CHAPTER 5: INTRODUCTION TO REDUCED BASIS

1. Introduction of the method

In this part, we will present another Galerkin Method, The Reduced Basis (RB), that apply when *parametrized* PDEs are considered.

1.1. **A finite dimensional example.** Let us present quickly this approach on a very simple example. Suppose one wants to solve the system

$$A(\mu)x = b(\mu), \tag{1}$$

where $A(\mu)$ and $b(\mu)$ are high dimensional matrix and vector, say $A(\mu) \in \mathbb{R}^{H,H}$ and $b(\mu) \in \mathbb{R}^H$, with $H >> 1$. The vector $\mu \in \mathcal{P}$ collects all the parameters of our problem, e.g. some coefficients, or parameter associated with some sets of coefficients in $A(\mu)$ and $b(\mu)$. By $\mathcal{P}$, we denote the set of admissible μ. Suppose also that a set of M solutions $(x(\mu_1), \ldots, x(\mu_M))$ associated with various parameters vectors $\mu_1, \ldots, \mu_M$ have been computed, from which a (reduced !) basis $(e_1, \ldots, e_N)$ has been extracted, with N small, $1 < N \leq M << H$. Let us assemble this basis in a matrix by $P = (e_1|\ldots|e_N)$. Now, instead computing a solution of (1), we look for a solution in $\text{Span}(e_1, \ldots, e_N)$. This means that we want to solve

$$A(\mu)Py = b(\mu),$$

which is problematic, since one has much more equations than unknowns. Instead, we decide to project these equations on the basis $(e_1, \ldots, e_N)$ and then consider

$$P^T A(\mu)Py = b(\mu), \tag{2}$$

which is now a $N \times N$ system. Note that this idea exacty corresponds to the Galerkin reduction, where one replaces u and v by u_h and v_h in a smaller space. In our case, we set $u_h = u_N = Py$ and $v_h = Pz$, which in turns gives[1]

$$a(u_h, v_h) = u_h^T A^T v_h = b(v_h) = B^T v_h$$

, i.e.

$$(Pz)^T A(Py) = (Pz)^T B,$$

which should holds for all z, which implies (2).

If the basis $(e_1, \ldots, e_N)$ correctly approximates the manifold of solutions $\{x(\mu) \in \mathbb{R}^H, \mu \in \mathcal{P}\}$, then we can expect that $x_N := Py \approx x$ in an accurate way.

Next, suppose that we further want to get a bound that quantify this approximation. We introduce $e := x_N - x$ and the *residual* $r := b(\mu) - A(\mu)x_N$. Remark that:

- the residual satisfies

  $$r = A(\mu)e,$$

  which means that there is a strong relation between r and the error $e := x - x_N$.

[1] We denote by A the matrix associated with a, i.e., $A = [a(f_j, f_i)]_{i,j=1,\ldots,H}$.

- the residual is an "a posteriori" quantity, in the sense that it can be computed using only the approximation x_N, i.e., without using the true solution x.

If $A(\mu)$ is invertible, we get

$$\|e\| \leq \|A(\mu)^{-1}\| \|r\|,$$

hence a bound for the error, that one can (up to some supplmentary informations about A) be estimated a posteriori.

We will detail this approach in the more general setting of elliptic equations.

1.2. **Offline-Online decomposition.** In this approach, we see that the approximation x_N results from a process involving two phases:

- An *offline phase*, where the solutions M solutions $(x(\mu_1), \dots, x(\mu_M))$ are computed and the basis $(e_1, \dots, e_N)$ is extracted.
- An *online phase*, where the system (2) is solved

The former may be be very expansive, but it is assumed that the computational time of this phase is not problematic, hence a very fine solving can be considered, meaning $H >> 1$. The latter, on the contrary, needs to be achieved very quickly. Hence the assumption $1 \leq N << H$.

2. General Setting

Given an Hilbert space X (endowed with the norm $\|\cdot\|$), we consider a parametrized elliptic problem: *Find $u \in X$ such that*

$$a(u, v; \mu) = b(v; \mu), \forall v \in X.$$

We assume moreover that the Lax-Milgram's assumptions are satisfied, and denote by $\alpha(\mu)$, $\gamma_a(\mu)$ and $\gamma_b(\mu)$ the coercivity constant of a, the continuity constant of a and the continuity constant of b, respectively. Recall that:

$$\alpha(\mu) = \inf_{u \in X} \frac{a(u, u; \mu)}{\|u\|^2}$$
$$\gamma_a(\mu) = \sup_{u,v \in X} \frac{a(v, u; \mu)}{\|u\|\|v\|}$$
$$\gamma_b(\mu) = \sup_{u \in X} \frac{b(u; \mu)}{\|u\|}.$$

Suppose moreover that for some reason, one is not interested in the whole solution u, but rather focuses on a scalar quantity $s(\mu)$ associated with u, e.g., its mean value over a subdomain, a boundary value, etc. Let us suppose that this *quantity of interest* consists of a linear form ℓ on u, i.e.

$$s(\mu) = \ell(u; \mu).$$

One usually considers this quantity to tackle *goal-oriented* applications. The form ℓ is assumed to be continuous, with a constant $\gamma_\ell(\mu)$. We will see that such a situation may help in designing specific basis to refine relevantly the computation and give an accurate approximation of $s(\mu)$.

3. Error analysis

We start with some results about the error induced by the method. Let us suppose that we have computed a set of snapshots $u(\mu_1), \dots, u(\mu_M)$, from which we have extracted a basis $e_1, \dots, e_N$. Denote by $X_N = \text{span}(e_1, \dots, e_N)$ the reduced space. The reduced problem then reads: *Find $u_N \in X_N$ such that*

$$a(u_N, v_N; \mu) = b(v_N; \mu), \forall v_N \in X_N.$$

As usual with Galerkin Methods, Lax-Milgram assumptions are satisfied in the reduced framework as soon as they are satisfied in the full framework. We also introduced a reduced version of the quantity of interest:

$$s_N(\mu) = \ell(u_N; \mu).$$

To perform our a posteriori analysis, we will make use of a particular linear mapping called *residual* (similar to the one introduced in Section 1.1) and defined on X by

$$r(v; \mu) := b(v; \mu) - a(u_N, v; \mu).$$

Of course, the residual cancels in the case $u_N = u$. By definition of u_N, it also cancels for all v in X_N. Next, it has the two properties already mentionned, namely, it is an a posteriori quantity, and it satisfies

$$r(e; \mu) := a(e, v; \mu),$$

where $e = u - u_N$. Let us mention a last property, usually called *reproduction property.*

Lemme 1. *If $u(\mu) \in X_N$, for a certain $\mu \in \mathcal{P}$, then $u = u_N$.*

Proof. We have:

$$\begin{aligned} \alpha(\mu)\|e\|^2 \leq & a(e, e; \mu) \\ = & a(u(\mu), e; \mu) - a(u_N(\mu), e; \mu) \\ = & b(e; \mu) - b(e; \mu) \\ = & 0, \end{aligned}$$

where we have used that $u(\mu)$ (hence also e) belongs to X_N. □

This result can be used profitably as a debbuging test.

3.1. Primal RB-error.

Let us present a first a posteriori error estimator.

Theorme 1. *We keep the previous assumption (coercivity of a and continuity of b and ℓ). We have:*

$$\|e\| \leq \Delta_u(\mu) := \frac{\|r(\cdot, \mu)\|}{\alpha(\mu)} \tag{3}$$

$$|s(\mu) - s_N(\mu)| \leq \Delta_s(\mu) := \|\ell(\cdot; \mu)\| \Delta_u(\mu). \tag{4}$$

Proof. We only establish (3) (the proof of (4) follows naturally from it). We have:

$$\begin{aligned} \alpha(\mu)\|e\|^2 \leq & a(e, e; \mu) \\ = & r(e; \mu) \\ \leq & \|r(\cdot, \mu)\| \|e\|, \end{aligned}$$

and the result follows. □

As corollary of the previous result one has that both $\Delta_u(\mu)$ and $\Delta_s(\mu)$ cancel in the case $e = 0$.

3.2. **Primal-Dual RB-error.** We can go further than the previous result (at least when considering $\Delta_s(\mu)$) by means of the auxiliary problem: *Find $u^d \in X$ such that*

$$a(v, u^d; \mu) = -\ell(v; \mu), \forall v \in X.$$

Here, the subscript ^{d} stands for "dual". This new function can also be approximated by reduction. Let us denote by $X_N^d \subset X$ the corresponding Galerkin space. The resulting reduced problems reads: *Find $u_N^d \in X$ such that*

$$a(v, u_N^d; \mu) = -\ell(v; \mu), \forall v \in X_N^d.$$

The associated resuidual is given by

$$r^d(v; \mu) = -a(v, u_N^d; \mu) - \ell(v; \mu).$$

We also introduce the corrected apporximation of the quantity of interest:

$$s'_N(\mu) = s_N(\mu) - r(u_N^d; \mu).$$

This approximation improves $s_N(\mu)$ in the following sense.

Theorme 2. *Keeping the previous notation and assumptions, we have:*

$$\|u^d(\mu) - u_N^d(\mu)\| \leq \Delta_u^d(\mu) := \frac{\|r^d(\cdot, \mu)\|}{\alpha(\mu)} \tag{5}$$

$$|s(\mu) - s'_N(\mu)| \leq \Delta'_s(\mu) := \frac{\|r^d(\mu)\|\|r(\mu)\|}{\alpha(\mu)}. \tag{6}$$

Proof. The proof of (5) is similar to the one of (3). To get (6), we write:

$$\begin{aligned} s(\mu) - s'_N(\mu) =& \ell(u; \mu) - \ell(u_N; \mu) + r(u_N^d; \mu) \\ =& \ell(u - u_N; \mu) + r(u_N^d; \mu) \\ =& - a(u - u_N, u^d; \mu) + b(u_N^d; \mu) - a(u_N, u_N^d; \mu) \\ =& - a(u - u_N, u^d - u_N^d; \mu). \end{aligned}$$

Denoting by $e^d = u^d - u_N^d$ the dual error, we then have:

$$\begin{aligned} |s(\mu) - s'_N(\mu)| =& |a(e, e^d; \mu)| \\ =& |r(e^d; \mu)| \\ \leq& \|r(\cdot; \mu)\|\|e^d(\mu)\| \\ \leq& \|r\|\Delta_u^d(\mu) \\ =& \frac{\|r\|\|r^d(\cdot; \mu)\|}{\alpha(\mu)}, \end{aligned}$$

which is the desired conclusion. □

4. Computational considerations

Recall first that the solving of system of size k is associated with a computational of order $\mathcal{O}(k^3)$ for full matrices and $\mathcal{O}(k^2)$ for sparse matrices. Suppose that the high dimensional is solved with finite elements, hence using sparse matrices. The offline solving of a large problem then requires $\mathcal{O}(H^2)$ multiplications whereas the online solving (which deals with full matrices) requires $\mathcal{O}(N^3)$ multiplications. The reduced basis will consequently be efficient if $N << H^{2/3}$.

Let us see in more detail how a reduced basis solving proceeds. One needs to compute:

1. $M \approx N$ snapshots: $\mathcal{O}(NH^2)$,
2. N^2 evaluations of $a(e_j, e_i; \mu)$ to assemble the reduced matrix: $\mathcal{O}(N^2 H)$,
3. N evaluations of $b(e_i; \mu)$ to assemble the reduced righ-hand side term: $\mathcal{O}(NH)$,
4. Solution of the reduced $N \times N$ system: $\mathcal{O}(N^3)$.

In this process, we need to assemble the large (finite element) matrix for each new value μ. This can be avoided in the case the operators are *parameters decomposable*. This means that the operators involved in our elliptic problem are of the form

$$\begin{aligned} a(u, v; \mu) &= \sum_{q=1}^{Q_a} \Theta_q^a(\mu) a_q(u, v), \\ b(v; \mu) &= \sum_{q=1}^{Q_b} \Theta_q^b(\mu) b_q(v), \\ \ell(v; \mu) &= \sum_{q=1}^{Q_\ell} \Theta_q^\ell(\mu) \ell_q(v), \end{aligned}$$

where the operators $a_q(u, v)$, $b_q(v)$ and $\ell_q(v)$ are parameters independent. It follows that the reduced matrix $A_N(\mu) = [a(e_j, e_i; \mu)]_{i,j=1,\ldots,N}$ and vectors $B_N(\mu) = [b(e_i; \mu)]_{i=1,\ldots,N}$ and $L_N(\mu) = [\ell(e_i; \mu)]_{i=1,\ldots,N}$ read:

$$\begin{aligned} A_N(\mu) &= \sum_{q=1}^{Q_a} \Theta_q^a(\mu) A_q, \\ B_N(\mu) &= \sum_{q=1}^{Q_b} \Theta_q^b(\mu) B_q(v), \\ L_N(\mu) &= \sum_{q=1}^{Q_\ell} \Theta_q^\ell(\mu) L_q(v), \end{aligned}$$

hence, up to a precomputation of A_q, B_q and L_q during the offline phase, the reduced system can be assembled in the online phase in a computational time independent from the high dimension H. The complexity of the reduced basis method can then be described as follows:

- offline phase: $\mathcal{O}(NH^2 + NH(Q_b + Q_\ell) + N^2 H Q_a)$,
- online phase: $\mathcal{O}(NQ_a + N(Q_b + Q_\ell) + N^3)$.

Note that evaluation of the residuals r and r^d can also be offline/online decomposed !

5. Basis generation

It remains to discuss the choice of the parameters $\mu_1, \dots, \mu_M$ used to compute the snapshots. Hence, this last section only deals with the offline phase, where one assumes to not care about the computational time (there are no *real time* computations constraint). We will see that there are anyway clever ways to proceed to scan $\mathcal{P}$ accuratly without performing a large number of high dimensional solutions.

Usual strategies consist either in computing a large number of snapshots and extract a basis by means of an SVD procedure, or in considering a greedy approach. We will now described the latter technique. Assume that we have an error bound $\Delta(Y, \mu)$, with $\|e(\mu)\| \leq \Delta(Y, \mu)$, where Y is the considered RB-Galerkin space. The greedy algorithm iterates as follows. Given a current space X_n,

(1) Set $\mu_{n+1} := \text{ argmax}_{\mu \in \mathcal{P}}(\Delta(X_n; \mu))$,
(2) Define $e_{n+1} := u(\mu_{n+1})$, $X_{n+1} := \text{ span}(X_n, e_{n+1})$.

Note that an orthonormalization of the resulting basis is generally done, since it helps in stabilizing the reduced solving.

In parctice, the maximization step to compute μ_{n+1} is not done by scanning $\mathcal{P}$, but a finite subset $\mathcal{P}_{train} \subset \mathcal{P}$ usually called *training set.*

Finally, three ways are usually considered to define $\Delta(Y, \mu)$:

(1) "Céa strategy": in this approach, one considers the error projection, namely

$$\Delta(Y, \mu) := \inf_{v \in Y} \|u(\mu) - v\|.$$

The drawback of this strategy is that it requires computations involving problems of dimension H. The advantage is that it does not require anything else than the mapping $\mu \mapsto u(\mu)$, which can consequently considered as a black box: one does not need to go into the code to extract matrices or vectors.

(2) True error: in this approach, one defines

$$\Delta(Y, \mu) := \|u(\mu) - u_N(\mu)\|.$$

This approach has the advantage that it really consider the error...hence, one can really select the worst snapshot (i.e. the parameter μ which maximizes the error). On the other hand, it has the drawbacks that it requires computations involving problems of dimension H, and that it is very intrusive: one needs indeed to assemble the reduced problem, hence extract the matrix associated with a, the right-hand side term...The method is said to be *intrusive.*

(3) A posteriori estimate. In this case one simply defines

$$\Delta(Y, \mu) := \Delta_u(\mu).$$

If the residual is offline/online decomposable, this method result in only N high dimensional computations. This means that very large training set $\mathcal{P}_{train}$ can be considered during the first step of the greedy algorithm. But this method is also intrusive.

Here, we have only considered the error e, but one can easily get similar methods for goal-oriented approaches: one the needs to consider $s(\mu)$ and the associated error estimates.

6. To go further

To read more about the reduced basis, we recommend the book

Reduced Basis Methods for Parametrized PDEs A Tutorial Introduction for Stationary and Instationary Problems,

from B. Haasdonk. It can be downloaded here

`https://pnp.mathematik.uni-stuttgart.de/ians/haasdonk/publications/RBtutorial_preprint_update_with_header.pdf`