu})\Gamma^q_{N_{du}} + \sum_{q=1}^{Q_a}\sum_{j=1}^{N} \theta_a^q(\boldsymbol{\mu})\Psi_{Nj}(\boldsymbol{\mu})\Upsilon^{qj}_{N_{du}} \ , \tag{44}$$

where

$$\begin{aligned} (\Gamma^q_{N_{du}}, v)_X &= -l^q(v) \qquad \forall v \in X^{\mathcal{N}} \ , \ 1 \le q \le Q_l \ , \\ (\Upsilon^{qj}_{N_{du}}, v)_X &= -a^q(v, \xi_j^{du}) \quad \forall v \in X^{\mathcal{N}} \ , \ 1 \le q \le Q_a \ , \ 1 \le j \le N \ . \end{aligned} \tag{45}$$

Consequently $\hat{e}_{du}(\boldsymbol{\mu})^2$ can be written as

$$\epsilon_{N_{du}}(\boldsymbol{\mu})^2 = C^{ll}_{N_{du}}(\boldsymbol{\mu}) + 2\sum_{j=1}^{N} \Psi_{Nj} C^{la}_{N_{du}j}(\boldsymbol{\mu}) + \sum_{j=1}^{N}\sum_{j'=1}^{N} \Psi_{Nj}\Psi_{Nj'} C^{aa}_{N_{du}jj'}(\boldsymbol{\mu}) \ , \tag{46}$$

where for $1 \le j, j' \le N$:

$$\begin{aligned} C^{ff}_{N_{du}}(\boldsymbol{\mu}) &= \sum_{q=1}^{Q_f}\sum_{q'=1}^{Q_f} \theta_f^q(\boldsymbol{\mu})\ \theta_f^{q'}(\boldsymbol{\mu})\ \Phi^{qq'}_{N_{du}} \ , \\ C^{fa}_{N_{du}j}(\boldsymbol{\mu}) &= \sum_{q=1}^{Q_a}\sum_{q'=1}^{Q_f} \theta_a^q(\boldsymbol{\mu})\ \theta_f^{q'}(\boldsymbol{\mu})\ \Phi^{qjq'}_{N_{du}} \ , \\ C^{aa}_{N_{du}jj'}(\boldsymbol{\mu}) &= \sum_{q=1}^{Q_a}\sum_{q'=1}^{Q_a} \theta_a^q(\boldsymbol{\mu})\ \theta_a^{q'}(\boldsymbol{\mu})\ \Phi^{qjq'j'}_{N_{du}} \ . \end{aligned} \tag{47}$$

Here $\Phi^{qq'}_{N_{du}}$ ($1 \leq q, q' \leq Q_f$) , $\Phi^{qjq'}_{N_{du}}$ ($1 \leq q \leq Q_a$,$1 \leq j \leq N$, $1 \leq q' \leq Q_f$) and $\Phi^{qjq'j'}_{N_{du}}$ ($1 \leq q, q' \leq Q_a$,$1 \leq j, j' \leq N$) do not depend on parameter $\boldsymbol{\mu}$ and are defined as

$$\Phi^{qq'}_{N_{du}} = (\Gamma^{q}_{N_{du}}, \Gamma^{q'}_{N_{du}})_X \quad , \quad \Phi^{qjq'}_{N_{du}} = (\Upsilon^{qj}_{N_{du}}, \Gamma^{q'}_{N_{du}})_X \quad \text{and} \quad \Phi^{qjq'j'}_{N_{du}} = (\Upsilon^{qj}_{N_{du}}, \Upsilon^{q'j'}_{N_{du}})_X \quad . \tag{48}$$

During the offline stage, we compute all $\boldsymbol{\mu}$-independent quantities and we store them in a database. During the online step, for a given $\boldsymbol{\mu} \in \mathcal{D}$, we evaluate the $\boldsymbol{\mu}$-dependent terms and assemble the residual norm terms using the online solution.

1.3. **Empirical Interpolation Method**

We describe here the Empirical Interpolation Method (see [Barrault et al., 2004], [Grepl et al., 2007]). This method is classically used to build an approximation of the affine decomposition needed by the RB method. Naturally, EIM is an essential tool to deal with non-linear models, since the associated affine decomposition can not be written by direct inspection of bilinear and linear forms (see example in section 3). In our computational framework, EIM is always used in our applications (with or without affine dependence in parameters). In the case of linear problems with affine parameter dependence, this method gives exactly the same affine decomposition as in the case of direct inspection of bilinear or linear forms. This allows to have a unified interface to access our RB framework from the applications.

We consider the non-linear $\boldsymbol{\mu}$-dependent function of sufficient regularity $\eta\left(\boldsymbol{\mu}; \boldsymbol{x}; \boldsymbol{u}(\boldsymbol{\mu})\right)$ which is non-affine in parameters and depends on the solution of a parametrized PDE $\boldsymbol{u}(\boldsymbol{\mu})$. We are interested in approximating $\eta\left(\boldsymbol{\mu}; \boldsymbol{x}; \boldsymbol{u}(\boldsymbol{\mu})\right)$ by a reduced-basis expansion $\eta_M\left(\boldsymbol{\mu}; \boldsymbol{x}; \boldsymbol{u}(\boldsymbol{\mu})\right)$ which is affine in the parameters such that

$$\eta_M\left(\boldsymbol{\mu}; \boldsymbol{x}; \boldsymbol{u}(\boldsymbol{\mu})\right) = \sum_{m=1}^{M} \beta_m(\boldsymbol{\mu}; \boldsymbol{u}(\boldsymbol{\mu})) q_m(x) \ . \tag{49}$$

To this end we introduce a nested sample set $S_M = \{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_M\} \in \mathcal{D}^M$ and associated nested space $W_M = span\{\boldsymbol{\xi}_m \equiv \eta\left(\boldsymbol{\mu}_m; \boldsymbol{x}; \boldsymbol{u}(\boldsymbol{\mu}_m)\right),\ 1 \leq m \leq M\}$ in which the approximation $\eta_M\left(\boldsymbol{\mu}; \boldsymbol{x}; \boldsymbol{u}(\boldsymbol{\mu})\right)$ shall reside. We first introduce Ξ a suitably large but finite-dimensional parameter set in $\mathcal{D}$. The first sample point $\boldsymbol{\mu}_1$ is picked into Ξ and assuming that $\boldsymbol{\xi}_1 \neq 0$, we define :

$$S_M = \{\boldsymbol{\mu}_1\}, \quad \boldsymbol{\xi}_1 \equiv \eta\left(\boldsymbol{\mu}_1; \boldsymbol{x}; \boldsymbol{u}(\boldsymbol{\mu}_1)\right) \quad \text{and} \quad W_M = span\{\boldsymbol{\xi}_1\} \ .$$

For $M \geqslant 2$, we determine $\boldsymbol{\mu}_M$ from a Greedy algorithm and deduce the associated $\boldsymbol{\xi}_M$:

$$\boldsymbol{\mu}_M = arg \max_{\boldsymbol{\mu} \in \Xi} \inf_{z \in W_{M-1}} ||\eta(\boldsymbol{\mu}; .; .) - z||_{L^\infty(\Omega)}, \quad \text{and} \quad \boldsymbol{\xi}_M = \eta\left(\boldsymbol{\mu}_M; \boldsymbol{x}; \boldsymbol{u}(\boldsymbol{\mu}_M)\right) \tag{50}$$

from which we complete $S_M = S_{M-1} \cup \{\boldsymbol{\mu}_M\}$ and $W_M = W_{M-1} \oplus span\{\boldsymbol{\xi}_M\}$.

The coefficients β_m of the particular linear combination (49) are determined through interpolation points $t_1, \ldots, t_M \in \Omega$ such that

$$\sum_{m=1}^{M} \beta_m(\boldsymbol{\mu}; \boldsymbol{u}(\boldsymbol{\mu})) q_m(\boldsymbol{t}_i) = \eta(\boldsymbol{\mu}; \boldsymbol{t}_i; \boldsymbol{u}(\boldsymbol{\mu})) \quad \forall\ \boldsymbol{t}_i\ (i = 1 \ldots M) \ . \tag{51}$$

The first interpolation point $\boldsymbol{t}_1$ is chosen such that first basis function $\boldsymbol{\xi}_1$ is maximum, and $\boldsymbol{q}_1$ consists in the normalization of $\boldsymbol{\xi}_1$:

$$\boldsymbol{t}_1 = arg \sup_{\boldsymbol{x} \in \Omega} |\boldsymbol{\xi}_1(\boldsymbol{x})|, \quad \boldsymbol{q}_1 = \frac{\boldsymbol{\xi}_1(\boldsymbol{x})}{\boldsymbol{\xi}_1(\boldsymbol{t}_1)} \quad \text{and} \quad B^1_{11} = \boldsymbol{q}_1(\boldsymbol{t}_1) = 1 \ . \tag{52}$$

For $M \geqslant 2$, we look for the vector $\boldsymbol{\sigma}^{M-1} = \left(\sigma_i^{M-1}\right)_{i=1,\dots,M-1}$ obtained through interpolation points

$$\sum_{j=1}^{M-1} \sigma_j^{M-1} \boldsymbol{q}_j(\boldsymbol{t}_i) = \boldsymbol{\xi}_M(\boldsymbol{t}_i) \ , \quad 1 \leq i \leq M-1 \ , \tag{53}$$

and then we compute the residual

$$\boldsymbol{r}_M(\boldsymbol{x}) = \boldsymbol{\xi}_M(\boldsymbol{x}) - \sum_{j=1}^{M-1} \sigma_j^{M-1} \boldsymbol{q}_j(\boldsymbol{x}) \ . \tag{54}$$

which allows to compute the next interpolation point t_M together with the basis functions $\boldsymbol{q}_M(x)$ for $M \geqslant 2$ along with the interpolation matrix B^M

$$\boldsymbol{t}_M = arg \sup_{\boldsymbol{x} \in \Omega} |\boldsymbol{r}_M(\boldsymbol{x})|, \quad \boldsymbol{q}_M(\boldsymbol{x}) = \frac{\boldsymbol{r}_M(\boldsymbol{x})}{\boldsymbol{r}_M(\boldsymbol{t}_M)} \quad \text{and} \quad B_{ij}^M = \boldsymbol{q}_j(\boldsymbol{t}_i) \ , \quad 1 \leq i,j \leq M \ . \tag{55}$$

Once all the interpolation points t_M and the basis functions $\boldsymbol{q}_M$ have been computed (offline part), the computation of the approximation $\eta_M(\boldsymbol{\mu}; \boldsymbol{x}; \boldsymbol{u}(\boldsymbol{\mu}))$ for a given $\boldsymbol{\mu}$ (online part) consists in finding the coefficients $\beta_m(\boldsymbol{\mu}, \boldsymbol{u}(\boldsymbol{\mu}))$ of the linear combination by solving :

$$\sum_{j=1}^{M} B_{ij}^M \beta_j \left(\boldsymbol{\mu}; \boldsymbol{u}(\boldsymbol{\mu})\right) = \eta \left(\boldsymbol{\mu}; \boldsymbol{t}_i; \boldsymbol{u}(\boldsymbol{\mu})\right) \ , \quad 1 \leq i \leq M \ . \tag{56}$$

In other words, coefficients of EIM expansion are solutions of

$$\begin{pmatrix} \boldsymbol{q}_1(\boldsymbol{t}_1) & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \boldsymbol{q}_1(\boldsymbol{t}_{M-1}) & \cdots & \boldsymbol{q}_{M-1}(\boldsymbol{t}_{M-1}) & 0 \\ \boldsymbol{q}_1(\boldsymbol{t}_M) & \cdots & \cdots & \boldsymbol{q}_M(\boldsymbol{t}_M) \end{pmatrix} \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_{M-1} \\ \beta_M \end{pmatrix} = \begin{pmatrix} \eta \left(\boldsymbol{\mu}; \boldsymbol{t}_1; \boldsymbol{u}(\boldsymbol{\mu})\right) \\ \vdots \\ \eta \left(\boldsymbol{\mu}; \boldsymbol{t}_{M-1}; \boldsymbol{u}(\boldsymbol{\mu})\right) \\ \eta \left(\boldsymbol{\mu}; \boldsymbol{t}_M; \boldsymbol{u}(\boldsymbol{\mu})\right) \end{pmatrix} \ . \tag{57}$$

Remark 1. *In the most general case, the function η depends not only on $\boldsymbol{x}$ but also on the solution $\boldsymbol{u}(\boldsymbol{\mu})$ of the problem. During the online step, we need to evaluate quickly the function η at interpolation points (see 57). Consequently, it implies to have a rapid evaluation of $\boldsymbol{u}(\boldsymbol{\mu})$ which is in fact replaced by $\boldsymbol{u}_N(\mu)$ and we have precomputed the reduced basis functions associated to $\boldsymbol{u}_N(\boldsymbol{\mu})$ at the interpolation points $(\boldsymbol{t}_i)_{i=1,\dots,M}$.*

1.4. **Elliptic non-affinely parametrized non-linear equations**

1.4.1. *General problem settings*

We consider the following problem : given $\boldsymbol{\mu} \in \mathcal{D} \subset \mathbb{R}^p$, evaluate the output of interest :

$$s(\boldsymbol{\mu}) = \ell \left(\boldsymbol{u}(\boldsymbol{\mu}); \boldsymbol{\mu}\right) \ , \tag{58}$$

where $u(\boldsymbol{\mu}) \in X$ satisfies

$$g(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}) = 0 \quad \forall v \in X \ . \tag{59}$$

Here we consider that (59) is a non-linear system of N_{equs} equations. To deal with (59), the Newton algorithm is used to find zero of $g(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu})$. Let ${}^k\boldsymbol{u}(\boldsymbol{\mu})$ the solution at the k^{th} iteration of Newton algorithm, $g\left({}^k\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}\right)$ the function g applied to the solution ${}^k\boldsymbol{u}(\boldsymbol{\mu})$ and $j\left(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}; {}^k\boldsymbol{u}(\boldsymbol{\mu})\right)$ the jacobian of

$g(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu})$ applied to the solution ${}^k\boldsymbol{u}(\boldsymbol{\mu})$. It reads

$$j\left(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}; {}^k\boldsymbol{u}(\boldsymbol{\mu})\right) = \begin{pmatrix} \dfrac{\partial g_1\left(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}; {}^k\boldsymbol{u}(\boldsymbol{\mu})\right)}{\partial u_1} & \cdots & \dfrac{\partial g_1\left(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}; {}^k\boldsymbol{u}(\boldsymbol{\mu})\right)}{\partial u_{N_{\text{equs}}}} \\ & \ddots & \\ \dfrac{\partial g_{N_{\text{equs}}}\left(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}; {}^k\boldsymbol{u}(\boldsymbol{\mu})\right)}{\partial u_1} & \cdots & \dfrac{\partial g_{N_{\text{equs}}}\left(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}; {}^k\boldsymbol{u}(\boldsymbol{\mu})\right)}{\partial u_{N_{\text{equs}}}} \end{pmatrix} . \tag{60}$$

Now, we make the assumption for the offline/online procedure by assuming that we can approximate $g\left(.,.;\boldsymbol{\mu}\right)$, $j\left(.,.;\boldsymbol{\mu};{}^k\boldsymbol{u}(\boldsymbol{\mu})\right)$ and the linear form $\ell(.;\boldsymbol{\mu})$ respectively by $g_{AD}\left(.,.;\boldsymbol{\mu}\right)$, $j_{AD}\left(.,.;\boldsymbol{\mu};{}^k\boldsymbol{u}(\boldsymbol{\mu})\right)$ and $\ell_{AD}(.;\boldsymbol{\mu})$ which are functions that are affine in parameters $\boldsymbol{\mu}$. That is to say that for given Q_g , Q_j and Q_ℓ the EIM determines $(M_q^g)_{q=1,..,Q_g}$, $(M_q^j)_{q=1,..,Q_j}$ and $(M_q^\ell)_{q=1,..,Q_\ell}$ such that we can write :

$$g_{AD}\left({}^k\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}\right) = \sum_{q=1}^{Q_g} \sum_{m=1}^{M_q^g} \beta_g^{qm}(\boldsymbol{\mu}; {}^k\boldsymbol{u}(\boldsymbol{\mu}))\ g^{qm}(\boldsymbol{v}) \ , \tag{61}$$

$$j_{AD}\left(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}; {}^k\boldsymbol{u}(\boldsymbol{\mu})\right) = \sum_{q=1}^{Q_j} \sum_{m=1}^{M_q^j} \beta_j^{qm}(\boldsymbol{\mu}; {}^k\boldsymbol{u}(\boldsymbol{\mu}))\ j^{qm}(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{v}) \ , \tag{62}$$

and

$$\ell_{AD}(\boldsymbol{v}; \boldsymbol{\mu}) = \sum_{q=1}^{Q_\ell} \sum_{m=1}^{M_q^\ell} \beta_\ell^{qm}(\boldsymbol{\mu})\ \ell^{qm}(\boldsymbol{v}) \ . \tag{63}$$

The Newton method is an iterative method (see algorithm 2). Starting with an initial guess which is reasonably close to the true root the method consists, for each step, in solving the system of linear equations

$$j_{AD}\left(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}; {}^k\boldsymbol{u}(\boldsymbol{\mu})\right) \quad \delta^k u(\boldsymbol{\mu}) = -g_{AD}\left({}^k\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}\right) \ , \tag{64}$$

for the increment $\delta^k\boldsymbol{u}(\boldsymbol{\mu})$ defined by

$$\delta^k u(\boldsymbol{\mu}) = \left({}^{k+1}\boldsymbol{u}(\boldsymbol{\mu}) - {}^k\boldsymbol{u}(\boldsymbol{\mu})\right) \ . \tag{65}$$

Algorithm 2 $\boldsymbol{u}(\boldsymbol{\mu}) = Newton(g({}^k\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}), j(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}; {}^k\boldsymbol{u}(\boldsymbol{\mu})), initial\_guess, tol)$

k $\leftarrow 1$
${}^1\boldsymbol{u}(\boldsymbol{\mu}) \leftarrow$ initial_guess
e $\leftarrow \infty$
while e $>$ tol **do**
  solve $j\left(\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}; {}^k\boldsymbol{u}(\boldsymbol{\mu})\right) \quad \delta^k\boldsymbol{u}(\boldsymbol{\mu}) = -g\left({}^k\boldsymbol{u}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}\right)$
  $e \leftarrow \left|\frac{{}^{k+1}\boldsymbol{u}(\boldsymbol{\mu}) - {}^k\boldsymbol{u}(\boldsymbol{\mu})}{{}^k\boldsymbol{u}(\boldsymbol{\mu})}\right|$
  $k \leftarrow k+1$
end while
$\boldsymbol{u}(\boldsymbol{\mu}) \leftarrow {}^{k+1}\boldsymbol{u}(\boldsymbol{\mu})$

1.4.2. *Reduced basis method*

The finite element discretization of problem (58)-(59) is as follows : given a $\boldsymbol{\mu} \in \mathcal{D}$, evaluate

$$s_{\mathcal{N}}\left(\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu})\right) = \ell_{AD}\left(\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}); \boldsymbol{\mu}\right) \ , \tag{66}$$

where $\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}) \in X^{\mathcal{N}}$ satisfies

$$g_{AD}(\boldsymbol{u}_{\mathcal{N}}(\boldsymbol{\mu}), \boldsymbol{v}; \boldsymbol{\mu}) = 0 \quad \forall \boldsymbol{v} \in X^{\mathcal{N}}. \tag{67}$$

Here is mentioned $g_{AD}(.,.;\boldsymbol{\mu})$ instead of $g(.,.;\boldsymbol{\mu})$ to recall that we use approximations described in (61),(62) and (63). The reduced basis solution $\boldsymbol{u}_N(\boldsymbol{\mu})$ can be expressed as

$$\boldsymbol{u}_N(\boldsymbol{\mu}) = \sum_{i=1}^{N} u_{Ni}(\boldsymbol{\mu}) \ \boldsymbol{\xi}_i \ . \tag{68}$$

Considering the k^{th} iteration of the Newton algorithm, by taking $\boldsymbol{v} = \boldsymbol{\xi}_i$ for $i = 1, ..., N$ and using (68) we can write

$$\sum_{n=1}^{N} j_{AD}\left(\boldsymbol{\xi}_n, \boldsymbol{\xi}_i; \boldsymbol{\mu}; {}^k\boldsymbol{u}_N(\boldsymbol{\mu})\right) \ \delta^k u_{Nn}(\boldsymbol{\mu}) = -g_{AD}\left({}^k\boldsymbol{u}_N(\boldsymbol{\mu}), \boldsymbol{\xi}_i; \boldsymbol{\mu}\right) \ , \tag{69}$$

where $\delta^k u_{Nn}(\boldsymbol{\mu})$ is the n^{th} component of the increment, defined as : $\delta^k u_{Nn}(\boldsymbol{\mu}) = {}^{k+1}u_{Nn} - {}^k u_{Nn}$. Now we use the affine parameter dependence to construct an efficient offline/online strategy. Choosing test functions as $v = \boldsymbol{\xi}_i, \ i = 1, \dots N$, the equation (69) can be expressed as

$$\sum_{n=1}^{N} \left(\sum_{q=1}^{Q_j} \sum_{m=1}^{M_q^j} \beta_j^{qm}(\boldsymbol{\mu}; {}^k\boldsymbol{u}_N(\boldsymbol{\mu})) \ j^{qm}(\boldsymbol{\xi}_n, \boldsymbol{\xi}_i) \right) \delta^k u_{Nn}(\boldsymbol{\mu}) = -\sum_{q=1}^{Q_g} \sum_{m=1}^{M_q^g} \beta_g^{qm}\left(\boldsymbol{\mu}; {}^k\boldsymbol{u}_N(\boldsymbol{\mu})\right) g^{qm}(\boldsymbol{\xi}_i) \ . \tag{70}$$

The equivalent matrix form is

$$\left(\sum_{q=1}^{Q_j} \sum_{m=1}^{M_q^j} \beta_j^{qm}(\boldsymbol{\mu}; {}^k\boldsymbol{u}_N(\boldsymbol{\mu})) \ J_N^{qm} \right) \left({}^{k+1}u_N(\boldsymbol{\mu}) - {}^k u_N(\boldsymbol{\mu})\right) = -\sum_{q=1}^{Q_g} \sum_{m=1}^{M_q^g} \beta_g^{qm}\left(\boldsymbol{\mu}; {}^k\boldsymbol{u}_N(\boldsymbol{\mu})\right) G_N^{qm} \ . \tag{71}$$

The unknown ${}^k u_N(\boldsymbol{\mu}) \in \mathbb{R}^N$ is defined as $\left({}^k u_N(\boldsymbol{\mu})\right)_n = {}^k u_{Nn}(\boldsymbol{\mu})$. The matrix $J_N^{qm} \in \mathbb{R}^{N \times N}$ and the vector $G_N^{qm} \in \mathbb{R}^N$ are defined as

$$(J_N^{qm})_{i,n} = j_{qm}(\boldsymbol{\xi}_n, \boldsymbol{\xi}_i) \quad \text{and} \quad (G_N^{qm})_i = g_{qm}(\boldsymbol{\xi}_i), \quad 1 \leqslant i, n \leqslant N. \tag{72}$$

The output can be expressed as

$$s_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_\ell} \sum_{m=1}^{M_q^\ell} \beta_\ell^{qm}(\boldsymbol{\mu}) \ \ell^{qm}\left(\boldsymbol{u}_N(\boldsymbol{\mu})\right) \right) \ , \tag{73}$$

or in a vector form

$$s_N(\boldsymbol{\mu}) = \left(\sum_{q=1}^{Q_\ell} \sum_{m=1}^{M_q^\ell} \beta_\ell^{qm}(\boldsymbol{\mu}) \ L_N^{qm\ T} \right) u_N(\boldsymbol{\mu}) \ , \tag{74}$$

where $(L_N^{qm})_i = \ell^{qm}(\boldsymbol{\xi}_i)$. During the offline step we compute basis functions $u_{\mathcal{N}}(\boldsymbol{\mu})$, then form matrices J_N^{qm}, vectors G_N^{qm} and vectors L_N^{qm}. During the online step, for a given parameter $\boldsymbol{\mu}$ and a given solution

${}^k\boldsymbol{u}_N(\boldsymbol{\mu})$ we update the jacobian matrix $J_N\left(\boldsymbol{\mu};{}^k\boldsymbol{u}_N(\boldsymbol{\mu})\right) = \sum_{q=1}^{Q_j}\sum_{m=1}^{M_q^j}\beta_j^{qm}(\boldsymbol{\mu};{}^k\boldsymbol{u}_N(\boldsymbol{\mu}))J_N^{qm}$, and the vector $G_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_g}\sum_{m=1}^{M_q^g}\beta_g^{qm}(\boldsymbol{\mu};{}^k\boldsymbol{u}_N(\boldsymbol{\mu}))G_N^{qm}$, to find ${}^{k+1}\boldsymbol{u}_N(\boldsymbol{\mu})$ such that

$$J_N\left(\boldsymbol{\mu};{}^k\boldsymbol{u}_N(\boldsymbol{\mu})\right)\left({}^{k+1}u_N(\boldsymbol{\mu}) - {}^k u_N(\boldsymbol{\mu})\right) = -G_N\left(\boldsymbol{\mu};{}^k\boldsymbol{u}_N(\boldsymbol{\mu})\right) \ , \tag{75}$$

until convergence. Then, when we have the solution $u_N(\boldsymbol{\mu})$ we assemble the vector

$$L_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_\ell}\sum_{m=1}^{M_q^\ell}\beta_\ell^{qm}(\boldsymbol{\mu})L_N^{qm} \ , \tag{76}$$

and finally we can evaluate the output as

$$s_N(\boldsymbol{\mu}) = L_N^T(\boldsymbol{\mu})\ u_N(\boldsymbol{\mu}) \ . \tag{77}$$

2. Computational framework for reduced basis

We now describe the RB framework used is the `Feel++` RB framework. Its relies on `Feel++` which we describe briefly before turning to the RB layer.

2.1. Feel++ : principles and design

The library `Feel++` provides a clear and easy to use interface to solve complex PDE systems. It aims at bringing the scientific community a tool for the implementation of advanced numerical methods and high performance computing.

`Feel++` relies on a so-called *domain specific embedded language* (DSEL) designed to closely match the Galerkin mathematical framework. In computer science, DS(E)Ls are used to partition complexity and in our case the DSEL splits low level mathematics and computer science on one side leaving the `Feel++` developer to enhance them and high level mathematics as well as physical applications to the other side which are left to the `Feel++` user. This enables using `Feel++` for teaching purposes, solving complex problems with multiple physics and scales or rapid prototyping of new methods, schemes or algorithms.

The DSEL on `Feel++` provides access to powerful, yet with a simple and seamless interface, tools such as interpolation or the clear translation of a wide range of variational formulations into the variational embedded language. Combined with this robust engine, lie also state of the art arbitrary order finite elements — including handling high order geometrical approximations, — high order quadrature formulas and robust nodal configuration sets. The tools at the user's disposal grant the flexibility to implement numerical methods that cover a large combination of choices from meshes, function spaces or quadrature points using the same integrated language and control at each stage of the solution process the numerical approximations.

The code 1 illustrates the clear and easy implementation - building of mesh and function spaces, writing of variational formulation - of a laplacian problem with homogeneous Dirichlet conditions, provided by the `Feel++` library :

Listing 1. Laplacian problem with homogeneous Dirichlet conditions

```
#include <feel/feel.hpp>

int main(int argc, char**argv )
{
  using namespace Feel;
  Environment env( _argc=argc, _argv=argv,
                   _desc=feel_options(),
                   _about=about(_name="laplacian",
```

```
                                _author="Feel++ Consortium",
                                _email="feelpp-devel@feelpp.org"));

    auto mesh = unitSquare();//define the mesh
    auto Vh = Pch<1>( mesh );//function space
    auto u = Vh->element();//element of function space
    auto v = Vh->element();//element of function space

    auto a = form2( _trial=Vh, _test=Vh );//bilinear form
    //a = ∫_Ω ∇u · ∇v
    a = integrate(_range=elements(mesh), _expr=gradt(u)*trans(grad(v)) );
    auto l = form1( _test=Vh );// linear form
    // l = ∫_Ω v
    l = integrate(_range=elements(mesh),_expr=id(v));
    // apply u = 0 on ∂Ω
    a+=on(_range=boundaryfaces(mesh), _rhs=l, _element=u, _expr=constant(0.) );

    //solve the equation
    a.solve(_rhs=l,_solution=u);

    //post processing
    auto e = exporter( _mesh=mesh );
    e->add( "u", u );
    e->save();
}
```

As to build the reduced basis during the offline step of the RBM we need to use the FE discretization, we used a recent development of `Feel++` which allows to operate on parallel computers. In order to create a parallel computing code, we use some strategies of domain decomposition methods with the MPI technology. A feature of our library is that all MPI communications are seamless, thanks to DSEL. Thus, the library `Feel++` provides a parallel data framework : we start with automatic mesh partitioning using `Gmsh` [Geuzaine and Remacle, 2009](Chaco/Metis) — adding information about ghosts cells with communication between neighbor partitions; — then `Feel++` data structures are parallel such as meshes, (elements of) function spaces — parallel degrees of freedom table with local and global views; — and finally `Feel++` uses the library `PETSc` [Balay et al., 2012b, Balay et al., 2012a, Balay et al., 1997] which provides access to a Krylov subspace solvers(KSP) coupled with `PETSc` preconditioners such as Block-Jacobi, ASM, GASM. The last preconditioner is an additive variant of the Schwarz alternating method for the case of many subregion, see [Smith et al., 2004]. A complete description of this HPC part with some blood flow applications is done in the thesis [Chabannes, 2013].

2.2. Feel++ reduced basis framework

The RB framework, depicted in figure 1, provides an interface to the reduced basis methodologies presented so far and automatically generated for the User different instantiations of the reduced basis applications namely command line executable as well as Python and Octave interfaces. These interfaces design follow the simple input-output relationships:

$$[s_1(\mu), s_2(\mu), \ldots, s_O(\mu)] = \ell(\mu) \tag{78}$$

where ℓ is now the simulation software which takes the input parameter set μ and s_i are the outputs of the software which is a set of performance metrics and, if available, the associated error bounds. The User needs to provide the specifications of the model: parameter space, geometry, variational formation with affine decomposition (possibly using EIM).

The design of the `C++` classes RB framework is illustrated by the figure 2. The class `ParameterSpace` generates and stores parameter samplings which are then used by `CRBSCM` and `EIM`. Various sampling strategies

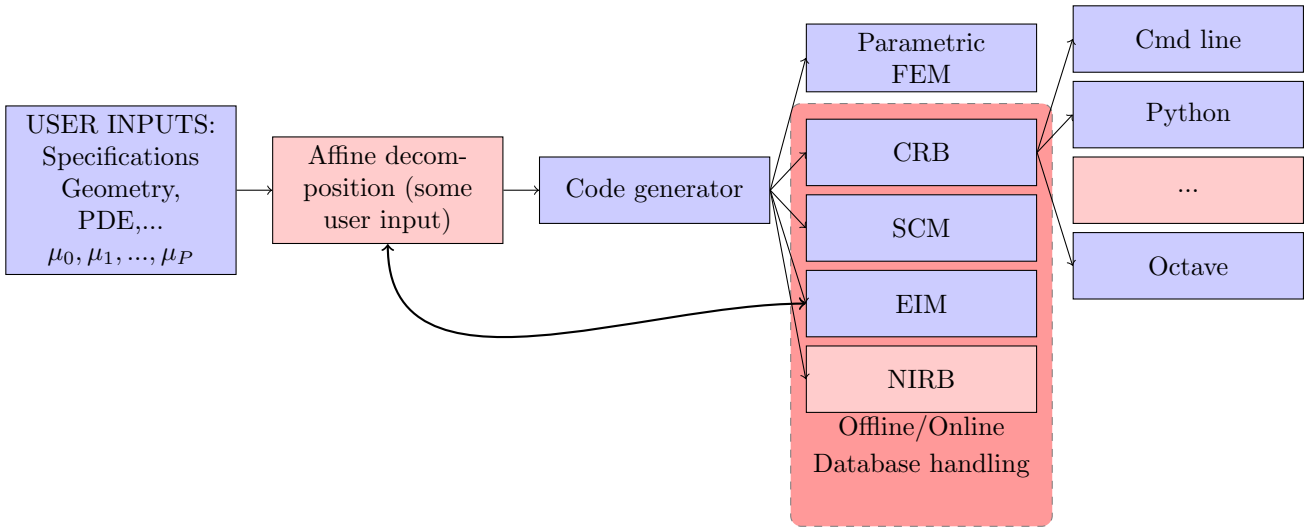


FIGURE 1. RB Framework

are available. `CRBModel` is the interface that the User model must support to use the RB framework and this is through this interface that the User provides the model specifications described previously.

The class `CRB` represents the certified reduced basis implementation for elliptic and parabolic. For elliptic problems by default the Greedy algorithm is used. For parabolic problems, the POD/Greedy algorithm is used ; POD in time and Greedy in space. Note that in both cases, the user can specified in a file the sampling S_N (used to build the space $W_{N_{pr}}$ and $W_{N_{du}}$). S_N can also contains log-equidistributed or log-randomized parameters. The class `CRBDB` handles the database storage using the `Boost.serialization` library. Concerning the online step, dense matrices and vectors are manipulated via the library `Eigen` [Guennebaud et al., 2010]. For linear equations linear solvers from `Eigen` are used. Non-linear solvers from `PETSc` are used to deal with non-linear equations. In order to interact with `PETSc` solvers, `Eigen::Map<>` is the bridge to communicate the non-linear data between `Eigen` and `PETSc`. The class `CRBSCM` implements the SCM algorithm thanks to standard and generalized Eigensolvers from `SLEPc` [Hernandez et al., 2005]. The class `EIM` implements the EIM algorithm. Finally the class `CRBApp` acts the driver for the RB framework.

To build the reduced basis, the offline step of the method can be very expensive. Once this expensive offline step is done, we save all scalar product results from the projection of matrices or vectors on the reduced basis in a database. Hence we can reuse an existing basis to perform online computations. We store the projection of matrices and vectors on the reduced basis, see 11. To estimate the error, we store scalar products defined in (42) and (48). For the SCM, all quantities computed during the offline step mentioned in section 1.2.2 are stored, they refer to equations (28), (26) and (30). Concerning EIM, the interpolation matrix and basis functions are stored, see (52) and (55). To save objects in a database, we use the serialization process provided by the `Boost.serialization` library. The framework is designed so that enriching an existing basis is possible.

A sensitivity analysis can be performed using the scientific library `OpenTURNS` [Works et al., 2012], specialized in the treatment of uncertainties. Through python scripts, the User set a range and a distribution for the inputs from which `OpenTURNS` build a sampling, given to `CRBApp`. Running the online step of the RBM, `CRBApp` supplies the associated set of outputs to `OpenTURNS` which can then compute quantities of interest such as standard deviation, quantiles and Sobol indices.

If the User wants to test the RBM on a model, `CRBApp` will generates a log-randomized or log-equidistributed sampling of parameters on which the output will be evaluated. To verify the RB approximation of the output,

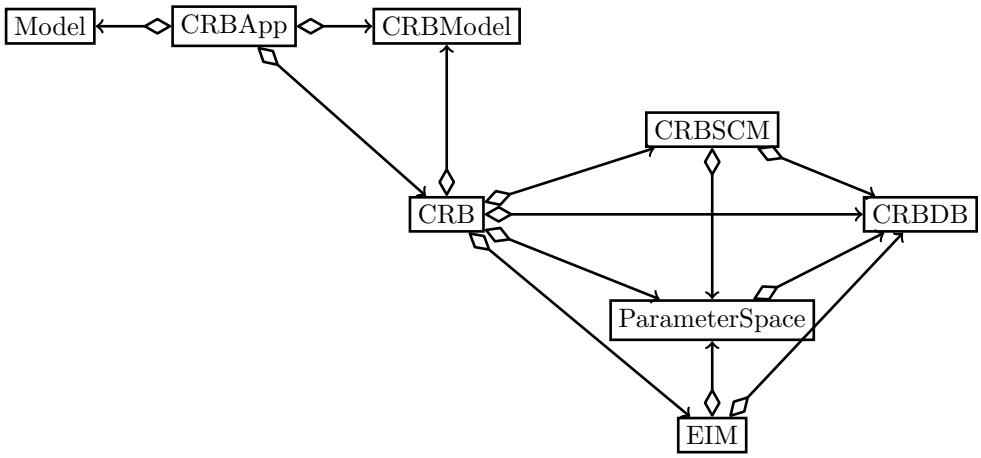


FIGURE 2. Class diagram for the Feel++ RB framework. Arrows represent instantiations of template classes

`CRBApp` provides also the output evaluated by using the finite element discretization. By assuming that, for a given parameter $\boldsymbol{\mu} \in \mathcal{D}$, the solution field $u_{\mathcal{N}}(\boldsymbol{\mu})$ obtained using the finite element discretization is the true solution, the error on the solution filed is then computed in L_2 and H_1 norm. In other words the class provides $e_{L_2} = \frac{||u_{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})||_{L_2(\Omega)}}{||u_{\mathcal{N}}(\boldsymbol{\mu})||_{L_2(\Omega)}}$ and $e_{H_1} = \frac{||u_{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})||_{H_1(\Omega)}}{||u_{\mathcal{N}}(\boldsymbol{\mu})||_{H_1(\Omega)}}$. It also computes the error on the output : $e_{output} = \frac{|s_{\mathcal{N}}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})|}{|s_{\mathcal{N}}(\boldsymbol{\mu})|}$.

2.2.1. *Parallel strategy*

The RB framework supports parallel architectures by relying on the `Feel++` parallel data structures. All data associated to the reduced basis (scalars, dense vectors and matrices, parameter space samplings) are actually duplicated on each processor. However note that since the mesh is partitioned according the number of processors, finite element approximations and thus the reduced basis functions are in fact spread on all processors. Currently the basis functions are saved in the RB database with their associated partitioning. If they are required for visualization purposes or reduced basis space enrichment, the same data partition as in the initial computations must be used. Another particular attention must be paid to parameter space sampling generation: we must ensure that all processors hold the same samplings. To this end, they are generated in a sequential way by only one processor and then broadcasted to other processors. Finally, in the implementation of the EIM algorithm, we need to compute the arg max over a fine sampling of infinity norm of a quantity that lives on the mesh, see equation (50). Each processor computes its local contribution and the a "max" MPI reduction is done.

2.2.2. *Practical use of EIM*

We consider now a practical example of the usage EIM taken from the non-linear electro-thermal problem described in section 3.1. The associated weak formulation can be expressed as :

$$\text{find } \boldsymbol{U} = (V, T) \in \boldsymbol{X}_h = X_h^V \times X_h^T \; s.t. \;\; \forall \boldsymbol{V} \in \boldsymbol{X}_h \text{ and } \forall \boldsymbol{\mu} \in \mathcal{D} : \qquad g(\boldsymbol{U}, \boldsymbol{V}; \boldsymbol{\mu}) = 0 \; . \tag{79}$$

where V and T are respectively the electrical potential and the temperature, and $\boldsymbol{X}_h$ is a composite space (product of the function spaces X_h^V and X_h^T to which V and T belong respectively). In this model, g depend on a term $\sigma(T)$, which is non-affine in parameter $\boldsymbol{\mu}$:

$$\sigma(T) = \frac{\boldsymbol{\mu}_0}{1 + \boldsymbol{\mu}_1 (T - T_0)} \; , \tag{80}$$

where T_0 is a set constant.

As a consequence, g is non-affine in parameter $\boldsymbol{\mu}$ and the associated affine decomposition can not be obtained without an EIM decomposition of the term $\sigma(T)$. To recover an affine decomposition of (80), the User has to use `EIM` class by instantiating an `EIM` object as shown in code 2. This `EIM` object gives a direct access to the basis functions q_M and the coefficients β_M of the EIM expansion (described in 1.3).

Listing 2. EIM approximation of a non-affine parameter and solution dependent term

```
parameterspace_ptrtype Dmu; //𝒟
auto mu_min = Dmu->element();
auto mu_max = Dmu->element();
//fill mu_min and mu_max to define "corners" of 𝒟
mu_min <<  ... ; mu_max <<  ... ;
//associate "corners" to 𝒟
Dmu->setMin( mu_min ); Dmu->setMax( mu_max );

parameter_type mu; // μ ∈ 𝒟

auto Pset = Dmu->sampling();
int eim_sampling_size = 1 000;
Pset->randomize(eim_sampling_size);

//nonlinear expression σ(T)
auto sigma = ref( mu(0) )/( 1+ref( mu(1) )*( idv(T)-T0 ) );

//eim object
auto eim_sigma = eim( _model=solve( g(U,V;μ)=0 ),
                      _element=T, // unkown needed for the evaluation of σ(T)
                      _parameter=mu,// μ
                      _expr=sigma,  // σ(T)
                      _space=X_h^T, // eim basis function space see (79)
                      _name="eim_sigma",
                      _sampling=Pset );

//then we can have access to β coefficients
std::vector<double> beta_sigma = eim_sigma->beta( mu );
```

Note that in Listing 2, for the construction of the object `eim_sigma` we use the abuse of notation `_model = solve(` $g(\boldsymbol{U},\boldsymbol{V};\boldsymbol{\mu})=0$ `)` to indicate that the problem (79) is solved. In practice the user provides as argument a functor providing an `operator()` that can be called by the EIM offline algorithm to execute the underlying finite element problem for a given parameter μ and retrieve the associated finite element solution. As to the function space to which the EIM basis belong it is given by the `_space` argument, we chose to use X_h^T for the function space of eim expansions. An example the object type is given in Listing 1.

Now, let us see how to have an expansion of a function which does not depend on the solution of the model. Say we want the expansion of $g(\boldsymbol{\mu};x) = \sin(\boldsymbol{\mu}_0\ \pi\ x)$ where $\boldsymbol{\mu}_0$ is a parameter of the model. Here there is no need to compute the solution of the model to have the expansion of $g(\boldsymbol{\mu};x)$. So we use the keyword `eim_no_solve`, like illustrated in code 3.

Listing 3. EIM approximation with no dependence on solution

```
//eim expansion of sin(μ_0 π x)
auto eim_sin = eim( _model=eim_no_solve(FemModel) ),
                    //our model called FemModel is passed as argument of eim_no_solve
                    _element=T,
                    _parameter=mu,// μ
```

```
_expr=sin( ref( mu(0) )*pi*Px() ),
_space=X_h^{FemModel},//object representing the function space
                      //used in the model FemModel
_name="eim_sin",
_sampling=Pset );
```

3. Numerical experiments

The Laboratoire National des Champs Magnétiques Intenses (LNCMI) is a French large scale facility ([Debray et al., 2002]) enabling researchers to perform experiments in the highest possible magnetic field (up to 35 T static field). High magnetic fields are obtained by using water cooled resistive magnets (cf. figure 3 and 4) connected with a 24 MW power supply (see [F.Debray et al., 2012] and references included for details).

The design and optimization of these magnets require from an engineering point of view the prediction of "quantities of interest," or performance metrics, which we shall denote *outputs* — namely magnetic field in the center, maximum stresses, maximum and average temperatures. These outputs are expressed as functionals of field variables associated with a set of coupled parametrized PDEs which describe the physical behavior of our magnets. The parameters, which we shall denote *inputs*, represent characterization variables — such as physical properties, heat transfer coefficients, water temperature and flowrate, and geometric variables in optimisation studies. To evaluate these implicit *input–output* relationships, solutions of a multi-physics model involving electro-thermal, magnetostatics, electro-thermal-mechanical and thermo-hydraulics are requested. It should be noted that this model is non-linear as the material properties depend on temperature. In practice these evaluations represent a huge computational time but they are mandatory to improve the magnet design as we can no longer rely on common physical sense.

To significantly reduce this computational time, we choose to use model order reduction strategies, and specifically to use the reduced basis method presented in the previous sections which is well adapted to tackle this question. First, we focused on the electro-thermal behaviour of the resistive magnets. We present now the non-linear electro-thermal coupled problem developed for this purpose. Design issues and challenges attached to this application are then explicited. Finally the Feel++ reduced basis framework is applied on two examples directly connected with actual LNCMI developments.

3.1. A non-linear electro-thermal model for High Fields Magnets

Two main technologies are developed at the LNCMI for the resistive magnets, namely the Bitter and Polyhelix techniques. A typical 24 MW magnet will consist in a set of Polyhelices and Bitter magnets (or inserts in LNCMI standard terminology) powered separatly by a 12 MW power supply. Polyhelices insert are in turns made of concentric copper alloy tubes, in which a helix have been cut by spark erosion techniques, electricaly connected in series. The helix cut in each tube may be filled with some glue or let free. In this case some insulator are introduced periodicaly to prevent electrical contact between turns.

FIGURE 3. A Polyhelix insert. The helices are actually copper alloy tubes cut by a spark erosion technique. Epoxy glue may be introduced into the slit to ensure the electrical insulation between turns : this is the so-called "Longitudinally cooled helices" as the water flow is "longitudinal". An another possibility to ensure electrical insulation is to introduce periodically some insulators into the slit : this is the so-called "Radially cooled helices" as the water may flow from the inner to the outer radius througth the open slit.

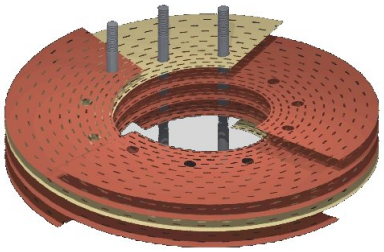


FIGURE 4. A Bitter insert consists in a stack of copper alloy plates. Insulators are introduced in between plates to create a helicoïdal current path. Small holes are drilled to enable the water flow throught the magnet. Some tie-rods are inserted to ensure a good electrical contact between each plate by applying a precompression.

The applied electrical current is about 30 kA in each insert. This leads to important Joules losses within the insert. A water flow (about 150 l/s) is then necessary to cool down the insert and to avoid that the temperature reaches some given threshold.

The temperature T in the magnets is given by the following electro-thermal coupled model, where V stands for the electrical potential :

$$\begin{cases} -\nabla \cdot (\sigma(T)\,\nabla V) = 0, \\ -\nabla \cdot (k(T)\,\nabla T) = \sigma(T)\,\nabla V \cdot \nabla V. \end{cases} \tag{81}$$

$\sigma(T)$ and $k(T)$ (respectively the electrical and thermal conductivity of the material) depend on T, hence the non linearity of this model :

$$\sigma(T) = \frac{\sigma_0}{1+\alpha(T-T_0)} \qquad \text{and} \qquad k(T) = \sigma(T)LT \tag{82}$$

where σ_0 denotes the electrical conductivity at $T = T_0$. α and L are characteristics of material, namely the ratio resistivity-temperature and the Lorentz number.

In a polyhelix magnet (see figure 3), copper solenoids are electrically connected in series. The circulation of electrical current in each of them can be modeled as a difference of electrical potential between the current input and the current output. The value of this difference can be evaluated from the current intensity - known from experiments -, and is handled in the model using Dirichlet boundary conditions. On all other boundary

Neumann conditions are applied, as no current flows out the magnets apart from the top and bottom regions :

$$\begin{cases} V = 0 \text{ on current input } (c_{in}) \text{ and } V = V_D \text{ on current output } (c_{out}) \\ -\sigma(T)\,\nabla V \cdot \mathbf{n} = 0 \text{ on other boundaries.} \end{cases}$$

We can assume that there is no thermal exchange on the "connection" surfaces (current input and output), nor on the interfaces with insulators. The thermal exchange between copper and cooling water is modeled by a forced convection conditions (with h the heat transfer coefficient, and T_w the water temperature) :

$$\begin{cases} -k(T)\,\nabla T \cdot \mathbf{n} = h(T - T_w) \text{ on cooled regions } (\textit{cooling}) \\ -k(T)\,\nabla T \cdot \mathbf{n} = 0 \text{ on other boundaries.} \end{cases}$$

h is obtained by mean of classical thermohydraulics correlations given the water flowrate.

3.2. **Design issues and Challenges**

In the international race for higher magnetic field we need to push our magnet technologies to their limits. The actual design of our inserts are mainly limited by thermal constraints for the inner parts and by mechanical constraints for the outer parts. Two routes are actively investigated to go beyond these limits : one consists in developing the "Radially cooled helices" (see comment on fig 3) which are less sensitive to thermal limitations from their design principles; the other consists in looking for materials with improved mechanical properties. This last aspect is a more long term research.

LNCMI current projects involves the use of those "Radially cooled helices" to achieve higher magnetic field by breaking the thermal limits. From a designer point of view it requires to control the temperature, - average and standard deviation - in the magnets considering several parameters. Those corresponding to operating conditions (ie. applied potential V_D and cooling conditions T_w, h), and those allowing to take into account uncertainties on materials properties (ie. electrical conductivity σ_0, and coefficients α and L). Indeed, materials properties are not precisely known - the copper producer only gives upper and lower bounds for σ_0, and α and L are only known from litterature to be in a given interval -.

To reach this goal we need fast, accurate and reliable estimates for the temperature field. This need may be adressed by controlling a posteriori error for T associated with an anisotropic mesh adaptation strategy ([Prud'Homme et al., 2012b]). This approach is however not sufficient nor efficient in terms of required computational ressources. An efficent alternative is to use the Reduced basis method (see 1.4.2) to greatly reduce the computional cost. In our case this method has to be combined with the Empirical Interpolation Method (see 1.3) to deal with non-affine dependence parameters, arising from the righthand side of (81) and (82). This new approach implemented using Feel++ reduced basis framework makes it possible to carry out parametric studies and sensitivity analysis at reasonnable cost which are mandatory to improve the design. In the sequel we will consider the input parameter $\boldsymbol{\mu} \in \mathbb{R}^6$ defined by :

$$\boldsymbol{\mu} = (\sigma_0, \alpha, L, V_D, h, T_w) \tag{83}$$

A first example of a parametric study on a small geometry is presented in 3.6. It also serves as a validation of the framework. A second example is given in 3.7 for a larger problem, illustrating the parallel computing possibilities of Feel++ RB framework. This second example also shows the computation of quantities of interest for sensitivity analysis -as a first step toward uncertainties quantification- adressed to applications such that design under uncertainties.

3.3. **Reduced basis approximation of the problem**

As the material properties makes our model non-affinely parametrized and non-linear, the use of reduced basis method in such a context involves to proceed as described in section 1.4. The output (equ. (58)) we focus

on in the next sections is the mean of temperature $T(\boldsymbol{\mu})$:

$$s(T(\boldsymbol{\mu})) = \frac{1}{|\Omega|}\int_{\Omega} T(\boldsymbol{\mu}) d\Omega \tag{84}$$

To handle seamlessly the Dirichlet conditions in the reduced basis framework, we have chosen to treat them weakly using the Nitsche formulation.

The solutions $V(\boldsymbol{\mu})$ and $T(\boldsymbol{\mu})$ of our coupled problem (c.f. section 3.1) are respectively zeros of the two functions $g_V(V(\boldsymbol{\mu}), \phi_V; \boldsymbol{\mu})$ and $g_T(T(\boldsymbol{\mu}), \phi_T; \boldsymbol{\mu})$ expressed as follows :

$$\begin{aligned} g_V(V(\boldsymbol{\mu}), \phi_V; \boldsymbol{\mu}) = &\int_{\Omega} \sigma(T)\nabla V \cdot \nabla \phi_V - \int_{c_{in}\cup c_{out}} \sigma(T)(\nabla V \cdot \mathbf{n})\phi_V + \int_{c_{in}\cup c_{out}} \frac{\sigma(T)\gamma}{h_s} V\phi_V \\ &- \int_{c_{in}\cup c_{out}} \sigma(T)(\nabla \phi_V \cdot \mathbf{n})V - \int_{c_{out}} \frac{\sigma(T)\gamma}{h_s} V_D\phi_V + \int_{c_{out}} \sigma(T)(\nabla \phi_V \cdot \mathbf{n})V_D \end{aligned} \tag{85}$$

$$g_T(T(\boldsymbol{\mu}), \phi_T; \boldsymbol{\mu}) = \int_{\Omega} k(T)\nabla T \cdot \nabla \phi_T + \int_{cooling} hT\phi_T - \int_{\Omega} \sigma(T)\phi_T \nabla V \cdot \nabla V - \int_{cooling} hT_w\phi_T \tag{86}$$

where γ and h_s denote respectively the penalization coefficient - set to a constant regarded as independent of μ - and the characteristic mesh size. Note that we have multiplied by $\sigma(T)$, the term $\frac{\gamma}{h_s}$ to ensure that the penalty term scales properly with respect to the other terms on the same boundary. Thanks to this, γ must be sufficiently large ($\gamma \approx 30$ say) to ensure coercivity and it does not depend on the parameters.

We can now use a Newton algorithm to solve $g(< V(\boldsymbol{\mu}), T(\boldsymbol{\mu}) >, < \phi_V, \phi_T >; \boldsymbol{\mu}) = g_V(V(\boldsymbol{\mu}), \phi_V; \boldsymbol{\mu}) + g_T(T(\boldsymbol{\mu}), \phi_T; \boldsymbol{\mu}) = 0$ (c.f. section 1.4.1, algorithm 2) where the residual and jacobian read:

$$r(< V(\boldsymbol{\mu}), T(\boldsymbol{\mu}) >, < \phi_V, \phi_T >; \boldsymbol{\mu}) = g_V(V(\boldsymbol{\mu}), \phi_V; \boldsymbol{\mu}) + g_T(T(\boldsymbol{\mu}), \phi_T; \boldsymbol{\mu}) \tag{87}$$

$$j(< V(\boldsymbol{\mu}), T(\boldsymbol{\mu}) >, < \phi_V, \phi_T >; \boldsymbol{\mu}) = \begin{pmatrix} \dfrac{\partial g_V\,(V(\boldsymbol{\mu}), \phi_V; \boldsymbol{\mu})}{\partial V} & \dfrac{\partial g_V\,(V(\boldsymbol{\mu}), \phi_V; \boldsymbol{\mu})}{\partial T} \\ \dfrac{\partial g_T\,(T(\boldsymbol{\mu}), \phi_T; \boldsymbol{\mu})}{\partial V} & \dfrac{\partial g_T\,(T(\boldsymbol{\mu}), \phi_T; \boldsymbol{\mu})}{\partial T} \end{pmatrix} \tag{88}$$

The terms σ and k (defined in (82)) which appears in g_V and g_T present a non-affine dependence in parameters. Consequently, we need to use of the Empirical Interpolation Method (c.f. section 1.3) to build the affinely parametrized approximations σ_{M_σ} and k_{M_k}

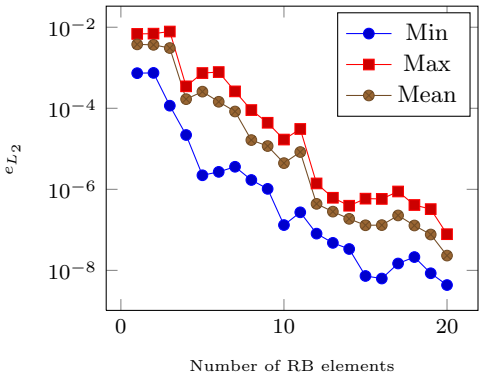
$$\sigma_{M_\sigma}(T(\boldsymbol{\mu}), x; \boldsymbol{\mu}) = \sum_{m=1}^{M_\sigma} \beta_m^\sigma(T(\boldsymbol{\mu}); \boldsymbol{\mu}) q_m^\sigma(x) \quad \text{and} \quad k_{M_k}(T(\boldsymbol{\mu}), x; \boldsymbol{\mu}) = \sum_{m=1}^{M_k} \beta_m^k(T(\boldsymbol{\mu}); \boldsymbol{\mu}) q_m^k(x) \tag{89}$$

3.4. **Convergence of the method**

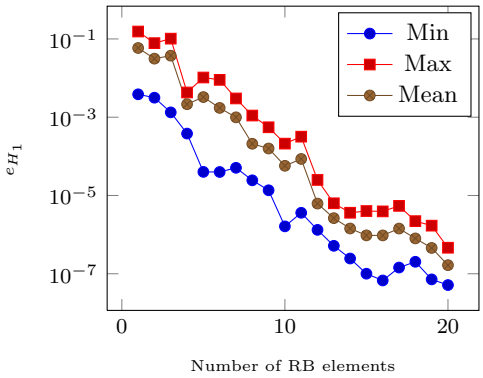
The use of the RB approximation for the following test cases requires to ensure the convergence of the method. The following study has been undertaken on a geometry corresponding to a sector of an helix magnet (see figure 11). Let e_{L_2} and e_{H_1} be the relative errors between the RB aprroximation (T_{RB}) of temperature and the PFEM one (T_{PFEM}) – PFEM is a FE resolution using the affine decomposition – , defined as

$$e_{L_2} = \frac{\| T_{RB} - T_{PFEM} \|_{L_2}}{\| T_{PFEM} \|_{L_2}} \quad \text{and} \quad e_{H_1} = \frac{\| T_{RB} - T_{PFEM} \|_{H_1}}{\| T_{PFEM} \|_{H_1}}. \tag{90}$$

Let Ξ_{test} a set of N_{test} randomly chosen parameters such that $\Xi_{test} \cap S_N = \emptyset$. Graphs 5(a) and 5(b) show the convergence of the RB approximation, by plotting e_{L_2} and e_{H_1} as functions of N, for $N_{test} = 20$. We recall here that we have $S_{N-1} \subset S_N$.



(a) Convergence of RB method : L_2 error statistics



(b) Convergence of RB method : H_1 error statistics

FIGURE 5. Convergence of the RB method.

From these graphs, we assess that the convergence of our reduced basis model is ensured.

However we note that despite the use of nested approximation spaces, the convergence is not monotone. Remember that here we deal with a multi-physics non-linear problem. Picard algorithm is used to treat the non-linearity. In order to limit number of iterations, a maximum number of iterations max_{it} is fixed.

For a given parameter, the Picard algorithm can stops owing to the fact that it reaches max_{it} iterations. In this case, the error criterion is not satisfied, and the solution can then be worst in terms of precision. Furthermore, there is currently no theory – to our knowledge – for the a priori convergence of the RB method applied on a non-linear model. These results require certainly a closer look, however we note that the RB approximation is converging very rapidly.

Remark 2. *For all non-linear problems exposed in this paper, no a posteriori error estimation is used to select parameters during the offline step of the algorithm. Parameters are log-randomly selected from the parameter space. A posteriori error estimation is an ongoing work in the* `Feel++` *RB framework.*

3.5. A posteriori error estimation on linear models

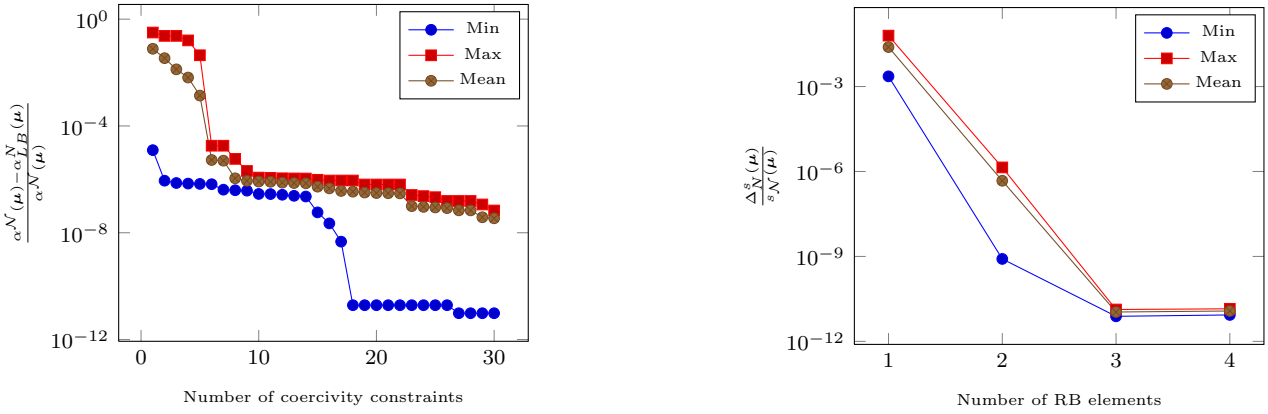
The RB framework developed in Feel++ handles a posteriori error estimation, we illustrate here the use of these estimators on 2D and 3D linear simplified models. Regarding non-linear models, work remains to develop a posteriori error bounds.

The simplified model uses a 2D geometry representing a quarter of a Bitter plate (see figure 4). For simplification purpose, the cooling holes are not considered in this case. Inner and outer radius are defined as water cooled regions. The current circulation in the quarter of plate is modeled by a difference of potential between the two remaining edges. The figure 6(a) shows the convergence of the SCM algorithm to determine the lower bound of the coercivity constant $\alpha^{\mathcal{N}}(\boldsymbol{\mu})$ of the bilinear form associated to linear version of the heat equation introduced in (81). The linear version of (81) consists in considering the electrical and thermal conductivity of the material as independent on temperature and constants within the domain. The electrical conductivity is equal to the component σ_0 of $\boldsymbol{\mu}$ and the thermal conductivity is deduced from the inputs σ_0 and L such that $k_0 = \sigma_0\ L\ T_0$ (T_0 is a reference temperature which is not treated as a parameter). As σ_0 and k_0 are independent on T, the temperature coefficient α has no influence in this case (see (82)). Therefore the input parameter $\boldsymbol{\mu}$

can be simplified and be defined as

$$\boldsymbol{\mu} = (\sigma_0, L, V_D, h, T_w) \ . \tag{91}$$

The RB and SCM algorithms are both applied to a set of 1000 parameters randomly chosen. In the graph 6(a) we focus on the relative error to approach the coercivity constant depending on the number of constraints. The graph 6(b) shows the normalized a posteriori error estimation depending on the number of RB elements.



(a) Convergence of the SCM algorithm

(b) Convergence of a posteriori error estimation

FIGURE 6. Convergence of the SCM algorithm and a posteriori error estimation (2D linear model).

These results show that just few reduced basis functions are sufficient to obtain a very good approximation of the FEM solution.
First, we shall note that the problem we are dealing with is not so rich, since we are in a linear case without geometrical input parameters. The RB approximation space is then of very low dimension. Furthermore, we note that the parametric dimension is lower than expected (see (91)). Indeed, after a dimensional analysis, only three parameters arise : the Biot number, an adimensionalized water temperature and some adimensionalized Joule effect parameter which involves V_D.

The previously introduced linear version of the electro-thermal model is here applied to a Bitter magnet (see figure 8). The error estimation $\Delta^s_N(\boldsymbol{\mu})$ is plotted in function of the number of RB elements on the figure 7. For now, we encounter stability problems with the SCM algorithm on this 3D geometry, – no convergence of generalized eigenvalue solvers – so $\alpha^{\mathcal{N}}_{LB}(\boldsymbol{\mu})$ has been set to 1 in the formula (35). For this study the RB algorithm was applied to a set of 500 parameters randomly chosen.

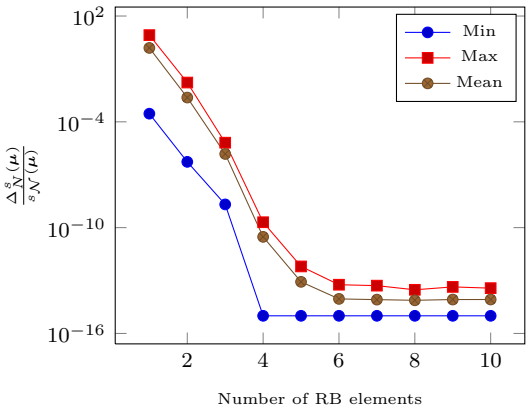


FIGURE 7. Convergence of a posteriori error estimation (3D linear model).

Again, only few basis functions are needed to obtain a sufficiently accurate approximation, due to the previously introduced reasons.

3.6. **Parametric study - example for a Bitter magnet**

The LNCMI is currently involved in the Hybrid Magnet Project, which consists in the assembly of a resistive inner coil with a superconducting outer one. This technology allows to generate the highest continuous magnetic field for a given electrical power installation (43T for 24MW at LNCMI). The inner coil is a combination of two magnets technology (Bitter and polyhelix), powered independently (see [F.Debray et al., 2012] and [A.Bourquard et al., 2010]).

To reach the target magnetic field, the Bitter magnet has to be redesigned to generate more than 9 Tesla which corresponds to the actual design. This may be achieved by increasing locally the current density $j = \sigma(T)\nabla V$. In practice it means modifying the stacking of copper plates (see comment on fig 4). This change will also lead to locally increase the temperature.

To determine how much we can possibly increase j while keeping a reasonable mean temperature not to damage the materials we have carried out a parametric study. This study consists in setting all input parameters except V_D, since j is proportional to V_D.

From an engineering point of view, for this kind of magnets - made of massive "solenoids" and not with stranded ones - j can be approximated by :

$$j(r) = \sigma_0 \left(\frac{V_D}{\theta\, r} \right) \tag{92}$$

where r is the radius - cylindrical coordinates - and θ is the angle of the sector.

The input parameter V_D can then be chosen such that the current density $j(r_{int})$ on inner radius r_{int} varies from 30.10^6 to 100.10^6 $A.m^{-2}$.

The study was performed with classical FE method by running calculations for a given set of parameters and with RB method in order to validate the framework.

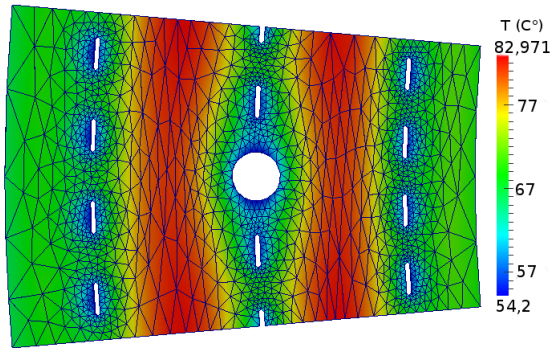


FIGURE 8. RB simulation on a sector of a Bitter magnet

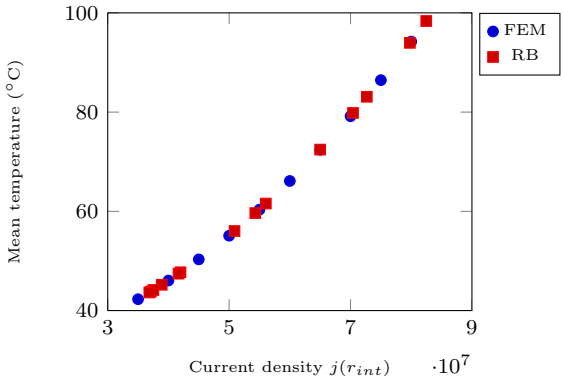


FIGURE 9. Parametric study on Bitter: Mean temperature VS current density

The simulation illustrated above (figure 8) have been performed on 8 processors (domain divided into 8 partitions), using the solver GMRES and the Additive Schwarz Method (GASM) as preconditioner. In a computational time point of view, the use of reduced basis framework for this computation represents a gain of 2 or 3 orders of magnitude in comparison with the classical FE model.

In the current design the mean temperature is about $40°C$. Allowing this value to reach $60°C$, we can increase the current density by a factor 1.5 as can be seen on fig 9. This allow us to redefine the stacking of the copper plates to safely reach 10 tesla. The new Bitter magnet design will be based on this result.

3.7. **Towards uncertainty quantification - example for a helix magnet**

As stated in introduction to this section, using radially cooled helices for the inner most helices of a polyhelix magnet (see [F.Debray et al., 2012]) allows to reach higher current density and hence to reach higher magnetic fields. This is possible as the radially cooled helices are by concept less thermally limited than longitudinally ones (see comments on fig 3). The water flows from the inner radius to the outer radius of the helix is indeed more efficient to cool it down.

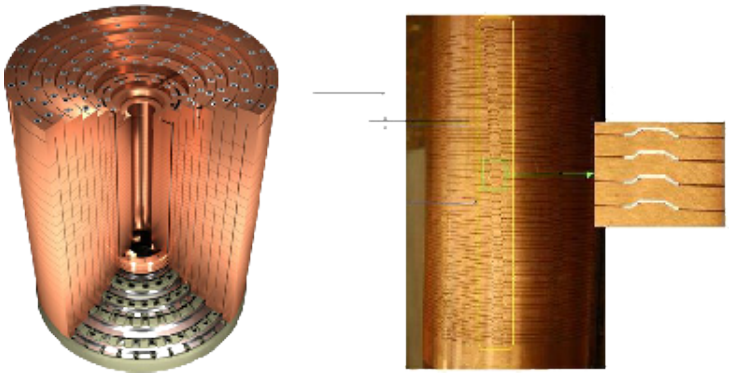


FIGURE 10. On the left : an example of a polyhelix insert; on the right a detailled view of the inner radially cooled helix with a zoom on the cooling channels and insulators.

However since insulators have to be introduced in between each turn (see fig 10) some hotspots are expected in these regions. Accurate estimates of the temperature are a key element for the design. In particular we want the mean temperature in the helix to remain below some threshold. This allows to limit the maximum temperature reached in the insulators and thus to keep them safe. To do so we must take into account the uncertainties on the copper alloy properties (σ_0, α and L), and the uncertainties on the operating parameters V_D and cooling conditions T_w, h. This can be done by performing a sensitivity analysis. The ranges of input parameters indicated in the table below are chosen from literature and experimental results.

Parameter	Range
σ_0 [10^6 Ω^{-1} m^{-1}]	$[50; 50, 2]$
α [10^{-3}]	$[3, 3; 3, 5]$
L [10^{-8}]	$[2, 5; 2, 9]$
V_D [V]	$[0, 14; 0, 15]$
h [W K^{-1} m^{-2}]	$[70000; 90000]$
T_w [K]	$[293; 313]$

TABLE 1. Input parameters ranges

Sensitivity analysis is managed by linking the Feel++ reduced basis framework with the library `Openturns` (see [Dutfoy et al., 2009] and `http://www.openturns.org`), dedicated to the treatment of uncertainties. To perform this study, `OpenTurns` build a sample of input parameters, determined upon a given probability distribution in the wanted ranges. The RB framework is then used to compute the associated set of outputs, from which the mean and the standard variation are deduced by `OpenTurns`.

Parallel computing possibilities available within CRB framework allows to apply model order reduction on such a complex geometry (see the left part of fig 11) with 1.5 10^6 dofs.

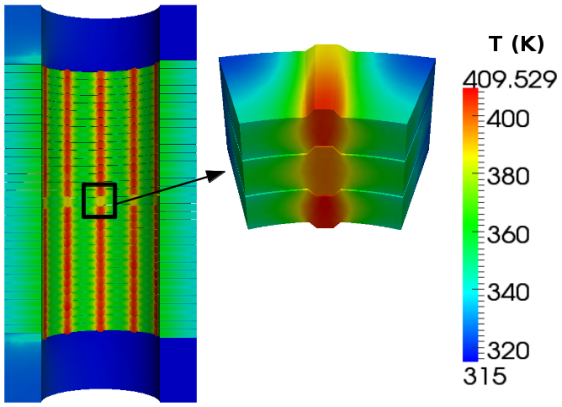


FIGURE 11. Temperature field computed with reduced basis method on a radial helix and on a sector, for inputs $[\sigma_0 : 5.01e+7; \alpha : 3.48e-3; L : 2.89e-8; h : 8.15e+4; T_w : 295.2]$. The value of parameter V_D corresponds to a given current of 25 kA.

Regarding computational time, each run of the classical FE model on helix geometry - 20 iterations of Picard algorithm, run on 15 processors - takes around 45 minutes. The reduced basis model used here consists in 20 basis for EIM approximation, and 10 for the reduced basis one. This gives a mean L_2 relative error between FE and RB model about 10^{-4} on a set of parameters. The online step of this RB model in the same configuration as FE one represents a computational time of 18 seconds - average on 20 model evaluations -. The use of

`Feel++` reduced basis framework leads consequently to an increase by a factor 150. Considering duration of both offline and online steps, the building of the EIM basis takes around 2 hours and a half, and each RB basis needs about 10 minutes. We assess that currently the use of reduced basis provides a gain in time compared with the FE model over 6 evaluations. However, since the RB offline step needs as many FE resolutions as the number of reduced basis, this gain in time should appear after at least 10 evaluations. This is indeed due to the time needed by the assembly process - which is not negligible - and we shall note that unlike for the FE model for which matrices assembly is made for each run - since no precomputations are feasible -, the use of affine decomposition in the RB model allows to do it at once during the offline step, whatever the number of reduced basis. Anyway, to ensure that it is indeed the assembly step, we make the same calculation, comparing the RB model with the parametric FE one - FE with affine decomposition, provided by EIM - for which the assembly is done once. As expected, this study confirms the hypothesis since the gain provided in this case is reached only over 13 evaluations.

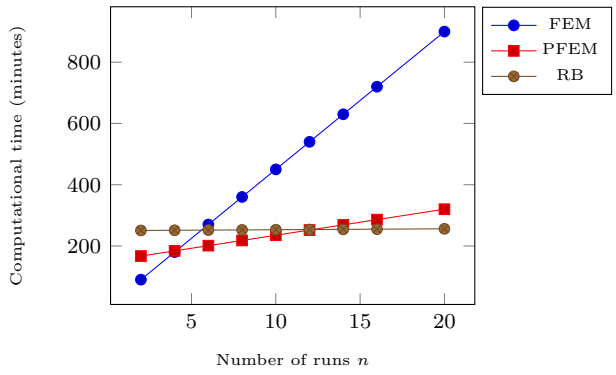


FIGURE 12. Computational time comparison

The graph (12) illustrates the observed times for several number of runs n. FEM time simply corresponds to the time needed by each evaluation (taking into account assembly step) of FE model multiplied by the number of runs (45 minutes $\times n$) . The PFEM model uses EIM to build an affine decomposition, which allows to make assembly process step once. The time required by PFEM is the sum of EIM time, and the time of the runs (8.5 minutes $\times n$). Finally, the time needed by the RB method is the sum of the offline construction time (EIM time and reduced basis elements building time) and the time for evaluations (0.3 minutes $\times n$).

Nevertheless, the number of runs required for a sensitivity analysis with `OpenTurns` is so large that we cannot perform it with the compution nal ressources available. For this reason we choose to select a "representative" helix sector, since the temperature behaviour is regular and exhibits some symmetry (see right part of fig 11).

The following results - on the helix sector - have been obtained from a uniform distribution of the inputs in the ranges defined in table 1, with a sample size of 300.

This experiment has been undertaken on different samples, of increasingly large size. We noticed that from a sample size around 300, the difference oberved on the computed quantities induced by the increasing of the sample size is no more significant. That is the reason why we consider that such a sample is sufficient to obtain satisfactory results in terms of precision.

The two following quantities are essential since they give a "reference" value for the mean temperature in the given ranges, associated to the standard deviation which can be considered as a "safety interval" around the previous reference value.

Mean of outputs [K]	368.66
Standard deviation [K]	6.22

TABLE 2. Mean and standard deviation

The values obtained for mean of outputs and standard deviation ensure that for the given input parameters ranges (table 1), the mean temperature in the magnet will be included in $[362,44; 374,88]$ K. The largest obtained value -the most critical- can then be "refined" using the quantiles. Quantiles correspond to the threshold $q(\gamma)$ we are sure not to exceed with a given probability γ. Considering Y the set of outputs obtained from the

sampling, this reads as:

$$\text{Find } q(\gamma) \text{ such that } P(Y < q(\gamma)) > \gamma \tag{93}$$

For the same configuration as for the mean and standard deviation (table 2), quantiles with respectively $\gamma = 80\%$ and $\gamma = 99\%$ have been computed :

Probability (γ)	Quantile [K]
80%	371.297
99%	376.014

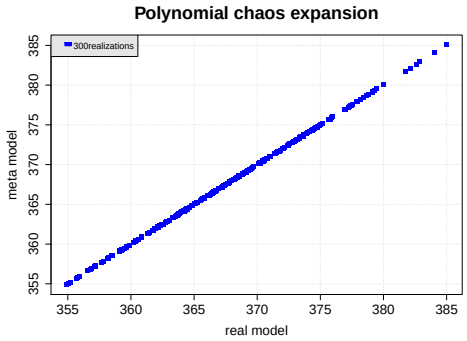
TABLE 3. Quantiles

These results (table 3) are also of great interest for our magnet control system in order to early detect unexpected thermal behaviours and thus to better predict incidents.

Furthermore, in a goal of model simplification and to guide easily the experimental measures campaign, we need to identify which parameters are the most influent on the temperature behaviour. This can be managed by the computation of sensitivity indices. The following Sobol indices of order 1 (see table 4) have been obtained using a chaos polynomial meta-model on the previous sample of input parameters. The graph below shows the comparison between outputs obtained with RB model and meta-model to ensure that this last one fits well with base model.

Parameter (γ)	Sobol indice
σ_0	$6,85.10^{-5}$
α	$4,5.10^{-4}$
L	$9,2.10^{-3}$
V_D	$0,12$
h	$0,24$
T_w	$0,62$

TABLE 4. Sobol indices



These results shows that copper alloys properties are greatly less influent than operating parameters, especially cooling conditions. Consequently, we have to focus on these quantities both in terms of quality of experimental measures, and cooling water behaviour understanding. Such an investment would represent a real improvement for the model, bringing even more reliability.

4. CONCLUSIONS AND PERSPECTIVES

`Feel++` Reduced Basis framework has been successfully applied to industrial class problems as illustrated in the examples above. It shows its ability to deal with a 3D elliptic non-affinely parametrized and non-linear problem. From a practical point of view, solving such non-linear coupled problems raise the challenge of management of large databases. The framework currently provides a posteriori error estimators for elliptic linar problems (see section 1.2). Their extension for the certification of the RB approximation in the case of non-affinely parametrized and non-linear problems (see for example [Cuong, 2005]) needs to be implemented. With adequate a posteriori error estimators, a greedy algorithm can then be applied to construct an optimal reduced basis for the electro-thermal coupled problem introduced in section 3.1.

More mid-terms developments involves the implementation of automatic differentiation to greatly help to account for the geometric input parameters. Moreover, the integration of stochastic approach possibilities such as chaos polynomial would simplify the uncertainty quantification. From applications point of view in the

context of LNCMI, the electro-thermal model presented here would be enrich with other physics - such as magnetostatic or elasticity - and evolve to handle transient simulations. Finally the possibilities offered by lego simulation linked with domain decomposition techniques (see [Vallaghé and Patera, 2012]) are very appealing for such large applications, involving multi-physics simulations on complex geometries. It would for example allow to perform numerical experiments on complete set of magnets by assembling magnet components.

Acknowledgements

The authors wish to thank Vincent Chabannes for fruitful discussions during the Cemracs 2012. Stéphane Veys and Christophe Prud'homme acknowledge the financial support of the project ANR HAMM ANR-2010-COSI-009 and FNRAE-RB4FASTIM. Finally the authors wish to thank the Cemracs 2012 and its organizers.

References

[A.Bourquard et al., 2010] A.Bourquard, D.Bresson, A.Daël, F.Debray, P.Fazilleau, B.Hervieu, W.Joss, F.P.Juster, C.Mayri, P.Pugnat, J.M.Rifflet, L.Ronayette, C.Trophime, C.Verwaerde, and L.Villars (2010). The 42+ t hybrid magnet project at cnrs-lncmi-grenoble. *Journal of Low Temperature Physics*, 159:332–335.

[Balay et al., 2012a] Balay, S., Brown, J., Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., Curfman McInnes, L., Smith, B. F., and Zhang, H. (2012a). *PETSc Users Manual*.

[Balay et al., 2012b] Balay, S., Brown, J., Buschelman, K., Gropp, W. D., Kaushik, D., Knepley, M. G., Curfman McInnes, L., Smith, B. F., and Zhang, H. (2012b). *PETSc Web page*.

[Balay et al., 1997] Balay, S., Gropp, W. D., Curfman McInnes, L., and Smith, B. F. (1997). *Efficient Management of Parallelism in Object Oriented Numerical Software Libraries*.

[Barrault et al., 2004] Barrault, M., Maday, Y., Nguyen, N. C., and Patera, A. (2004). An empirical interpolation method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667–672.

[Chabannes, 2013] Chabannes, V. (2013). *Vers la simulation des écoulements sanguins*. PhD thesis, Université Joseph Fourier, Grenoble.

[Cuong, 2005] Cuong, N. N. (2005). Reduced-basis approximations and a posteriori error bound for nonaffine and nonlinear partial differential equations: Application to inverse analysis.

[Debray et al., 2002] Debray, F., Jongbloets, H., Joss, W., Martinez, G., Mossang, E., Petmezakis, P., Picoche, J., Plante, A., Rub, P., Sala, P., and Wyder, P. (2002). The grenoble high magnetic field laboratory as a user facility. *IEEE transactions on applied superconductivity,*, 12(1).

[Dutfoy et al., 2009] Dutfoy, A., Dutka-Malen, I., Pasanisi, A., Lebrun, R., Mangeant, F., Gupta, J. S., Pendola, M., and Yalamas, T. (2009). OpenTURNS, an Open Source initiative to Treat Uncertainties, Risks'N Statistics in a structured industrial approach. In *41èmes Journées de Statistique, SFdS, Bordeaux (France)*.

[F.Debray et al., 2012] F.Debray, J.Dumas, C.Trophime, and N.Vidal (2012). Dc high field magnets at the lncmi. *IEEE transactions on applied superconductivity,*, 22(3).

[Geuzaine and Remacle, 2009] Geuzaine, C. and Remacle, J.-F. (2009). *Gmsh: a three-dimensional finite element mesh generator with built-in pre-and post-processing facilities*.

[Grepl et al., 2007] Grepl, M., Maday, Y., Nguyen, N. C., and Patera, A. (2007). Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 41(03):575–605.

[Guennebaud et al., 2010] Guennebaud, G., Jacob, B., et al. (2010). Eigen v3. http://eigen.tuxfamily.org.

[Hernandez et al., 2005] Hernandez, V., Roman, J. E., and Vidal, V. (2005). SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Transactions on Mathematical Software*, 31(3):351–362.

[Huynh et al., 2007] Huynh, D., Rozza, G., Sen, S., and Patera, A. (2007). A successive constraint linear optimization method for lower bounds of parametric coercivity and inf–sup stability constants. *Comptes Rendus Mathematique*, 345(8):473–478.

[Knezevic and Peterson, 2010] Knezevic, D. and Peterson, J. (2010). A high-performance parallel implementation of the certified reduced basis method. Pre-print submitted to CMAME.

[Nguyen et al., 2009] Nguyen, N., Rozza, G., Huynh, D., and Patera, A. (2009). Reduced basis approximation and a posteriori error estimation for parametrized parabolic pdes; application to real-time bayesian parameter estimation. *Biegler, Biros, Ghattas, Heinkenschloss, Keyes, Mallick, Tenorio, van Bloemen Waanders, and Willcox, editors, Computational Methods for Large Scale Inverse Problems and Uncertainty Quantification, John Wiley & Sons, UK*.

[Patera and Rozza, 2007] Patera, A. and Rozza, G. (2007). Reduced basis approximation and a posteriori error estimation for parametrized partial differential equations. MIT Pappalardo Graduate Monographs in Mechanical Engineering. Copyright MIT 2006-2007.

[Prud'homme, 2006] Prud'homme, C. (2006). A domain specific embedded language in C++ for automatic differentiation, projection, integration and variational formulations. Scientific Programming, 14(2):81-110.

[Prud'Homme et al., 2012a] Prud'Homme, C., Chabannes, V., Doyeux, V., Ismail, M., Samake, A., and Pena, G. (2012a). Feel++: A computational framework for galerkin methods and advanced numerical methods. In *ESAIM: PROCEEDINGS*, pages 1–10.

[Prud'Homme et al., 2012b] Prud'Homme, C., Chabannes, V., Doyeux, V., Ismail, M., Samake, A., Pena, G., Daversin, C., and Trophime, C. (2012b). Advances in feel++ : A domain specific embedded language in C++ for partial differential equations. European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2012).

[Prud'homme and Patera, 2004] Prud'homme, C. and Patera, A. (2004). Reduced-basis output bounds for approximately parameterized elliptic coercive partial differential equations. *Computing and Visualization in Science*, 6(2-3):147–162.

[Prud'homme et al., 2002] Prud'homme, C., Rovas, D. V., Veroy, K., Machiels, L., Maday, Y., Patera, A. T., and Turinici, G. (2002). Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods. *Journal of Fluids Engineering*, 124(1):70–80.

[Quarteroni et al., 2011] Quarteroni, A., Rozza, G., and Manzoni, A. (2011). Certified reduced basis approximation for parametrized partial differential equations and applications. *Journal of Mathematics in Industry*, 1(1):1–49.

[Rozza et al., 2007] Rozza, G., Huynh, D., and Patera, A. (2007). Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *Archives of Computational Methods in Engineering*, 15(3):1–47.

[Smith et al., 2004] Smith, B., Bjorstad, P., and Gropp, W. (2004). *Domain decomposition: parallel multilevel methods for elliptic partial differential equations*. Cambridge University Press.

[Vallaghé et al., 2011] Vallaghé, S., Fouquembergh, M., Le Hyaric, A., and Prud'Homme, C. (2011). A successive constraint method with minimal offline constraints for lower bounds of parametric coercivity constant.

[Vallaghé and Patera, 2012] Vallaghé, S. and Patera, A. (2012). The static condensation reduced basis element method for a mixed-mean conjugate heat exchanger model. Submitted to SIAM Journal on Scientific Computing.

[Veroy et al., 2003a] Veroy, K., Prud'homme, C., and Patera, A. (2003a). Reduced-basis approximation of the viscous Burgers equation: Rigorous *a posteriori* error bounds. *C. R. Acad. Sci. Paris, Série I*, 337(9):619–624.

[Veroy et al., 2003b] Veroy, K., Prud'homme, C., Rovas, D. V., and Patera, A. (2003b). *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*.

[Works et al., 2012] Works, E. I., Development, E. R. ., and PhiMECA (2012). OpenTURNS Web page. http://www.openturns.org.