The Reference Point Method, a "hyperreduction" technique: Application to PGD-based nonlinear model reduction

Matteo Capaldo, Pierre-Alain Guidault, David Néron, Pierre Ladevèze

The Reference Point Method, a "hyperreduction" technique: application to PGD-based nonlinear model reduction

M. Capaldo[a], P.-A. Guidault[b,*], D. Néron[b], P. Ladevèze[b]

[a] *EDF R&D, THEMIS/R22, 91120 Palaiseau, France*
[b] *LMT, ENS Cachan, CNRS, Université Paris-Saclay, 94235 Cachan, France*

Abstract

A new approximation technique, called Reference Point Method (RPM), is proposed in order to reduce the computational complexity of algebraic operations for constructing reduced-order models in the case of time dependent and/or parametrized nonlinear partial differential equations. Even though model reduction techniques enables one to decrease the dimension of the initial problem in the sense that far fewer degrees of freedom are needed to represent the solution, the complexity of evaluating the nonlinear terms and assembling the low dimensional operator associated with the reduced-order model still scales with the size of the original high-dimensional model. This point can be critical, especially when the reduced-order basis changes throughout the solution strategy as it is the case for model reduction techniques based on Proper Generalized Decomposition (PGD). Based on the concept of spatial, parameter/time reference points and influence patches, the RPM defines a *compressed* version of the data from which an approximate low-rank separated representation by patch of the operators can be constructed by explicit formulas at low-cost without resorting to SVD-based techniques. An application of the RPM to PGD-based model reduction for a nonlinear parametrized elliptic PDE previously studied by other authors with reduced-basis method and EIM is proposed. It is shown that computational complexity to construct the reduced-order model can be divided in practice by one order of magnitude compared with the classical PGD approach.

Keywords: RPM, nonlinear model reduction, PGD, reduced basis, LATIN method, hyperreduction

1. Introduction

Numerical simulation has been playing an increasingly important role in science and engineering due to the need to describe realistic scenarios and derive tools to facilitate the virtual design of new structures while reducing the use of real prototypes. Most of the time, engineers are interested into specific quantities called *outputs* (e.g. forces, critical stresses, pressure drops...) in some particular zones, rather than into the description of the model at each point of the considered domain. These outputs are functions of the system parameters, called *inputs* (e.g. geometry parameter, material properties, boundary conditions and loadings...), that define a particular configuration of the model. The evaluation of these outputs needs the solution of the underlying PDE that requires a computational cost which, in some engineering fields, can be very high, especially when the simulations concern nonlinear analyses of complex high-fidelity models. Moreover, engineering design and optimization may require thousands of these evaluations, sometimes in real-time. A challenging issue would be to provide engineers with a kind of *Virtual Chart* constructed during an *offline* stage that can be CPU intensive. Once constructed, it can be used *online* to solve real-time or many queries simulations. Despite of the continuing progress in computer speeds and hardware capabilities, the construction of those charts would be hardly tractable with classical numerical approaches.

*Corresponding author
Email address: pierre-alain.guidault@ens-paris-saclay.fr (P.-A. Guidault)

Model reduction techniques aim at circumventing this obstacle by seeking the solution of a given problem in a reduced-order basis (ROB), whose dimension is much smaller than the size of the original high-dimensional model. These techniques take advantage of the redundancy of information that usually exists when describing the solution. In applied mathematics, the first proposed model reduction technique was the Proper Orthogonal Decomposition (see [1, 2, 3, 4, 5] or the review [6] for more examples). Techniques based on POD involve a learning phase which consists in solving the problem at some particular time instants and/or parameter values arbitrarily chosen, either from the full-order model in space or, sometimes, from a simplified model. These solutions, called snapshots, can be CPU-expensive. A ROB is subsequently formed by considering only the most relevant POD modes of these snapshots. The reduced-order model (ROM) is then classically generated by Galerkin projection onto the ROB and solved for the entire time and/or parameter domain, but other approaches such as Petrov-Galerkin or minimization techniques can also be used. The strong point of these techniques is the fact that, the number of the most relevant POD modes is often much lower than the size of the full-order model in space, however it is *case sensitive* [7]. For instance, in dynamics computations, changes in boundary conditions can drastically affect the POD accuracy and a high number of snapshots may be required [7, 8]. Since the 80's, the Reduced-Basis (RB) approach has been developed and consists in a greedy algorithm to select the most relevant calculations to be performed on the parametric space in order to enrich the ROB [9, 10, 11, 12, 13, 14, 15, 16, 17]. In this technique, snapshots are selected by a greedy algorithm such that the new $(n+1)^{th}$ snapshot minimizes the residue, associated with a chosen norm, of the solution achieved by solving the original problem projected onto the n-order reduced basis. In that way, this strategy ensures the quality of the reduced-order basis for the construction of the ROM and palliates the *case-sensitivity* of POD.

Another appealing family of model reductions which has received a growing interest during the last decade is based on the Proper Generalized Decomposition (PGD). This technique does not need to solve the full-order model since it does not require snapshots to build up the ROB. Basically, PGD consists in seeking the solution of a problem in a relevant ROB which is generated automatically and on-the-fly by a greedy or power iterations algorithm. The interested reader is referred to [18] for a review on PGD-based techniques and to the book [19] for a handbook on separated representations and model reduction techniques. A valuable survey on the use of separated representations for solving parametric models can also be found in [20]. In the field of computational mechanics, PGD was used in [21] under the name of *radial loading approximation* as one of the fundamental ingredients of the LATIN (Large Time Increment) method [22], a non-incremental solution strategy for nonlinear evolution problems. By an alternating-direction scheme, the strategy generates approximations of the solution on the entire time-space domain by successive corrections and, consequently, is likely appropriate for time-space separated representation. In this context, PGD enabled to circumvent efficiently storage issue of iterates defined on the whole time-space domain while saving drastically the computational time. One of the significant improvements of the strategy concerns the introduction of a *Preliminary step*, which consists in first updating the time functions at each LATIN iteration by using the ROB generated at the previous iteration in order to compute a new iterate [23, 24, 25, 26, 27, 28]. Called *Update step* in [29, 30] with Newton-like solution strategies, this step has proved efficiency to reduce the number of PGD functions generated while keeping the most relevant ones to produce the best approximate separated representation of the solution. However, this step can be time consuming in practice and may represent between fifty and seventy percent of the total CPU time [31].

More generally, model reduction techniques are particularly efficient when the ROM needs to be constructed only once or when this step can be performed offline, prior to the online resolution of this model which can then be very fast. This is the case of parametrized time-invariant systems [32], linear stationary and quasi-stationary systems whose operators are affine functions of the input parameters [16]. On the contrary, when the projection is applied to linear dynamics or stationary systems with non-affine parameter dependence, or general nonlinear problems, the resulting ROM is costly to assemble, decreasing the efficiency of reduced-order modeling. Indeed, for any model reduction technique based on the projection of the problem on a given reduced-order basis, the computational complexity of evaluating the nonlinear terms (Jacobian or residue of the solution strategy) and assembling the ROM's low dimensional operator scales with the size of the original high-dimensional model. This point can be critical especially when the reduced-order basis changes throughout the solution strategy as it is the case for model reduction tech-

niques based on Proper Generalized Decomposition (PGD). This makes the bottleneck of model reduction strategies applied to nonlinear problems.

Several techniques have been introduced in the literature in order to tackle this issue, especially for model reduction techniques based on a learning stage (POD-Galerkin or reduced basis method for instance). Among them, the Empirical Interpolation Method (EIM) is probably one of the most popular. It has been developed in particular for linear elliptic problems with non-affine parameter dependence [33] as well as for nonlinear elliptic and parabolic problems [15]. Several variants can be found in the literature. To name a few, one can cite the Best Interpolation Point Method (BPIM) [34] or the Discrete EIM (DEIM) [35]. Another family of techniques that tackles nonlinear reduced-order modeling is the one that belongs to the Gappy-POD application, as the A priori Hyper-Reduction (APHR) [36], the Missing Point Estimation (MPE) [37], the Gauss Newton with approximated tensors (GNAT) [38]. More recently, [39] has introduced the Energy-Conserving Sampling and Weighting (ECSW), allowing not only to focus on the accuracy of the approximated function, but also to preserve the energetic aspect of the solution.

Concerning PGD-based model reduction, techniques to overcome the aforementioned bottleneck seems to be by far less numerous. As previously said, the possibility to enrich the reduced-order basis throughout the computation is definitely an advantage to tackle nonlinear problems efficiently. However the fact that ROB changes quite often makes the previous techniques such as EIM less suitable in the context of PGD, a priori. Indeed, for a fixed number of best/magic interpolation points the error in the coefficient functions used to interpolate the nonlinear terms ultimately dominates when the size of the ROB increases [15]. Adding more magic points may be necessary as long as the ROB evolves. Such an adaptative strategy can obviously be derived but it may be not very efficient in this context, at least in the LATIN framework.

In the present article, a more pragmatic approach is proposed. Based on the concept of spatial, time and parameter reference points associated to influence patches, the Reference Point Method (RPM) defines a *compressed* version of the data from which an approximate low-rank separated representation by patch of operators can be constructed by explicit formulas at low-cost. In this work, RPM is applied to PGD-based nonlinear model reduction with the LATIN solution strategy. More precisely, it is used at each LATIN iteration to solve the *Preliminary step*, which consists in an update of the reduced model from the current ROB. As previously said, this step is time consuming and may represent up to seventy percents of the total CPU time. In practice, the RPM enables to decrease the cost of this stage of one order of magnitude to make it represent less than ten percents of the total computational time, which is very promising. It is important to note that contrary to an interpolation technique, it is rather an approximation technique of the integrals involved in the ROM construction similarly to quadrature techniques in classical finite element methods. The key point is that, even though the integral computations are not enough accurate, the PGD-model reduction technique can anyway improve it by adding new correction functions to the ROB at the next LATIN iterations, which ensures the convergence of the whole process.

The article is structured as follows. A reference problem previously investigated by other authors by reduced-basis techniques with EIM [15] or POD with DEIM [35] is first presented in Section 2. The studied nonlinear parametrized elliptic PDE is solved by the LATIN-PGD method and the aforementioned bottleneck of nonlinear model reduction is illustrated. The Reference Point Method is presented in Section 3 and its basic features (reference points, patches, generalized components) are introduced. It is shown that the space of generalized components provides a framework that presents interesting properties dealing with the elementary algebraic operations [40], which simplifies the evaluation of integrands. The way to reconstruct by explicit formulas an approximate low-rank separated representation by patch of operators is also detailed. Keeping a separated representation of the quantities and operators without resorting to SVD-based techniques is indeed a great advantage regarding integral computation by separation of variables. A comparison between the RPM applied to PGD-based nonlinear model reduction and EIM with reduced-order basis technique is proposed in Section 4 in order to appreciate and illustrate the performances of the RPM.

2. PGD-based model reduction of a nonlinear parametrized elliptic PDE

The reference problem that motivates the study is first presented in **Section 2.1**. It was previously investigated through EIM with reduced-basis technique in [15] and DEIM with POD in [35]. In these works,

a Newton-based nonlinear solution strategy was used. Here, an alternating-direction scheme, the LATIN method, is used as detailed in **Appendix A**. The LATIN iterative scheme generates approximations of the solution on the entire space-parameter domain by successive corrections and, consequently, is likely appropriate for space-parameter separated representation. A PGD-based model reduction is introduced within the LATIN method as detailed in **Section 2.2**. The issue of computational complexity of nonlinear model reduction is discussed in **Section 2.3**.

2.1. Reference problem formulation

Let us consider as a reference problem, the nonlinear parametrized elliptic PDE first studied in [15, 35] defined on a spatial domain $\Omega \subset \mathbb{R}^2$ and a parameter domain $\mathcal{D} \subset \mathbb{R}^2$ and stated as follows:

Problem 1 (Strong form). *Find* $u(\boldsymbol{x}, \boldsymbol{\mu})$ *with* $\boldsymbol{x} = (x_1, x_2) \in \Omega =]0,1[^2$ *and* $\boldsymbol{\mu} = (\mu_1, \mu_2) \in \mathcal{D} = [0.01, 10]^2$ *and homogeneous Dirichlet boundary condition on* $\partial\Omega \times \mathcal{D}$*, such that:*

$$\left\{ \begin{array}{rcll} -\nabla^2 u + g(u;\boldsymbol{\mu}) & = & f(\boldsymbol{x}) & \text{on} \quad \Omega \times \mathcal{D} \\ u & = & 0 & \text{on} \quad \partial\Omega \times \mathcal{D} \end{array} \right.$$

with:

$$g(u;\boldsymbol{\mu}) = \frac{\mu_1}{\mu_2}\,(e^{\mu_2 u} - 1) \quad \text{and} \quad f(\boldsymbol{x}) = 100\ \sin(2\pi x_1)\ \sin(2\pi x_2)$$

This problem can be interpreted as a stationary diffusion problem with a parametrized nonlinear interior heat source term.

A space weak formulation of **Problem 1** can be proposed. In this case, the solution u is identified with a function defined on $\mathcal{D}$ with values in Hilbert space $\mathcal{V} = H_0^1(\Omega) = \{u \in H^1(\Omega);\ u = 0 \text{ on } \partial\Omega\}$, i.e. $u(\boldsymbol{\mu}) : \boldsymbol{x} \mapsto u(\boldsymbol{\mu})(\boldsymbol{x}) \approx u(\boldsymbol{x}, \boldsymbol{\mu})$, and **Problem 1** can be stated as follows:

Problem 2 (Space weak form). *Given* $\boldsymbol{\mu} \in \mathcal{D}$*, find* $u : \mathcal{D} \to \mathcal{V}$ *such that:*

$$\forall\, v \in \mathcal{V}, \quad a(u,v) + \int_\Omega g(u;\boldsymbol{\mu})\, v \,\mathrm{d}x = l(v)$$

where $a(\cdot,\cdot)$ *is a bilinear form on* $\mathcal{V}$ *and* $l(\cdot)$ *is a linear form on* $\mathcal{V}$ *defined by:*

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}x \quad \text{and} \quad l(v) = \int_\Omega f(\boldsymbol{x})\, v \,\mathrm{d}x$$

Term $g(u;\boldsymbol{\mu})$ is a nonaffine nonlinear function of the parameter $\boldsymbol{\mu}$, spatial coordinate $\boldsymbol{x}$, and field variable $u(\boldsymbol{x}, \boldsymbol{\mu})$. Bilinear term $a(u,v)$ and linear term $l(v)$ are $\mathcal{V}$-continuous bounded functionals and are parameter-independent. It is shown in [15] that **Problem 2** is well-posed and that it admits a unique solution $u \in \mathcal{V}$.

A space-parameter weak formulation of **Problem 1** can also be proposed. The following function space is introduced:

$$\mathcal{S} = L^2(\mathcal{D}, \mathcal{V}) = \{\ u\ : \mathcal{D} \to \mathcal{V},\ \int_\mathcal{D} \|u\|_\mathcal{V}^2 \,\mathrm{d}\mu\ <\ \infty\ \}$$

where $\|\cdot\|_\mathcal{V}$ is a norm on $\mathcal{V}$. A space-parameter weak formulation of **Problem 2** can be defined by the following problem:

Problem 3 (Space-parameter weak form). *Find* $u \in \mathcal{S}$ *such that:*

$$\forall\, v \in \mathcal{S}, \quad A(u,v) + G(u,v) = L(v)$$

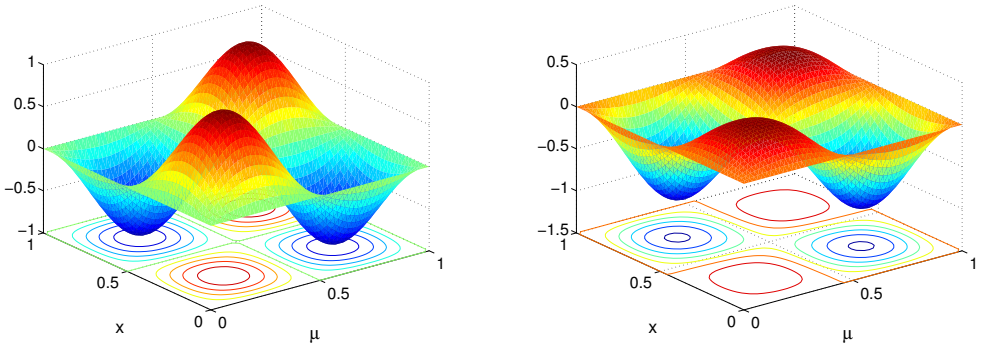
where A *and* L *are bilinear and linear forms defined by:*

$$A(u,v) = \int_\mathcal{D} a(u(\boldsymbol{\mu}), v(\boldsymbol{\mu}))\,\mathrm{d}\mu \quad \text{and} \quad L(v) = \int_\mathcal{D} l(v(\boldsymbol{\mu}))\,\mathrm{d}\mu$$

The nonlinear form G *is defined by:*

$$G(u,v) = \int_{\mathcal{D}\times\Omega} g(u;\boldsymbol{\mu})\, v \,\mathrm{d}x\,\mathrm{d}\mu$$

Problem 2 can be easily solved by introducing a finite element approximation space $\mathcal{V}_h \in \mathcal{V}$ of u and by using a Newton-based nonlinear solution strategy. In the following, the reference solution refers to the solution obtained by a finite element mesh of 50×50 bilinear quadrilateral elements leading to a finite element approximation space of dimension $N = 2601$. The approximate solution u_h of u for two sets of parameters, $\boldsymbol{\mu} = (0.01, 0.01)$ and $\boldsymbol{\mu} = (10, 10)$, is given in **Figure 1**. It can be seen that the parameter μ_1 controls the strength of the sink term whereas μ_2 changes the strength of the nonlinearity, the exponential function $\mu_1\, e^{\mu_2 u}$ increasing more the peak magnitude of the negative part of u than the magnitude of the positive one.



(a) $\boldsymbol{\mu} = (0.01, 0.01)$ (b) $\boldsymbol{\mu} = (10, 10)$

Figure 1: Solution of **Problem 2** for different parameter values and a finite element mesh of 50×50 bilinear quadrilateral elements ($N = 2601$)

2.2. The LATIN-PGD method

In order to solve **Problem 3**, the LATIN method is used as a nonlinear solution strategy (see **Appendix A** for further details). By an alternating-direction scheme, the strategy generates approximations of the solution on the entire space-parameter domain by successive corrections. In this case, it shares similar features with alternating-direction nonlinear solution strategy and augmented Lagrangian methods [41].

The LATIN iterative scheme generates approximations of the solution on the entire space-parameter domain by successive corrections and, consequently, is likely appropriate for space-parameter separated representation. By denoting $\mathcal{P} = L^2(\mathcal{D}; \mathbb{R}^2) := L^2(\mathcal{D})$ and by identifying the space $\mathcal{S} = L^2(\mathcal{D}, \mathcal{V})$ with the tensor product space $\mathcal{V} \otimes \mathcal{P}$, the correction $\delta u^{(n+1)}$ of the global stage (see **Problem 6** in **Appendix A**) is sought in $\mathcal{V} \otimes \mathcal{P}$. The following approximation of order k of the iterate $u^{(n+1)}$ is proposed:

$$u^{(n+1)}(\boldsymbol{x}, \boldsymbol{\mu}) \approx u_k^{(n+1)}(\boldsymbol{x}, \boldsymbol{\mu}) = u^{(0)}(\boldsymbol{x}, \boldsymbol{\mu}) + \sum_{i=1}^{k} \Phi_i(\boldsymbol{x})\ \lambda_i(\boldsymbol{\mu}) \tag{1}$$

where $u^{(0)} \in \mathbf{A}_d$ is an initial approximation of the solution that verifies the boundary conditions and which can be possibly given under separated representation provided that it is admissible. Each PGD pair $(\lambda_i, \Phi_i) \in \mathcal{P} \times \mathcal{V}$ is unknown and determined throughout the computation by a *greedy algorithm*. The construction of a new space function Φ_i is by far the most expensive step of this process. Thus, at a given iteration $n + 1$ of the nonlinear solver, it is advantageous to first reuse the reduced-order basis $\mathcal{W}_k = \{\Phi_i\}_{1 \leqslant i \leqslant k}$ generated up to iteration n by updating the parameter functions $\{\lambda_i\}_{1 \leqslant i \leqslant k}$ [27]. One proceeds with the *global stage* at the iteration $n + 1$ of the nonlinear solver as follows:

1. *Preliminary step*: reuse of the current reduced-order basis [23, 24, 25, 26, 27, 30]. This step consists in building an approximation of the solution, denoted by $\breve{s}^{(n+1)} = (\breve{u}^{(n+1)}, \breve{w}^{(n+1)})$, thanks to the ROB

generated at the previous iteration n of the nonlinear iterative scheme. This is similar to what is done during the *online* stage of classical reduced-order basis techniques. The main difference comes from the fact that here the ROB evolves throughout the iterations which makes pre-computations of operators inefficient. Due to the bilinearity and linearity of forms involved in (A.7) associated to global stage (see **Problem 5** in **Appendix A**), one seeks an approximation of order k of the correction $\delta u^{(n+1)}$ such that:

$$\left\{\begin{array}{rcl} \breve{u}^{(n+1)} & = & u^{(n)} + \delta u^{(n+1)} \\ \\ \delta u^{(n+1)}(\boldsymbol{x},\boldsymbol{\mu}) & \approx & \delta u_k^{(n+1)}(\boldsymbol{x},\boldsymbol{\mu}) \approx \displaystyle\sum_{i=1}^{k} \Phi_i(\boldsymbol{x})\ \delta\lambda_i(\boldsymbol{\mu}) \end{array}\right. \tag{2}$$

Assuming that an approximation of order k of $u^{(n)}$ is available from the previous iteration n of the nonlinear iterative scheme, an approximation of $\breve{u}^{(n+1)}$ can be deduced as follows :

$$\breve{u}^{(n+1)} \approx u^{(n)} + \delta u_k^{(n+1)} = u^{(0)}(\boldsymbol{x},\boldsymbol{\mu}) + \sum_{i=1}^{k} \Phi_i(\boldsymbol{x})\ \left(\lambda_i(\boldsymbol{\mu}) + \delta\lambda_i(\boldsymbol{\mu})\right) \tag{3}$$

Here, the only unknowns are the functions $\{\delta\lambda_i\}_{1\leqslant i\leqslant k}$ depending on the parameters. Given a ROB of space functions $\mathcal{W}_k = \{\Phi_i\}_{1\leqslant i\leqslant k}$, one seeks the best linear combination of this ROB which solves **Problem 6** defined in **Appendix A**. By choosing for test function $v = \Phi_j(\boldsymbol{x})\ \lambda^*(\boldsymbol{\mu})$ with $1 \leqslant j \leqslant k$, the *Preliminary step* problem reads:

Problem 4 (Preliminary/update step). *Find parameter corrections* $\{\delta\lambda_i\}_{1\leqslant i\leqslant k}$ *such that:*

$$1 \leqslant j \leqslant k,\ \forall\, \lambda^* \in \mathcal{P},$$
$$\sum_{i=1}^{k} \int_{\mathcal{D}} \lambda^* \left(a(\Phi_i,\Phi_j) + \int_{\Omega} \Phi_i\ \mathbf{H}^-(u^{(n)};\boldsymbol{\mu})\ \Phi_j\ \mathrm{d}x \right) \delta\lambda_i\ \mathrm{d}\mu = -\int_{\mathcal{D}} \lambda^*\ \mathcal{R}(u^{(n)},\Phi_j;\boldsymbol{\mu})\ \mathrm{d}\mu \tag{4}$$

It is worth noting at the fact that terms $a(\Phi_i,\Phi_j)$ are parameter-independent and can be computed once for all parameter values provided that the ROB does not change. However, nonlinear terms $\Phi_i\ \mathbf{H}^-(u^{(n)};\boldsymbol{\mu})\ \Phi_j$ and residue $\mathcal{R}(u^{(n)},\Phi_j\ \lambda^*;\boldsymbol{\mu})$ depend on $g(u^{(n)};\boldsymbol{\mu})$ and have to be evaluated for each new parameter value and iterate $u^{(n)}$ anyway. As shown in **Section 2.3**, this results in a computational cost that scales with the size of the original high-dimensional model.

2. *Preliminary step performance indicator*: The preliminary step produces a first approximation $\breve{u}^{(n+1)} \approx u^{(n)} + \delta u_k^{(n+1)}$ of **Problem 6** (see **Appendix A**) at the iteration $n+1$ by seeking the solution in the span of the already existing ROB, generated at the previous iteration n (see (2)). In order to appreciate the improvement of the solution between two consecutive LATIN iterations thanks to the preliminary step, the following performance indicator [42], based on LATIN error indicator δ_L (see (A.9)), is proposed:

$$\eta_0 = \frac{e_1 - e_2}{e_1} \geqslant 0 \quad \text{with} \quad e_1 = \frac{\|u^{(n)} - \hat{u}^{(n-1/2)}\|}{\frac{1}{2}\ \|u^{(n)} + \hat{u}^{(n-1/2)}\|} \quad \text{and} \quad e_2 = \frac{\|\breve{u}^{(n+1)} - \hat{u}^{(n+1/2)}\|}{\frac{1}{2}\ \|\breve{u}^{(n+1)} + \hat{u}^{(n+1/2)}\|} \tag{5}$$

Due to the particular choice made for search direction $\mathbf{H}^+$ (see (A.2)), one has:

$$\left\{\begin{array}{rcl} e_1 & = & \dfrac{\|u^{(n)} - u^{(n-1)}\|}{\frac{1}{2}\ \|u^{(n)} + u^{(n-1)}\|} = \dfrac{\|\delta u^{(n)}\|}{\frac{1}{2}\ \|u^{(n)} + u^{(n-1)}\|} = \delta_L^{(n)} \\ \\ e_2 & = & \dfrac{\|\breve{u}^{(n+1)} - u^{(n)}\|}{\frac{1}{2}\ \|\breve{u}^{(n+1)} + u^{(n)}\|} = \dfrac{\|\delta u_k^{(n+1)}\|}{\frac{1}{2}\ \|\breve{u}^{(n+1)} + u^{(n)}\|} \end{array}\right. \tag{6}$$

If the value of η_0 is higher than a given threshold, the correction $\delta u_k^{(n+1)}$ is significant and the global stage at the iteration $n+1$ is considered to be solved. One can proceed to a new LATIN iteration, that is to say to a new local stage. Otherwise, one proceeds to the generation of a new PGD pair as described hereafter.

3. *Generation of a new PGD pair*: The prediction $\breve{u}^{(n+1)} \approx u^{(n)} + \delta u_k^{(n+1)}$ previously computed during the preliminary step is considered to be known but it is not good enough and the performance indicator (5) is lower than a given threshold. Then a new PGD pair is sought to enrich the previous approximation by solving **Problem 6** (see **Appendix A**) by a progressive Galerkin PGD procedure. The new PGD pair is generated to approximate the correction $\delta u^{(n+1)} = u^{(n+1)} - u^{(n)}$ of the LATIN iteration $n+1$ (see (2)). Let assume that a decomposition $\delta u_k^{(n+1)}$ (denoted by δu_k in the following to alleviate the notations) of order k is known. Thus, a new approximation of $u^{(n+1)}$ can be defined as:

$$u^{(n+1)} \approx u_{k+1}^{(n+1)} = u^{(n)} + \delta u_k^{(n+1)} + \Phi\lambda = u^{(n)} + \delta u_{k+1}^{(n+1)} \tag{7}$$

The progressive definition of a new couple $(\Phi, \lambda) \in \mathcal{V} \otimes \mathcal{P}$ is defined as the one that verifies the following Galerkin orthogonality condition :

$$\forall \lambda^* \in \mathcal{V}, \forall \lambda^* \in \mathcal{P},$$
$$\int_{\mathcal{D}} a(\delta u_k + \Phi\lambda, \Phi\lambda^* + \Phi^*\lambda)\,\mathrm{d}\mu + \int_{\mathcal{D}\times\Omega} \mathbf{H}^-(u^{(n)};\boldsymbol{\mu})(\delta u_k + \Phi\lambda)(\Phi\lambda^* + \Phi^*\lambda)\,\mathrm{d}x\,\mathrm{d}\mu = \ldots$$
$$\ldots - \int_{\mathcal{D}} \mathcal{R}(u^{(n)}, \Phi\lambda^* + \Phi^*\lambda;\boldsymbol{\mu})\,\mathrm{d}\mu \tag{8}$$

That is to say:

$$\forall \Phi^* \in \mathcal{V}, \forall \lambda^* \in \mathcal{P},$$
$$\int_{\mathcal{D}} a(\Phi\lambda, \Phi\lambda^* + \Phi^*\lambda)\,\mathrm{d}\mu + \int_{\mathcal{D}\times\Omega} \mathbf{H}^-(u^{(n)};\boldsymbol{\mu})\ \Phi\lambda\ (\Phi\lambda^* + \Phi^*\lambda)\,\mathrm{d}x\,\mathrm{d}\mu = \ldots$$
$$\ldots - \int_{\mathcal{D}} \breve{\mathcal{R}}(u^{(n)}, \delta u_k, \Phi\lambda^* + \Phi^*\lambda;\boldsymbol{\mu})\,\mathrm{d}\mu \tag{9}$$

where :

$$\begin{cases} \breve{\mathcal{R}}(u^{(n)}, \delta u_k, v;\boldsymbol{\mu}) &= \mathcal{R}(u^{(n)}, v;\boldsymbol{\mu}) + a(\delta u_k, v) + \displaystyle\int_{\Omega} \mathbf{H}^-(u^{(n)};\boldsymbol{\mu})\ \delta u_k\ v\,\mathrm{d}x \\ \\ \mathcal{R}(u^{(n)}, v;\boldsymbol{\mu}) &= a(u^{(n)}, v)\ -\ \displaystyle\int_{\Omega} f(\boldsymbol{x})\ v\ \mathrm{d}x\ +\ \int_{\Omega} g(u^{(n)};\boldsymbol{\mu})\ v\ \mathrm{d}x \end{cases}$$

The two following mappings are introduced:

Definition 1 (Space function mapping). *$S_{k+1} : \mathcal{P} \to \mathcal{V}$ is the application that maps a parameter function $\lambda \in \mathcal{P}$ into a space function $\Phi = S_{k+1}(\lambda) \in \mathcal{V}$ is defined as follows:*

$$\forall \Phi^* \in \mathcal{V},$$
$$a(\Phi, \Phi^*)\left(\int_{\mathcal{D}} \lambda^2\,\mathrm{d}\mu\right) + \int_{\mathcal{D}\times\Omega} \lambda\Phi\ \mathbf{H}^-(u^{(n)};\boldsymbol{\mu})\,\Phi^*\lambda\,\mathrm{d}x\,\mathrm{d}\mu = -\int_{\mathcal{D}} \breve{\mathcal{R}}(u^{(n)}, \delta u_k, \Phi^*\lambda;\boldsymbol{\mu})\,\mathrm{d}\mu \tag{10}$$

Definition 2 (Parameter function mapping). *$P_{k+1} : \mathcal{V} \to \mathcal{P}$ is the application that maps a space function $\Phi \in \mathcal{V}$ into a parameter function $\lambda = P_{k+1}(\Phi) \in \mathcal{P}$ is defined as follows:*

$$\forall \lambda^* \in \mathcal{P},$$
$$a(\Phi, \Phi)\left(\int_{\mathcal{D}} \lambda\,\lambda^*\,\mathrm{d}\mu\right) + \int_{\mathcal{D}\times\Omega} \lambda\Phi\ \mathbf{H}^-(u^{(n)};\boldsymbol{\mu})\,\Phi\lambda^*\,\mathrm{d}x\,\mathrm{d}\mu = -\int_{\mathcal{D}} \breve{\mathcal{R}}(u^{(n)}, \delta u_k, \Phi\lambda^*;\boldsymbol{\mu})\,\mathrm{d}\mu \tag{11}$$

Definition 3 (Progressive Galerkin PGD). *A couple* $(\Phi, \lambda) \in \mathcal{V} \otimes \mathcal{P}$ *verifies* (8) *if and only if* $\Phi = S_{k+1}(\lambda)$ *and* $\lambda = P_{k+1}(\Phi)$*, i.e.* $\Phi = S_{k+1}(\lambda)$ *and* λ *(resp.* $\lambda = P_{k+1}(\Phi)$ *and* Φ*) is a fixed point of mapping* $G_{k+1} = P_{k+1} \circ S_{k+1}$ *with* $\lambda = G_{k+1}(\lambda)$ *(resp.* $G'_{k+1} = S_{k+1} \circ P_{k+1}$ *with* $\Phi = G'_{k+1}(\Phi)$*).*

Algorithm 1 Progressive PGD to generate the $(k+1)^{th}$ new PGD pair (Power iterations algorithm)

$\lambda \leftarrow \lambda^{(0)}$ ▷ Initialize parameter function, e.g. $\lambda^{(0)} = a\mu_1 + b\mu_2$ with $(a, b) \in \mathbb{R}^2$
for $l = 1$ to l_{max} **do**
  Compute $\Phi = S_{k+1}(\lambda)$ ▷ Solve problem of **Definition 1**
  Normalize Φ
  Compute $\lambda = P_{k+1}(\Phi)$ ▷ Solve problem of **Definition 2**
  Check stagnation of λ ▷ Use for instance the $L^2(\mathcal{D})$ norm
end for
$\Phi_{k+1} \leftarrow \Phi$ and $\lambda_{k+1} \leftarrow \lambda$

A power iterations algorithm that iteratively generates parameter function λ and space function Φ is used as shown in **Algorithm 1**. The interested reader could refer to [29, 22, 43] for further details. Once this new pair of parameter and space functions is computed, the $(k+1)^{th}$ space function is orthogonalized and added to the reduced-basis to form $\mathcal{W}_{k+1}$. In practice, only one new PGD pair is added per LATIN iteration to approximate correction $\delta u^{(n+1)}$ but nothing prevents one to generate more than one.

The final algorithm of the LATIN-PGD nonlinear solution strategy is finally given in **Algorithm 2** of **Appendix B**.

Remark 1. *In the case of high-dimensional parametric models, different update strategies can be proposed [30]. More precisely, for a large number* r *of parameters* $(\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_r)$*, a separated variables representation of the solution under the form* $u(\boldsymbol{x}, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_r) \approx \sum_{i=1}^{k} \Phi_i(\boldsymbol{x}) \lambda_i^1(\boldsymbol{\mu}_1) ... \lambda_i^r(\boldsymbol{\mu}_r)$ *can be proposed similarly to* (1) *(see for instance [20]). Among others, a possible strategy consists in successive updates of the* k *functions* $\{\lambda_i^l\}_{1 \leqslant i \leqslant k}$ *along each selected dimension* l *according to a particular sequence [30]. In this case, the update of the* k *functions* $\{\lambda_i^l\}_{1 \leqslant i \leqslant k}$ *for a given dimension* l *with* $\lambda_i^l \in \mathcal{P}_l$ *can be stated as* ***Problem 4*** *whose dimension is* $k \times dim(\mathcal{P}_l)$*. Obviously, the updating step along a particular dimension* l *is affordable provided that* $dim(\mathcal{P}_l)$ *is reasonably small.*

2.3. Bottleneck of nonlinear model reduction – A brief complexity analysis

In order to illustrate the bottleneck of nonlinear model reduction and to explain why the construction of the reduced-order model involved during the preliminary step is CPU intensive, let come back to **Problem 4** solved at the preliminary step. By introducing a finite element approximation space $\mathcal{V}_h \in \mathcal{V}$, with:

$$\mathcal{V}_h = \left\{ v(\boldsymbol{x}) = \sum_{i=1}^{N} v^i \varphi_i(\boldsymbol{x});\ \varphi_i \in \mathcal{V}, v^i \in \mathbb{R} \right\} \tag{12}$$

a space function $\Phi_i(\boldsymbol{x})$ can be approximated as follows : $\Phi_i(\boldsymbol{x}) \approx \sum_{j=1}^{N} \Phi_i^j \varphi_j(\boldsymbol{x}) = \Phi_i^j \varphi_j$ in indicial notation where repeated indices are summed over their range. Thus (4) can be discretized as follows:

$1 \leqslant j \leqslant k,\ \forall\, \lambda^* \in \mathcal{P},$

$$\sum_{i=1}^{k} \int_{\mathcal{D}} \lambda^*(\boldsymbol{\mu}) \left(\Phi_i^l\ \mathbf{A}_{lm}\ \Phi_j^m + \Phi_i^r\ \boldsymbol{G}_{rs}(u^{(n)}; \boldsymbol{\mu})\ \Phi_j^s \right) \delta\lambda_i(\boldsymbol{\mu})\ \mathrm{d}\mu = - \int_{\mathcal{D}} \lambda^*(\boldsymbol{\mu})\ \boldsymbol{R}_t(\boldsymbol{\mu})\ \Phi_j^t\ \mathrm{d}\mu \tag{13}$$

or:

$$1 \leqslant j \leqslant k,\ \forall\, \lambda^* \in \mathcal{P},$$
$$\sum_{i=1}^{k} \underbrace{\left(\Phi_i^l\ \mathbf{A}_{lm}\ \Phi_j^m\right)\left(\int_{\mathcal{D}} \lambda^*(\boldsymbol{\mu})\ \delta\lambda_i(\boldsymbol{\mu})\ \mathrm{d}\mu\right)}_{\text{First term}} + \underbrace{\Phi_i^r\left(\int_{\mathcal{D}} \lambda^*(\boldsymbol{\mu})\ \boldsymbol{G}_{rs}(u^{(n)};\boldsymbol{\mu})\ \delta\lambda_i(\boldsymbol{\mu})\ \mathrm{d}\mu\right)\Phi_j^s}_{\text{Second term}} = \ldots$$
$$\ldots - \Phi_j^t \int_{\mathcal{D}} \lambda^*(\boldsymbol{\mu})\ \boldsymbol{R}_t(\boldsymbol{\mu})\ \mathrm{d}\mu \quad (14)$$

where $\mathbf{A}$ and $\boldsymbol{G}$ are $N \times N$ matrices whose entries are defined by :

$$\mathbf{A}_{ij} = a(\varphi_i, \varphi_j) \quad \text{and} \quad \boldsymbol{G}_{ij} = \int_{\Omega} \varphi_i\ \mathbf{H}^{-}\left(u^{(n)};\boldsymbol{\mu}\right)\varphi_j\ \mathrm{d}x$$

Elements of the N-length vector $\boldsymbol{R}$ are given by: $\boldsymbol{R}_i = -\mathcal{R}(u^{(n)}, \varphi_i; \boldsymbol{\mu})$.

If a finite element approximation space $\mathcal{P}_h \in \mathcal{P}$ is now introduced, with:

$$\mathcal{P}_h = \left\{ \lambda(\boldsymbol{\mu}) = \sum_{i=1}^{p} \lambda^i \xi_i(\boldsymbol{\mu});\ \xi_i \in \mathcal{P}, \lambda^i \in \mathbb{R} \right\} \quad (15)$$

a parameter function $\lambda_i(\boldsymbol{\mu})$ can be approximated as follows : $\lambda_i(\boldsymbol{\mu}) \approx \sum_{j=1}^{p} \lambda_i^j \xi_j(\boldsymbol{\mu}) = \lambda_i^j \xi_j$. By denoted by $[\Phi_i] \in \mathbb{R}^N$ (resp. $[\lambda_i] \in \mathbb{R}^p$) the nodal vector associated with the discretization of space function Φ_i (resp. parameter function λ_i) and by $\boldsymbol{W}_k = [[\Phi_1][\Phi_2]\ldots[\Phi_k]]$ the $N \times k$ matrice of the discretized $k-$order reduced basis, the sum over i of first terms in (14) gives:

$$\left[\left(\underbrace{\boldsymbol{W}_k^T}_{k\times N}\underbrace{\mathbf{A}}_{N\times N}\underbrace{\boldsymbol{W}_k}_{N\times k}\right) \otimes \underbrace{\boldsymbol{M}}_{p\times p}\right] \underbrace{\begin{pmatrix}[\delta\lambda_1]\\ [\delta\lambda_2]\\ \vdots\\ [\delta\lambda_k]\end{pmatrix}}_{pk\times 1} \quad (16)$$

where $\otimes$ denotes the tensor product of matrices (Kronecker product) and $\boldsymbol{M}$ is a $p \times p$ matrice whose entries are defined by $\boldsymbol{M}_{ij} = \int_{\mathcal{D}} \xi_i\ \xi_j\ \mathrm{d}\mu$.

Projection of discretized operator $\mathbf{A}$ onto the discretized reduced-order basis $\boldsymbol{W}_k$ involves two matrice-matrice products resulting into a computational complexity that scales with $\mathcal{O}(Nk)$ (resp. $\mathcal{O}(N^2\ k)$) if $\mathbf{A}$ is sparse (resp. full) to compute $\mathbf{A}\boldsymbol{W}_k$ and $\mathcal{O}(Nk^2)$ for left multiplication by $\boldsymbol{W}_k^T$. Complexity for evaluating tensor product is $\mathcal{O}(pk^2)$ if $\boldsymbol{M}$ is sparse. Thus, computational complexity for computing the $kp \times kp$ matrice $\left(\boldsymbol{W}_k^T\mathbf{A}\boldsymbol{W}_k\right) \otimes \boldsymbol{M}$ is $\mathcal{O}(Nk + Nk^2 + pk^2)$ (resp. $\mathcal{O}(N^2k + Nk^2 + p^2k^2)$) if $\mathbf{A}$ is sparse (resp. full). Since discretized operator $\mathbf{A}$ is parameter-independent, separation of variables is possible to compute the integrals over the space-parameter domain (Kronecker product). Consequently, provided that the ROB does not evolve, the $k \times k$ matrice $\boldsymbol{W}_k^T\mathbf{A}\boldsymbol{W}_k$ can be pre-computed and factorized, if needed, offline. This is the reason why model reduction techniques are particularly efficient and appealing for parametrized time-invariant systems [4] or linear stationary and quasi-stationary systems whose operators are affine functions of the input parameters [16].

On the contrary, the parameter-dependency of the discretized operator $\boldsymbol{G}$ in the second term of (14) prevents the separation of integrals. Separation of the operator as a tensor product is not straightforward. Moreover, it has to be evaluated for each new iterate $u^{(n)}(\boldsymbol{x}, \boldsymbol{\mu})$ at a computational complexity of $\mathcal{O}(pNk^2)$ (resp. $\mathcal{O}(pN^2k)$) if the discretized operator is sparse (resp. full) [15, 35]. One can also show that complexity

for evaluating nonlinear residue (right-hand side member of (14)) is $\mathcal{O}(pNk)$. In short, the expected decrease in computational cost by using reduced-basis approximation is consequently quite modest whatever the dimension reduction $k \ll N$ is.

This bottleneck is clearly not specific to PGD-based model reduction but more generally to reduced-basis approximation in nonlinear model reduction. This problem arises due to the fact that: (i) Galerkin projection onto the ROB has to be performed as soon as the reduced-order basis evolves, (ii) non-linear terms (Jacobian and residue) have to be evaluated at each iteration of the solution strategy for each new iterate which prevents pre-computations of operators. Many works have already been done to tackle this issue as discussed in the introduction.

3. The Reference Point Method

The goal of the Reference Point Method described in this section, is to decrease the computational cost of integrals arising from Galerkin projection. For that two main features are introduced: (i) an approximation framework with the introduction of reference points to simplify the evaluation of integrands, (ii) a low-cost technique to reconstruct by explicit formulas an approximate low-rank separated representation by patch of operators without resorting to SVD-based techniques, which is a great advantage regarding integral computation by separation of variables. The motivations of the proposed technique are first presented in **Section 3.1**. The basics of the Reference Point Method are given in **Section 3.2** before being applied to **Problem 1** in **Section 3.3.1**. A brief computational complexity analysis is proposed in **Section 3.3.2** to estimate the expected gain in computational cost.

3.1. Motivations

Among the techniques introduced to tackle the aforementioned bottleneck, the Empirical Interpolation Method (EIM) [33] is probably one of the most popular. It proposes an inexpensive interpolation procedure of nonlinear terms based on the offline construction of a suitable additional reduced-basis approximation space. More precisely, EIM enables to provide an approximation of nonlinear function g as follows:

$$g(u^{(n)};\boldsymbol{x},\boldsymbol{\mu}) \approx g_M(u^{(n)};\boldsymbol{x},\boldsymbol{\mu}) = \sum_{m=1}^{M} \varphi_m^M(\boldsymbol{\mu})\, q_m(\boldsymbol{x}) \tag{17}$$

where M (with $M \ll N, p$) interpolation "Magic" points $\{\boldsymbol{x}_m^M\}_{1\leqslant m\leqslant M}$ of the space domain and M "Magic" points of the parameter domain $\{\boldsymbol{\mu}_m^M\}_{1\leqslant m\leqslant M}$ are computed by a greedy selection process in order to build the M interpolation functions $\{q_m(\boldsymbol{x})\}_{1\leqslant m\leqslant M}$ during an offline stage. For each iterate $u^{(n)}$, coefficient functions $\{\varphi_m^M(\boldsymbol{\mu})\}_{1\leqslant m\leqslant M}$ have to be updated online for the reduced-order model of $u^{(n)}$ evaluated at interpolation "Magic" points $\{\boldsymbol{x}_m^M\}_{1\leqslant m\leqslant M}$. Note that, according to [15], approximation function g_M is interpolant at the "Magic" spatial points, i.e. $g(u^{(n)};\boldsymbol{x}_m^M,\boldsymbol{\mu}) = g_M(u^{(n)};\boldsymbol{x}_m^M,\boldsymbol{\mu})$ for $1 \leqslant m \leqslant M$.

Let consider **Problem 4** involved at the preliminary step, which is nothing more than an update of the reduced model from the current ROB. Thanks to EIM, tangent operator can be approximated as follows:

$$\mathbf{H}^-(u^{(n)};\boldsymbol{x},\boldsymbol{\mu}) = \left.\frac{\partial g(u;\boldsymbol{\mu})}{\partial u}\right|_{u=u^{(n)}} \approx \mathbf{H}_M^-(u^{(n)};\boldsymbol{x},\boldsymbol{\mu}) = \sum_{m=1}^{M} \bar{\varphi}_m^M(\boldsymbol{\mu})\, q_m(\boldsymbol{x}) \tag{18}$$

where $\bar{\varphi}_m^M$ is the partial derivative of φ_m^M with respect to argument $\boldsymbol{\mu}$. Approximation of residue $\mathcal{R}$ with EIM is done accordingly but not detailed hereafter. Interested reader can refer to [15] for further details. Thus, (4) can be approximated with EIM as follows:

$$1 \leqslant j \leqslant k,\ \forall\, \lambda^* \in \mathcal{P},$$
$$\sum_{i=1}^{k} \int_{\mathcal{D}} \lambda^* \left(a(\Phi_i,\Phi_j) + \sum_{m=1}^{M} \bar{\varphi}_m^M(\boldsymbol{\mu}) \left(\int_{\Omega} \Phi_i\ q_m(\boldsymbol{x})\ \Phi_j\ \mathrm{d}x \right) \right) \delta\lambda_i\ \mathrm{d}\mu = -\int_{\mathcal{D}} \lambda^*\ \mathcal{R}(u^{(n)},\Phi_j;\boldsymbol{\mu})\ \mathrm{d}\mu \tag{19}$$

Provided that the M functions $\{q_m(\boldsymbol{x})\}_{1\leqslant m\leqslant M}$ and reduced-order basis $\mathcal{W}_k = \{\Phi_i\}_{i=1\leqslant i\leqslant k}$ are known, terms $a(\Phi_i, \Phi_j)$ and $\int_\Omega \Phi_i\, q_m(\boldsymbol{x})\, \Phi_j \,\mathrm{d}x$ are parameter-independent and thus can be pre-computed. In this case, it is shown in [15] that the computational complexity at each iteration for the Galerkin projection (resp. the determination of coefficient functions $\{\varphi_m^M(\boldsymbol{\mu})\}_{1\leqslant m\leqslant M}$) is $\mathcal{O}(k^2M)$ (resp. $\mathcal{O}(M^2)$). Since $M, k \ll N$, the online complexity is consequently independent of N, the dimension of the underlying initial finite element approximation space.

EIM could be used to solve the preliminary step, however the fact that the ROB changes quite often throughout the iterations of the LATIN nonlinear solver makes of this technique less suitable. Indeed, for a fixed number M of best/magic interpolation points the error in the coefficient functions $\{\varphi_m^M(\boldsymbol{\mu})\}_{1\leqslant m\leqslant M}$ used to interpolate the nonlinear terms ultimately dominates when the size k of the ROB increases [15]. Adding more magic points may be necessary as long as the ROB evolves. Such an adaptative strategy could be derived but it may be not very efficient in this context, at least in the LATIN framework. Moreover, the determination of "Magic" parameter points $\{\boldsymbol{\mu}_m^M\}_{1\leqslant m\leqslant M}$ may be costly particularly for applications where nonlinear function g is time-dependent as for parabolic parametrized problem [15] or implicitly dependent to the field variable u for nonlinear problem as it is the case here.

The RPM proposes a more pragmatic approach. It is not based on the approximation of the nonlinear function g (i.e. tangent operator $\mathbf{H}^-$). On the contrary, it is rather an approximation technique of the integrals involved in the Galerkin projection similarly to quadrature techniques in classical finite element methods. More precisely, it aims to compute, at low-cost, each contribution to the ROM, i.e. each term/entry α_{ij}:

$$\alpha_{ij} = \int_{\mathcal{D}\times\Omega} \lambda^*(\boldsymbol{\mu})\, \Phi_i(\boldsymbol{x})\, \mathbf{H}^-(u^{(n)}; \boldsymbol{x}, \boldsymbol{\mu})\, \Phi_j(\boldsymbol{x})\, \delta\lambda_i(\boldsymbol{\mu})\, \mathrm{d}x\, \mathrm{d}\mu, \quad 1 \leqslant i, j \leqslant k \tag{20}$$

where repeated indices are not summed over their range. It is worth noting at the fact that RPM is used at each iteration of the LATIN method only to solve the preliminary step. Even though the integral computations are not enough accurate, the PGD-model reduction technique can improve it by adding new correction functions to the ROB throughout the LATIN iterations, which ensures the convergence of the whole process.

3.2. Basics of the RPM

The RPM aims at constructing an approximate low-rank separated representation by patch of integrand in (20). In this section, two main features are introduced: (i) an approximation framework with the introduction of reference points (see **Section 3.2.1**) to simplify the evaluation of integrands (see **Section 3.2.2**), (ii) a low-cost technique to reconstruct by explicit formulas an approximate low-rank separated representation by patch of integrands (see **Section 3.2.3**).

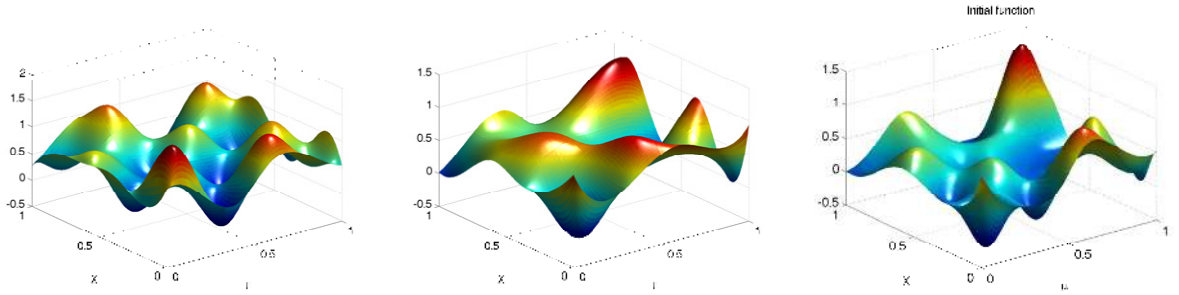
Integrand of (20) involves the product of several operands: the nonlinear function $\mathbf{H}^-(u^{(n)}; \boldsymbol{x}, \boldsymbol{\mu})$ pre and post multiplied by the terms $\lambda^*(\boldsymbol{\mu})\Phi_i(\boldsymbol{x})$ and $\Phi_j(\boldsymbol{x})\delta\lambda_i(\boldsymbol{\mu})$. For the sake of clarity, instead of integrand of (20), let consider the product of two scalar functions depending on two variables. This simple example will be much more convenient to figure the approximation procedure:

$$F(\mu, x) = f(\mu, x)\, f'(\mu, x) \tag{21}$$

where $f(\mu, x)$ and $f'(\mu, x)$ are two scalar functions of two variables, $\mu \in \mathcal{D} = [0, 1]$ and $x \in \Omega = [0, 1]$. An example of functions f and f' and their product F is given in **Figure 2**.

Let assume that f and f' are known under separated representation (even though separation property is not always satisfied for any quantity, like $\mathbf{H}^-$):

$$f(\mu, x) = \sum_{i=1}^{k} \lambda_i(\mu)\Lambda_i(x) \qquad \text{and} \qquad f'(\mu, x) = \sum_{i=1}^{k'} \theta_i(\mu)\Theta_i(x). \tag{22}$$



(a) Scalar function $f(\mu, x)$ (b) Scalar function $f'(\mu, x)$ (c) Product $F(\mu, x) = f(\mu, x)f'(\mu, x)$

Figure 2: Example of two scalar functions depending on two variables and their product

Note that each evaluation of product F for a given value of x and μ involves $2(k+k')-1$ multiplications and additions. If a separated representation of product F is needed – to benefit from integral computation by separation of variables for instance – it would result into a M-term separated representation with $M = k \times k'$. This M-term representation is likely non-optimal and M may increase swiftly if terms are added to f and f' representations (k and k' increase). Performing a singular value decomposition of F may be necessary to achieve a separated representation of F with a reasonable number of terms. RPM follows a different path to avoid the artificial increasing of the number PGD modes. As described in the following, RPM enables one to build an approximate low-rank separated representation by patch of any quantity without resorting to SVD-based techniques.

3.2.1. Compressed format and generalized components

The RPM approximation framework is based on the concept of *reference points* and enables one to define a reduced representation of the data [40]. The parameter domain is split into m_μ sub-intervals $\mathcal{D}_i$ of size $\Delta\mu_i$. The center μ_i of sub-interval $\mathcal{D}_i$ is called *parameter reference point*. For the space domain, m_x points x_j are introduced and the domain Ω is divided into m_x sub-domains Ω_j of size $\Delta\omega_j$. The points x_j are called *spatial reference points*. These reference points are arbitrary and can be chosen independently of discretization spaces for unknown quantities.

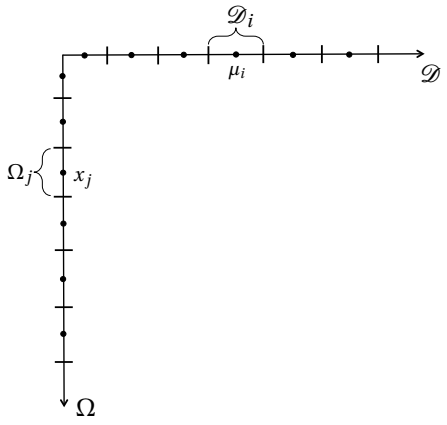


Figure 3: Parameter and spatial reference points for parameter domain $\mathcal{D} = [0, 1]$ and spatial domain $\Omega = [0, 1]$

An influence zone is defined around each reference point of the space-parameter domain defined by the coordinates of the spatial reference points x_j and parameter reference points μ_i. Part of the domain, $\mathcal{D}_i \times \Omega_j$, is called reference patch (i, j). Thus, the entire domain $\mathcal{D} \times \Omega$ is divided into $m_\mu \times m_x$ patches.

A function f defined on the domain $\mathcal{D}\times\Omega$ is represented by its *generalized components*, $\bar{f} := \{(\bar{a}_{ij}, \bar{b}_{ij})\}$, defined as follows. For $1 \leqslant i \leqslant m_\mu$ and $1 \leqslant j \leqslant m_x$:

$$\bar{f} := \left\{ \begin{array}{rcccl|rcccl} \bar{a}_{ij}(\mu) & = & f(\mu, x_j) & \text{if } \mu \in \mathcal{D}_i & | & \bar{b}_{ij}(x) & = & f(\mu_i, x) & \text{if } x \in \Omega_j \\ \bar{a}_{ij}(\mu) & = & 0 & \text{otherwise} & | & \bar{b}_{ij}(x) & = & 0 & \text{otherwise} \end{array} \right\} \tag{23}$$

The generalized components $\{\bar{a}_{ij}(\mu)\}_{i=1,...,m_\mu}$ related to spatial reference point x_j gives the description of function f at spatial point x_j over the entire parameter domain $\mathcal{D}$. Similarly, the generalized components $\{\bar{b}_{ij}(x)\}_{i=1,...,m_x}$ related to parameter value μ_i gives the description of function f at parameter value μ_i over the entire spatial domain Ω. Note that by construction, for $1 \leqslant i \leqslant m_\mu$ and $1 \leqslant j \leqslant m_x$:

$$\bar{a}_{ij}(\mu_i) = \bar{b}_{ij}(x_j) \tag{24}$$

If function f is expressed under separated representation, each pair of functions is described by their generalized components. Hence, for $f(\mu, x) = \sum_{i=1}^{k} \lambda_i(\mu)\Lambda_i(x)$, it yields: for $1 \leqslant i \leqslant m_\mu$ and $1 \leqslant j \leqslant m_x$:

$$\bar{f} := \left\{ \begin{array}{lll|lll} \bar{a}_{ij}(\mu) = \sum_{l=1}^{k} \lambda_l(\mu)\Lambda_l(x_j) & \text{if } \mu \in \mathcal{D}_i & | \; \bar{b}_{ij}(x) = \sum_{l=1}^{k} \lambda_l(\mu_i)\Lambda_l(x) & \text{if } x \in \Omega_j \\ \bar{a}_{ij}(\mu) = 0 & \text{otherwise} & | \; \bar{b}_{ij}(x) = 0 & \text{otherwise} \end{array} \right\} \tag{25}$$

In a sense, generalized components provide a kind of *compressed format* of the quantities. **Figure 4** depicts surfaces defined by functions f and f' and their generalized components when $m_\mu = 10$ and $m_x = 10$. In this case, for a two-dimensional function, a generalized component associated with a spatial or parameter reference point is a quantity defined on a line. For higher dimensions, it corresponds more generally to hyperplanes, hypersurfaces or submanifolds of the domain accordingly. Some patches for functions f and f' are depicted in **Figure 6**. In two dimensions, these patches can be easily represented. In 3D, patches become volumes. For a scalar function of three variables ($f(\mu, \boldsymbol{x})$ with $\boldsymbol{x} = (X, Y)$), a patch (i, j) becomes a parallelepiped, generalized space components $\bar{b}_{ij}(\boldsymbol{x})$ are snapshots of the two-dimensional spatial solution for a given value of parameter μ_i and generalized parameter components $\bar{a}_{ij}(\mu)$ are lines (**Figure 5**).

The algorithm that partitions the domain into patches for an arbitrary number of reference points for each coordinate, can easily takes into account the geometry (holes, corners...) of the spatial domain by moving some spatial reference points where functions are not defined and by slightly adapting the patches according to the space domain geometry (**Figure 7**).

Remark 2. *For a multivariable function $f(\mu^1, ..., \mu^r, x)$, m_l reference points along a given dimension l can be introduced. The domain $\mathcal{D}_1 \times ... \times \mathcal{D}_r \times \Omega$ is divided into $m_{\mu^1} \times ... \times m_{\mu^r} \times m_x$ patches. The definition of a reference patch $(i_{\mu^1}, ..., i_{\mu^r}, j)$ with $1 \leqslant i_{\mu^l} \leqslant m_{\mu^l}$ and $1 \leqslant j \leqslant m_x$ defined on the part of the domain $\mathcal{D}_{i_{\mu^1}} \times ... \times \mathcal{D}_{i_{\mu^r}} \times \Omega_j$ and the definition of the generalized components similarly to* (23) *are straightforward.*

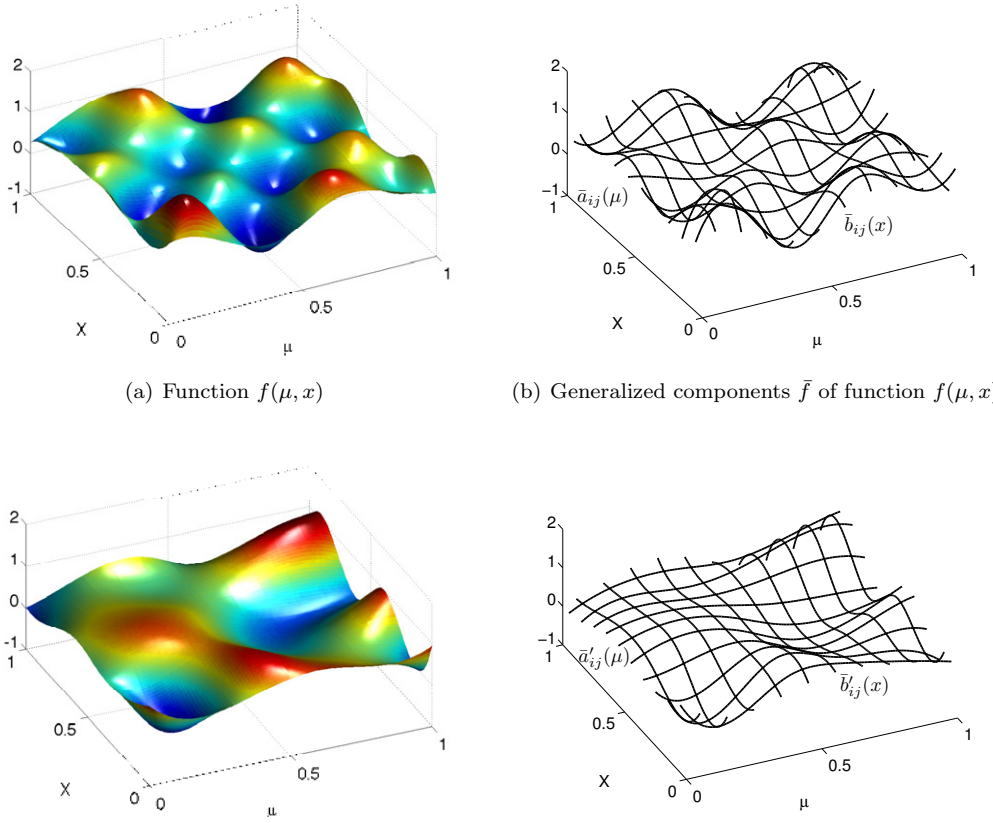
3.2.2. Algebra in the compressed framework

It is straightforward to show that the RPM framework shows interesting properties regarding elementary operations on quantities represented under their compressed formats (see **Table 1**). For instance, **Figure 8**

Addition	$\overline{f \; + \; f'}$	$=$	$\bar{f} \; + \; \bar{f}'$
Multiplication	$\overline{f \; f'}$	$=$	$\bar{f} \; \bar{f}'$
Derivative	$\overline{\partial f/\partial\mu}$	$=$	$\partial\bar{f}/\partial\mu$
Operator	$\overline{H \; f}$	$=$	$\bar{H} \; \bar{f}$

Table 1: Elementary operations in the compressed framework

illustrates the evaluation of the product F of two functions f and f' (see (21)) under their compressed formats. In order to evaluate the product all over the domain, a reconstruction of the product from the generalized components $\bar{F}$ of F has to be proposed as described in the following section.



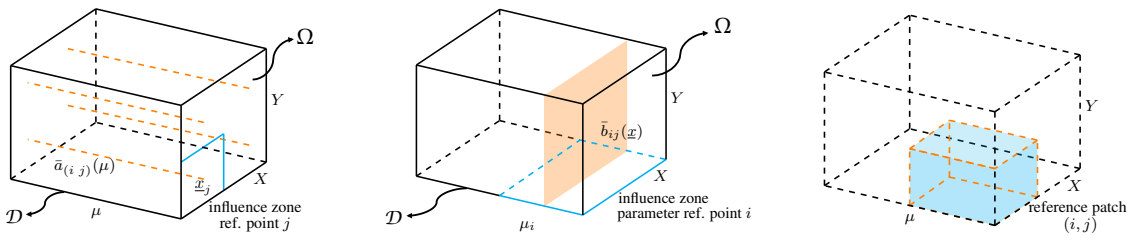
(a) Function $f(\mu, x)$

(b) Generalized components $\bar{f}$ of function $f(\mu, x)$

(c) Function $f'(\mu, x)$

(d) Generalized components $\bar{f}'$ of function $f'(\mu, x)$

Figure 4: Generalized components of functions f and f' for $m_\mu = 10$ and $m_x = 10$



(a) Generalized parameter component $\bar{a}_{ij}(\mu)$ related to a spatial point x_j

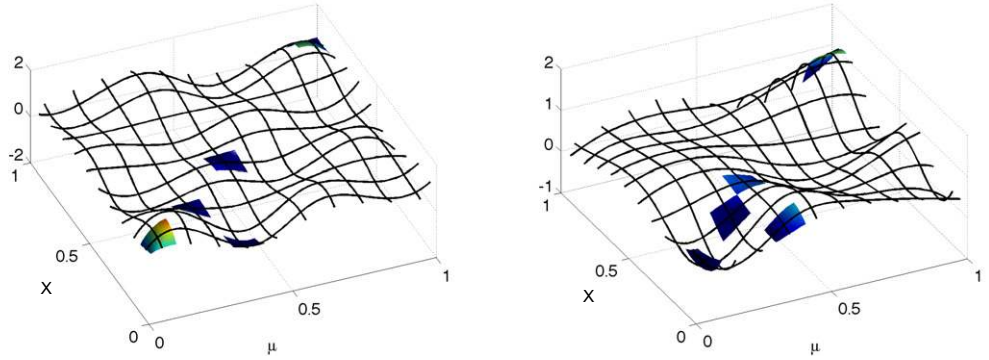
(b) Generalized spatial component $\bar{b}_{ij}(\boldsymbol{x})$ related to parameter value μ_i

(c) Patch (i, j)

Figure 5: Generalized components and patch (i, j) for a scalar function of three variables ($f(\mu, \boldsymbol{x})$ with $\boldsymbol{x} = (X, Y)$)

3.2.3. Reconstruction of a low-rank separated representation by patch

In this section, the reconstruction procedure of a rank-one separated representation by patch of F is shown. Denoted by $\bar{\bar{F}}$, this reconstruction is obtained from the compressed format $\bar{F}$ by generating one



(a) Some patches for function $f(\mu, x)$ (b) Some patches for function $f'(\mu, x)$

Figure 6: Some of the space-parameter patches $\mathcal{D}_i \times \Omega_j$ of functions f and f' for $m_\mu = 10$ and $m_x = 10$

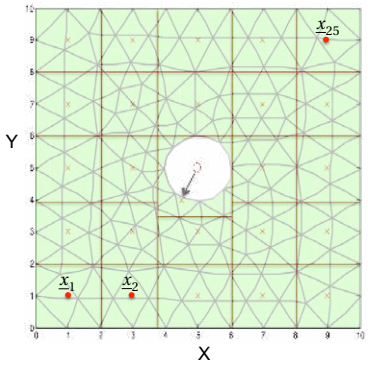


Figure 7: Correction of a spatial reference point located in a hole for a two-dimensional spatial domain and adaptation of the patches

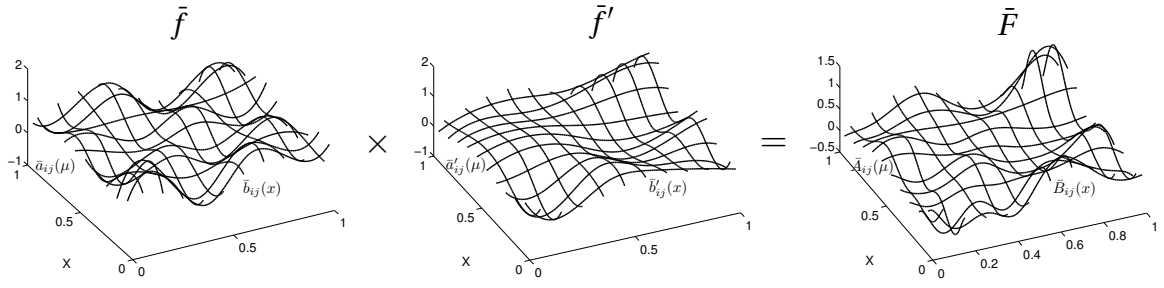


Figure 8: Compressed format $\bar{F} = \bar{f}\,\bar{f}'$ of product $F = f\,f'$

product of functions per space-parameter patch $\mathcal{D}_i \times \Omega_j$: for $1 \leqslant i \leqslant m_\mu$ and $1 \leqslant j \leqslant m_x$:

$$\forall (\mu, x) \in \mathcal{D}_i \times \Omega_j, \quad F(\mu, x) \approx \bar{\bar{F}}(\mu, x) = a_{ij}(\mu)\, b_{ij}(x) \tag{26}$$

where products of functions $a_{ij}(\mu)b_{ij}(x)$ defined on each patch $\mathcal{D}_i \times \Omega_j$ are determined from the generalized components $\{(\bar{A}_{ij}(\mu), \bar{B}_{ij}(x))\}$ of $\bar{F}$ thanks to the minimization of the following functional:

$$1 \leqslant i \leqslant m_\mu,$$

$$J(\{(a_{ij}, b_{ij})\}_{1 \leqslant j \leqslant m_x}) = \sum_{k=1}^{m_x} \left[\Delta\omega_k \| (\bar{A}_{ik}(\mu) - a_{ik}(\mu) b_{ik}(x_k)) \ \lambda_{ik} \|^2_{\mathcal{D}_i} + \Delta\mu_i \| \bar{B}_{ik}(x) - a_{ik}(\mu_i) b_{ik}(x) \|^2_{\Omega_k} \right] \quad (27)$$

where $\| \cdot \|_{\mathcal{D}_i}$ and $\| \cdot \|_{\Omega_j}$ are the classical $L^2(\mathcal{D}_i)$ and $L^2(\Omega_j)$ norms. Term λ_{ik} is an influence coefficient which gives more importance to the neighboring patches of patch $\mathcal{D}_i \times \Omega_k$ along space coordinate. This functional measures the distance between the patch reconstruction and the generalized components. Minimization of functional (27) (see **Appendix C**) leads to the following explicit formulas: for $1 \leqslant i \leqslant m_\mu$ and $1 \leqslant j \leqslant m_x$:

$$a_{ij}(\mu) = \frac{\sum_{k=1}^{m_x} \Delta\omega_k \ \bar{A}_{ik}(\mu) \ \bar{A}_{ik}(\mu_i) \ \lambda_{ik}^2}{\sum_{k=1}^{m_x} \Delta\omega_k \ \bar{A}_{ik}(\mu_i)^2 \ \lambda_{ik}^2} \qquad \text{and} \qquad b_{ij}(x) = \bar{B}_{ij}(x) \quad (28)$$

It can be noticed that space dimension and parameter dimension are not equally treated. Indeed, space domain is favored by holding all the information arising from the spatial generalized component $\bar{B}_{ij}(x)$. As illustrated on **Figure 9**, one can also see that: for $1 \leqslant i \leqslant m_\mu$ and $1 \leqslant j \leqslant m_x$:

$$\forall x \in \Omega, \quad \bar{\bar{F}}(x, \mu_i) \ = \ \bar{B}_{ij}(x)$$

This choice is suitable for structural mechanics where the spatial gradients of quantities are usually stronger than their variations in parameter (or time) variable. Influence coefficients λ_{ik} have an influence on parameter function $a_{ij}(\mu)$ by weighting the values $\bar{A}_{ik}(\mu_i)$ ($= \bar{B}_{ik}(x_k)$ due to (24)) at the center of patches along space coordinate.

If one chooses $\lambda_{ik} = 1$ for $1 \leqslant k \leqslant m_x$ (uniform value), one obtains: for $1 \leqslant i \leqslant m_\mu$ and $1 \leqslant j \leqslant m_x$:

$$a_{ij}(\mu) = \frac{\sum_{k=1}^{m_x} \Delta\omega_k \ \bar{A}_{ik}(\mu) \ \bar{A}_{ik}(\mu_i)}{\sum_{k=1}^{m_x} \Delta\omega_k \ \bar{A}_{ik}(\mu_i)^2} = \frac{\sum_{k=1}^{m_x} \Delta\omega_k \ \bar{A}_{ik}(\mu) \ \bar{B}_{ik}(x_k)}{\sum_{k=1}^{m_x} \Delta\omega_k \ \bar{B}_{ik}(x_k)^2} \qquad \text{and} \qquad b_{ij}(x) = \bar{B}_{ij}(x) \quad (29)$$

In this case, parameter functions $a_{ij}(\mu)$ are the same for any subdomain Ω_j for a given i. Reconstruction $\bar{\bar{F}}$ is continuous not only for $\mu = \mu_i$ but also for any $\mu \in \mathcal{D}_i$ with $1 \leqslant i \leqslant m_\mu$ (see **Figure 9(c)**). This may lead to high discontinuities between sub-intervals $\mathcal{D}_i$ if variations of product according to parameter variable are significant.

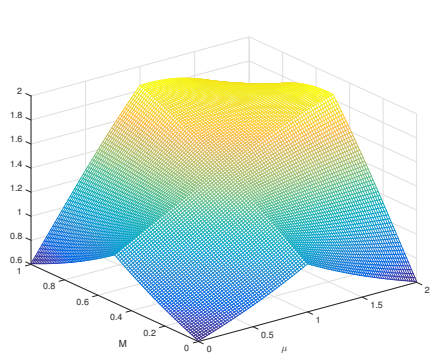
Another limit choice is to consider influence functions such that $\lambda_{ik} = 1$ only for $k = j$. This leads to minimizations by patch independent of each other and:

$$a_{ij}(\mu) = \frac{\bar{A}_{ij}(\mu)}{\bar{A}_{ij}(\mu_i)} = \frac{\bar{A}_{ij}(\mu)}{\bar{B}_{ij}(x_j)} \qquad \text{and} \qquad b_{ij}(x) = \bar{B}_{ij}(x) \quad (30)$$

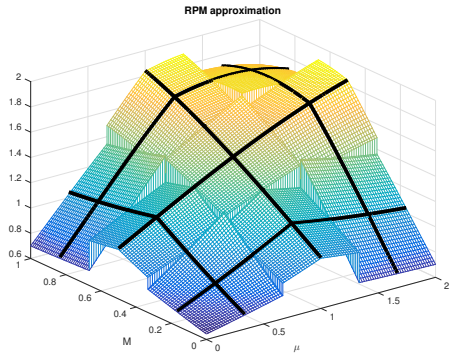
Parameter function $a_{ij}(\mu)$ is normalized by the value $\bar{A}_{ij}(\mu_i) = \bar{B}_{ij}(x_j)$ (due to (24)) at the center of patch $\mathcal{D}_i \times \Omega_j$ and is consequently independent from one patch to another. Note that in this case, denominator of a_{ij} can be null and can lead to numerical difficulties. This may also happen for (28) and (29) but is by far less probable. In this situation, adding an appropriate constant value to generalized components before reconstruction and subtracting the same value subsequently to the reconstruction can easily palliate this issue. Reconstruction $\bar{\bar{F}}$ is discontinuous between sub-cells Ω_j for all $\mu \in \mathcal{D}_i$ except for $\mu = \mu_i$ with $1 \leqslant i \leqslant m_\mu$ (see **Figure 9(b)**).

An optimization of weight functions λ_{ik} with compact support along spatial coordinate between the two extreme situations leading to (29) (uniform weight) and (30) (independent minimizations by patch) has to be done according to the variations of the function F which is approximated by the RPM (see **Figure 9(d)**). For all the numerical examples presented hereafter, one chose for (28):

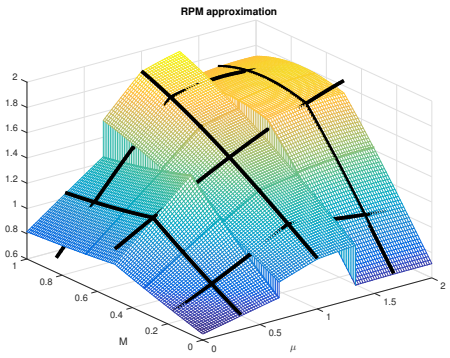
$$\lambda_{ik}^2 = \begin{cases} 1 & \text{if } k = j \\ 0.1 & \text{for } k = j-1 \text{ and } k = j+1 \\ 0 & \text{otherwise} \end{cases} \quad (31)$$



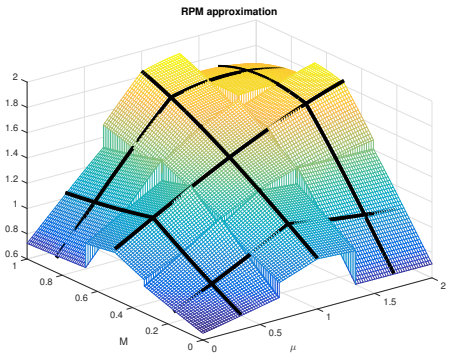
(a) Function F_1



(b) Independent minimizations by patch, $\lambda_{ik} = \delta_{ik}$ ($e = 2.08\%$)



(c) Uniform value, $\lambda_{ik} = 1$ ($e = 3.18\%$)



(d) Optimized value given by (31) ($e = 2.00\%$)

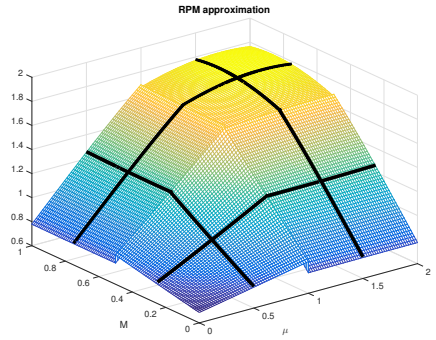
Figure 9: The function $F_1(x,\mu) = e^{-|(x-0.5)(\mu-1)|} + \sin(x\mu)$ and its approximation $\bar{\bar{F}}_1$ for $m_\mu = 3$ parameters and $m_x = 3$ spatial reference points and different choices of influence functions λ_{ik}

This optimized value of λ_{ik} has been obtained by numerical experiments and is suitable for most of the functions that have been tested in the present works. **Figure 10** shows different approximations $\bar{\bar{F}}_1$ of function $F_1(x,\mu) = e^{-|(x-0.5)(\mu-1)|} + \sin(x\mu)$ for increasing numbers of reference points and the optimized value of λ_{ik} given by (31).

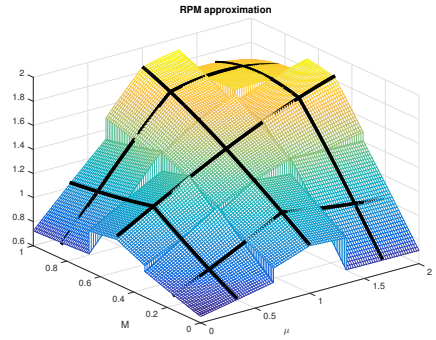
The reconstruction of product F (see **Figure 2**) thanks to its generalized components depicted in **Figure 4** is shown in **Figure 11** for an arbitrary regular grid of $m_\mu = 10$ parameter reference points and $m_x = 10$ spatial reference points. This leads to an error of $e = 5\%$ with respect to the exact function F. Error e is defined as follows:

$$e = \frac{\|F - \bar{\bar{F}}\|_{\mathcal{D}\times\Omega}}{\|\bar{\bar{F}}\|_{\mathcal{D}\times\Omega}} \quad \text{with} \quad \|F\|^2_{\mathcal{D}\times\Omega} = \int_{\mathcal{D}\times\Omega} F^2 \, dx \, d\mu = \sum_{i=1}^{m_\mu}\sum_{j=1}^{m_x} \int_{\mathcal{D}_i\times\Omega_j} F^2 \, dx \, d\mu \tag{32}$$

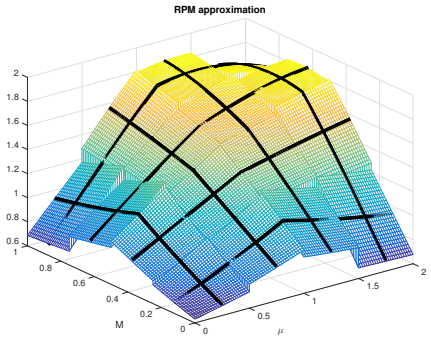
Remark 3. *Reconstruction $\bar{\bar{F}}$ of a separated representation by patch for a multivariable function $F(\mu^1, ..., \mu^r, x)$ needs further investigations. With notations of **Remark 2** and by introducing the $L^2(\mathcal{D}_{i_{\mu^l}})$ norm, $\|\cdot\|_{\mathcal{D}_{i_{\mu^l}}}$, for each sub-interval $\mathcal{D}_{i_{\mu^l}}$ of size $\Delta\mu^l_i$ associated with dimension l and variable μ^l, a functional similar to (27) can easily be defined. A definition that favors space domain (i.e. continuity in space variable x)*



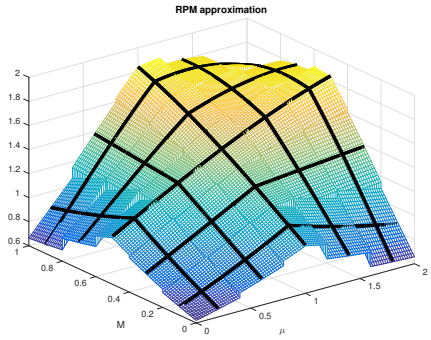
(a) $m_\mu = m_x = 2$ ($e = 4.26\%$)



(b) $m_\mu = m_x = 3$ ($e = 2.00\%$)

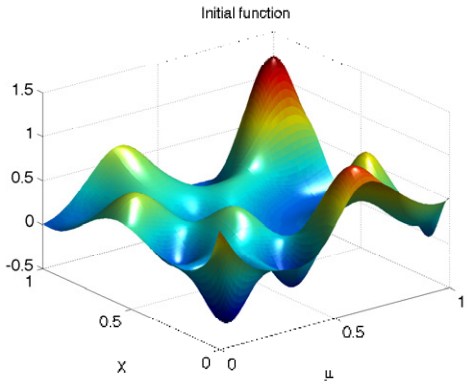


(c) $m_\mu = m_x = 4$ ($e = 1.24\%$)

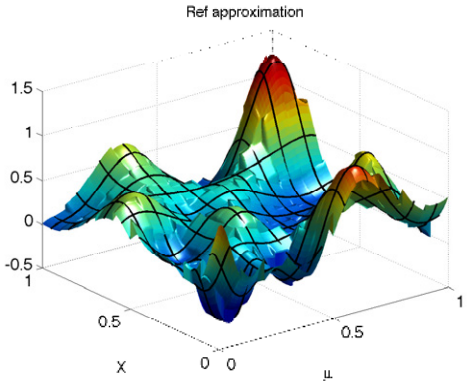


(d) $m_\mu = m_x = 5$ ($e = 0.77\%$)

Figure 10: Approximations $\bar{\bar{F}}_1$ of F_1 for increasing numbers of reference points and the optimized value of λ_{ik} given by (31)



(a) Function F



(b) RPM Reconstruction $\bar{\bar{F}}$ ($e = 5\%$)

Figure 11: The function F and its approximation $\bar{\bar{F}}$ by $m_\mu = 10$ parameter and $m_x = 10$ spatial reference points

and weak continuity from one patch to another according to other variables thanks to the introduction of influence/weight coefficients $\lambda_{i_{\mu^l}k}$ can similarly be considered. In this multivariable case, the choice of these coefficients is not obvious. An easy and straightforward choice is to consider the independent minimization by patch by considering that $\lambda_{i_{\mu^l}k} = \delta_{i_{\mu^l}k}$, leading to a simple explicit reconstruction similar to (30).

3.3. Integral computation by RPM

Once the low-rank separated representation by patch $\bar{\bar{F}}$ defined by (26) has been reconstructed to approximate function F thanks to explicit formulas (28), the integral of F over the space-parameter domain can be approximated as follows:

$$\int_{\mathcal{D}\times\Omega} F(\mu,x)\,dx\,d\mu = \sum_{i=1}^{m_\mu}\sum_{j=1}^{m_x}\int_{\mathcal{D}_i\times\Omega_j} F(\mu,x)\,dx\,d\mu \approx \sum_{i=1}^{m_\mu}\sum_{j=1}^{m_x}\int_{\mathcal{D}_i\times\Omega_j} a_{ij}(\mu)\,b_{ij}(x)\,dx\,d\mu \tag{33}$$

Similarly, RPM can now be used to approximate each contribution to the ROM, α_{ij} defined by (20) and involved in the Galerkin projection of the preliminary step (see **Problem 4**).

3.3.1. Application to the reference problem

By introducing approximation space $\mathcal{P}_h \in \mathcal{P}$ defined by (15) into contribution α_{ij} given by (20), one gets:

$$\alpha_{ij} = \int_{\mathcal{D}\times\Omega} \lambda^*(\boldsymbol{\mu})\,\Phi_i(\boldsymbol{x})\,\mathbf{H}^-(u^{(n)};\boldsymbol{x},\boldsymbol{\mu})\,\Phi_j(\boldsymbol{x})\,\delta\lambda_i(\boldsymbol{\mu})\;\mathrm{d}x\;\mathrm{d}\mu \tag{34}$$

$$\approx \int_{\mathcal{D}\times\Omega} \left(\sum_{r=1}^{p}\lambda^{*r}\xi_r(\boldsymbol{\mu})\right)\,\Phi_i(\boldsymbol{x})\,\mathbf{H}^-(u^{(n)};\boldsymbol{x},\boldsymbol{\mu})\,\Phi_j(\boldsymbol{x})\,\left(\sum_{s=1}^{p}\delta\lambda_i^s\xi_s(\boldsymbol{\mu})\right)\;\mathrm{d}x\;\mathrm{d}\mu \tag{35}$$

$$\approx \sum_{r=1}^{p}\sum_{s=1}^{p}\lambda^{*r}\left(\int_{\mathcal{D}\times\Omega}\underbrace{\xi_r(\boldsymbol{\mu})\,\Phi_i(\boldsymbol{x})\,\mathbf{H}^-(u^{(n)};\boldsymbol{x},\boldsymbol{\mu})\,\Phi_j(\boldsymbol{x})\,\xi_s(\boldsymbol{\mu})}_{\text{Integrand }\omega_{rsij}(\boldsymbol{\mu},\boldsymbol{x})}\;\mathrm{d}x\;\mathrm{d}\mu\right)\delta\lambda_i^s \tag{36}$$

where repeated indice i is not summed over its range.

Let split the two-dimensional parameter domain $\mathcal{D}$ into m_μ sub-domains $\mathcal{D}_i$. The center $\boldsymbol{\mu}_i$ of each sub-domain $\mathcal{D}_i$ is considered as a parameter reference point. Similarly, the space domain Ω is divided into m_x sub-domains Ω_j whose centers $\boldsymbol{x}_j$ defined m_x spatial reference points. Spatial and parameter reference points $\boldsymbol{x}_j$ and $\boldsymbol{\mu}_i$ can be chosen arbitrary on a regular grid for instance. In this case, a given reference patch (i,j) that occupies the part $\mathcal{D}_i\times\Omega_j$ of the four-dimensional space-parameter domain $\mathcal{D}\times\Omega$ is complicated to figure.

Generalized components of integrand ω in (36) (where subscripts $rsij$ have been omitted to simplify the notations) are denoted by $\bar{\omega} := \{(\bar{\omega}^\mu_{ab},\bar{\omega}^x_{ab})\}$ with: for $1\leqslant a\leqslant m_\mu$ and $1\leqslant b\leqslant m_x$:

$$\bar{\omega} := \left\{\begin{array}{rcccl|rcccl} \bar{\omega}^\mu_{ab}(\boldsymbol{\mu}) & = & \omega(\boldsymbol{\mu},\boldsymbol{x}_b) & \text{if } \boldsymbol{\mu}\in\mathcal{D}_a & | & \bar{\omega}^x_{ab}(\boldsymbol{x}) & = & \omega(\boldsymbol{\mu}_a,\boldsymbol{x}) & \text{if } \boldsymbol{x}\in\Omega_b \\ \bar{\omega}^\mu_{ab}(\boldsymbol{\mu}) & = & 0 & \text{otherwise} & | & \bar{\omega}^x_{ab}(\boldsymbol{x}) & = & 0 & \text{otherwise} \end{array}\right\} \tag{37}$$

The approximation $\bar{\bar{\omega}}$ of integrand ω is defined by the following rank-one separated representation by patch: for $1\leqslant a\leqslant m_\mu$ and $1\leqslant b\leqslant m_x$:

$$\forall(\boldsymbol{\mu},\boldsymbol{x})\in\mathcal{D}_a\times\Omega_b,\quad \omega(\boldsymbol{\mu},\boldsymbol{x})\approx\bar{\bar{\omega}}(\boldsymbol{\mu},\boldsymbol{x}) = \bar{\bar{\omega}}^\mu_{ab}(\boldsymbol{\mu})\,\bar{\bar{\omega}}^x_{ab}(\boldsymbol{x}) \tag{38}$$

where functions $\bar{\bar{\omega}}^\mu_{ab}(\boldsymbol{\mu})$ and $\bar{\bar{\omega}}^x_{ab}(\boldsymbol{x})$ are defined by explicit formulas defined in **Section 3.2.3**. To alleviate the notations in the following, let assume that these functions are obtained by explicit formulas (30) (RPM with independent minimizations by patch) instead of general formulas (28) such that:

$$\bar{\bar{\omega}}^\mu_{ab}(\boldsymbol{\mu}) = \frac{\bar{\omega}^\mu_{ab}(\boldsymbol{\mu})}{\bar{\omega}^\mu_{ab}(\boldsymbol{\mu}_a)} = \frac{\bar{\omega}^\mu_{ab}(\boldsymbol{\mu})}{\bar{\omega}^x_{ab}(\boldsymbol{x}_b)} \qquad \text{and} \qquad \bar{\bar{\omega}}^x_{ab}(\boldsymbol{x}) = \bar{\omega}^x_{ab}(\boldsymbol{x}) \tag{39}$$

In this case, definition of integrand ω in (36) leads to: for $1\leqslant a\leqslant m_\mu$ and $1\leqslant b\leqslant m_x$: $\forall(\boldsymbol{\mu},\boldsymbol{x})\in\mathcal{D}_a\times\Omega_b$,

$$\omega(\boldsymbol{\mu},\boldsymbol{x}) \approx \bar{\bar{\omega}}^\mu_{ab}(\boldsymbol{\mu})\,\bar{\bar{\omega}}^x_{ab}(\boldsymbol{x}) \tag{40}$$

$$\approx \frac{1}{\mathbf{H}^-(u^{(n)};\boldsymbol{x}_b,\boldsymbol{\mu}_a)}\Big(\xi_r(\boldsymbol{\mu})\,\mathbf{H}^-(u^{(n)};\boldsymbol{x}_b,\boldsymbol{\mu})\;\xi_s(\boldsymbol{\mu})\Big)\Big(\Phi_i(\boldsymbol{x})\,\mathbf{H}^-(u^{(n)};\boldsymbol{x},\boldsymbol{\mu}_a)\,\Phi_j(\boldsymbol{x})\Big) \tag{41}$$

The integral computation of ω over patch (a, b) by separation of variables can now be done as follows:

$$\int_{\mathcal{D}_a \times \Omega_b} \omega \, dx \, d\mu \approx \ldots$$
$$\ldots \frac{1}{\mathbf{H}^-(u^{(n)}; \boldsymbol{x}_b, \boldsymbol{\mu}_a)} \underbrace{\left(\int_{\mathcal{D}_a} \xi_r(\boldsymbol{\mu}) \, \mathbf{H}^-(u^{(n)}; \boldsymbol{x}_b, \boldsymbol{\mu}) \; \xi_s(\boldsymbol{\mu}) \, d\mu \right)}_{= \left(\overline{\boldsymbol{M}}_a(\boldsymbol{x}_b) \right)_{rs}} \underbrace{\left(\int_{\Omega_b} \Phi_i(\boldsymbol{x}) \, \mathbf{H}^-(u^{(n)}; \boldsymbol{x}, \boldsymbol{\mu}_a) \, \Phi_j(\boldsymbol{x}) \, dx \right)}_{\approx [\Phi_i]_b^T \, \overline{\boldsymbol{K}}_b(\boldsymbol{\mu}_a) [\Phi_j]_b} \quad (42)$$

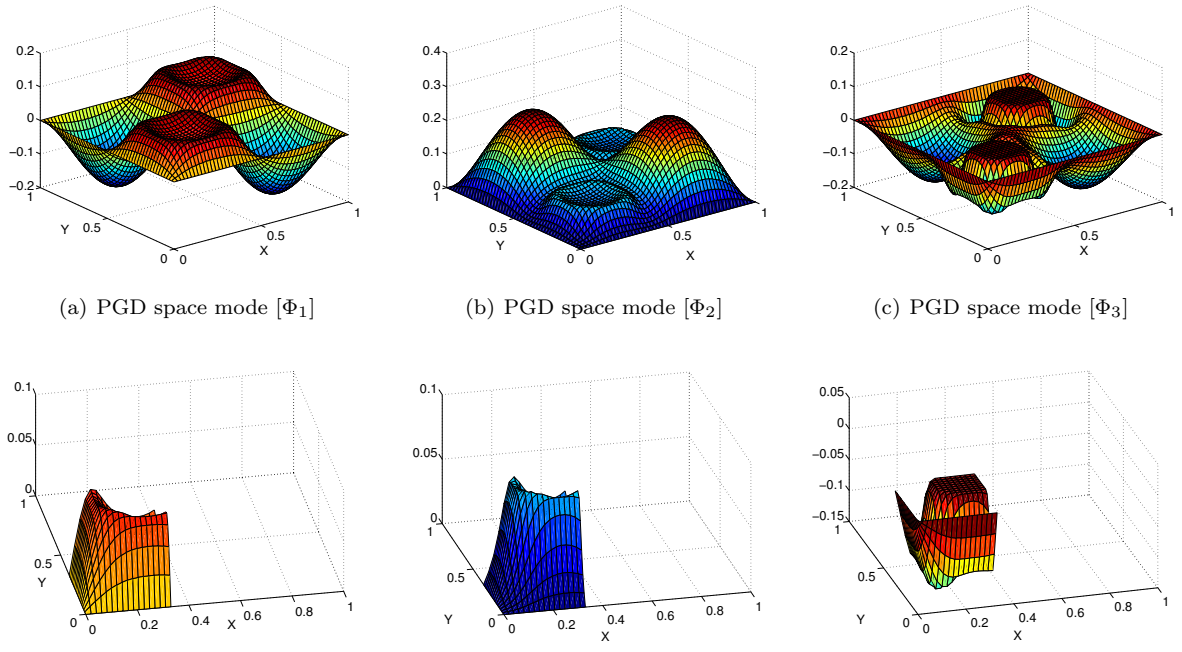
where approximation space $\mathcal{V}_h \in \mathcal{V}$ defined by (12) has been introduced to approximate the second integral on Ω_b. The nodal vector associated with the discretization of space function Φ_i ($\Phi_i(\boldsymbol{x}) \approx \sum_{j=1}^{N} \Phi_i^j \varphi_j(\boldsymbol{x}) = \Phi_i^j \varphi_j$) restricted to space sub-domain Ω_b is denoted by $[\Phi_i]_b$. Similarly, matrice $\overline{\boldsymbol{K}}_b(\boldsymbol{\mu}_a)$ is obtained by the assembly of spatial finite elements contributions located in sub-domain Ω_b and whose entries are defined as follows:

$$\left(\overline{\boldsymbol{K}}_b(\boldsymbol{\mu}_a) \right)_{ij} = \int_{\Omega_b} \varphi_i(\boldsymbol{x}) \; \mathbf{H}^- \left(u^{(n)}; \boldsymbol{x}, \boldsymbol{\mu}_a \right) \varphi_j(\boldsymbol{x}) \; \mathrm{d}x \quad (43)$$

Let also introduce the restriction of the discretized k-order reduced-basis to sub-domain Ω_b:

$$\boldsymbol{W}_{k,b} = [[\Phi_1]_b [\Phi_2]_b \ldots [\Phi_k]_b] \quad (44)$$

Figure 12 depicts the restriction of a discretized 3-order reduced-basis to a sub-domain Ω_b for a 3×3 regular grid of spatial reference points ($m_x = 9$). Thanks to (42), contribution α_{ij} given by (36) can be



(a) PGD space mode $[\Phi_1]$ (b) PGD space mode $[\Phi_2]$ (c) PGD space mode $[\Phi_3]$

(d) Restriction $[\Phi_1]_b$ of mode $[\Phi_1]$ (e) Restriction $[\Phi_2]_b$ of mode $[\Phi_2]$ (f) Restriction $[\Phi_3]_b$ of mode $[\Phi_3]$

Figure 12: Restriction of a discretized 3-order reduced-basis to a sub-domain Ω_b for a 3×3 regular grid of spatial reference points ($m_x = 9$)

approximated as follows:

$$\alpha_{ij} \approx \sum_{r=1}^{p}\sum_{s=1}^{p} \lambda^{*r} \left(\sum_{a=1}^{m_\mu}\sum_{b=1}^{m_x} \int_{\mathcal{D}_a\times\Omega_b} \omega_{rsij}\, dx\, d\mu \right) \delta\lambda_i^s \tag{45}$$

$$\approx \sum_{r=1}^{p}\sum_{s=1}^{p} \lambda^{*r} \left(\sum_{a=1}^{m_\mu}\sum_{b=1}^{m_x} \frac{1}{\mathbf{H}^-(u^{(n)};\boldsymbol{x}_b,\boldsymbol{\mu}_a)} \left(\overline{\boldsymbol{M}}_a(\boldsymbol{x}_b)\right)_{rs} \left([\Phi_i]_b^T\, \overline{\boldsymbol{K}}_b\left(\boldsymbol{\mu}_a\right)[\Phi_j]_b\right) \right) \delta\lambda_i^s \tag{46}$$

Similarly to (16), one deduces that the contribution to preliminary step of the sum over i of second terms in left-hand side of (4) (or (14)) gives:

$$\underbrace{\mathop{\mathrm{A}}_{Patches\ (a,b)} \left[\frac{1}{\mathbf{H}^-(u^{(n)};\boldsymbol{x}_b,\boldsymbol{\mu}_a)} \left(\underbrace{\boldsymbol{W}_{k,b}^T}_{k\times N_b} \underbrace{\overline{\boldsymbol{K}}_b\left(\boldsymbol{\mu}_a\right)}_{N_b\times N_b} \underbrace{\boldsymbol{W}_{k,b}}_{N_b\times k} \right) \otimes \underbrace{\overline{\boldsymbol{M}}_a(\boldsymbol{x}_b)}_{p_a\times p_a} \right]}_{pk\times pk} \underbrace{\begin{pmatrix} [\delta\lambda_1] \\ [\delta\lambda_2] \\ \vdots \\ [\delta\lambda_k] \end{pmatrix}}_{pk\times 1} \tag{47}$$

where the symbol A denotes the assembly of the $m_\mu m_x$ tensor products of matrices by patch (a,b). It is worth noting at the fact that matrices $\overline{\boldsymbol{K}}_b\left(\boldsymbol{\mu}_a\right)$ (resp. $\overline{\boldsymbol{M}}_a(\boldsymbol{x}_b)$) are computed only at the m_μ parameter reference points $\boldsymbol{\mu}_a$ (resp. m_x spatial reference points $\boldsymbol{x}_b$). They have to be updated each time that the LATIN search direction $\mathbf{H}^-$ is updated. If $\mathbf{H}^-$ is not updated and if, in addition, the discretized reduced-basis $\boldsymbol{W}_k$ remains unchanged (k is unchanged), the projection onto the reduced-basis $\boldsymbol{W}_k$ and tensor products on each patch can be pre-computed accordingly.

3.3.2. Computational complexity analysis

As noted in **Section 2.3**, RPM could also be used to approximate the nonlinear residue (right-hand side member of (4)) whose complexity is $\mathcal{O}(pNk)$ for each new iterate. In this section, only the gain in complexity associated with the evaluation of second terms in left-hand side of (4) (or (14)) is discussed. Let recall that the parameter-dependency of the search direction $\mathbf{H}^-$ prevents the separation of integrals which leads to a computational complexity of $\mathcal{O}(pNk^2)$ (resp. $\mathcal{O}(pN^2k)$) if the discretized operator is sparse (resp. full) for each new iterate $u^{(n)}(\boldsymbol{x},\boldsymbol{\mu})$. As shown in **Section 3.3.1**, RPM enables one to provide an approximation of this term as an assembly of tensor products of matrices by patch (see (47)).

By denoting by N_b (resp. p_a) the number of spatial (resp. parameter) unknowns contains within sub-domain Ω_b (resp. sub-domain $\mathcal{D}_a$) in (47), the projection onto the restriction of the discretized k-order reduced-basis to sub-domain Ω_b, $\boldsymbol{W}_{k,b}$, involves two matrice-matrice products resulting into a computational complexity that scales with $\mathcal{O}(N_bk + N_bk^2)$ if $\overline{\boldsymbol{K}}_b\left(\boldsymbol{\mu}_a\right)$ is sparse. Complexity for evaluating tensor product is $\mathcal{O}(p_ak^2)$ if $\overline{\boldsymbol{M}}_a(\boldsymbol{x}_b)$ is sparse. If the number of arithmetical operations for evaluating one entry of matrice $\overline{\boldsymbol{K}}_b\left(\boldsymbol{\mu}_a\right)$ (resp. $\overline{\boldsymbol{M}}_a(\boldsymbol{x}_b)$) is supposed to be β (resp. γ), complexity for evaluating one patch contribution is:

$$\mathcal{O}(\ \underbrace{N_b\,k + N_b\,k^2}_{\text{Projection on } \boldsymbol{W}_{k,b}} + \underbrace{N_b\,\beta}_{\text{Evaluation of } \overline{\boldsymbol{K}}_b\left(\boldsymbol{\mu}_a\right)} + \underbrace{p_a\,k^2}_{\text{Tensor product}} + \underbrace{p_a\,\gamma}_{\text{Evaluation of } \overline{\boldsymbol{M}}_a(\boldsymbol{x}_b)}\ ) \tag{48}$$

Cost for the RPM reconstruction is negligible. Assuming that $N_b \approx \dfrac{N}{m_x}$ and $p_a \approx \dfrac{p}{m_\mu}$, the total complexity for evaluating the contributions of all the patches to the $pk \times pk$ matrice in (47) is roughly:

$$\mathcal{O}\left(\left(\frac{N}{m_x}k + \frac{N}{m_x}k^2 + \frac{N}{m_x}\beta + \frac{p}{m_\mu}k^2 + \frac{p}{m_\mu}\gamma\right) m_x\, m_\mu\right) \sim \mathcal{O}\left(N\, m_\mu\, k^2 + p\, m_x\, k^2\right) \tag{49}$$

if $\beta \sim \mathcal{O}(1)$ and $\gamma \sim \mathcal{O}(1)$. The total gain in case of sparse operators is consequently about:

$$\mathcal{O}\left(\left(\frac{N\, m_\mu\, k^2 + p\, m_x\, k^2}{p\, N\, k^2}\right)^{-1}\right) \sim \mathcal{O}\left(\left(\frac{m_\mu}{p} + \frac{m_x}{N}\right)^{-1}\right) \tag{50}$$

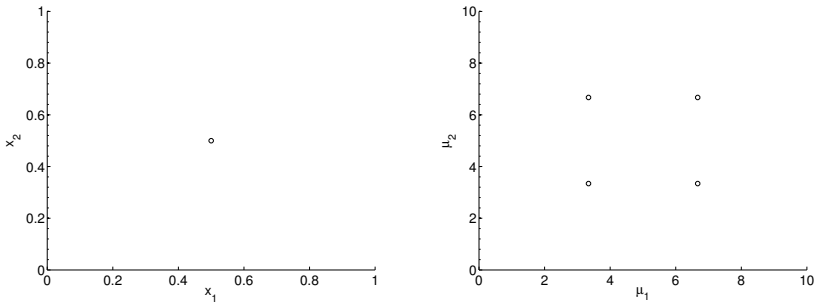
In practice for **Problem 3**, the number of spatial reference points m_x is negligible compared with N and one has $\frac{N}{m_x} \gg \frac{p}{m_\mu}$. For instance, if one chooses for **Problem 3**, $N = 2500$, $p = 225$ and $m_x = m_\mu = 10$, the gain is about one order of magnitude: $\mathcal{O}\left(\frac{p}{m_\mu}\right) \sim 10$.

4. Numerical example

In this section, the numerical simulation of **Problem 1** is investigated. The RPM is here used to approximate the preliminary step of the LATIN-PGD computational strategy for **Problem 3**. In the following this strategy is denoted by LATIN-PGD-RPM. The computational gain according to the number of reference points is investigated in **Section 4.2** and compared to the expected gain (see (50)). Finally, a comparison between the LATIN-PGD-RPM and the EIM combined with reduced-order basis techniques and Newton-based solution strategy [15], called RB-EIM, is proposed in **Section 4.3**.

4.1. On the choice of the reference points

Problem 1 is defined on the space-parameter domain $\mathcal{D} \times \Omega$ with $\mathcal{D} = [0.01, 10]^2 \subset \mathbb{R}^2$ and $\Omega =]0, 1[^2 \subset \mathbb{R}^2$. A spatial (resp. parameter) point is defined by its components $\boldsymbol{x} = (x_1, x_2) \in \Omega =]0, 1[^2$ (resp. $\boldsymbol{\mu} = (\mu_1, \mu_2) \in \mathcal{D} = [0.01, 10]^2$). **Figure 13** depicts the situation where $m_x = 1 \times 1$ spatial and $m_\mu = 2 \times 2$ parameter reference points are chosen on a regular grid.



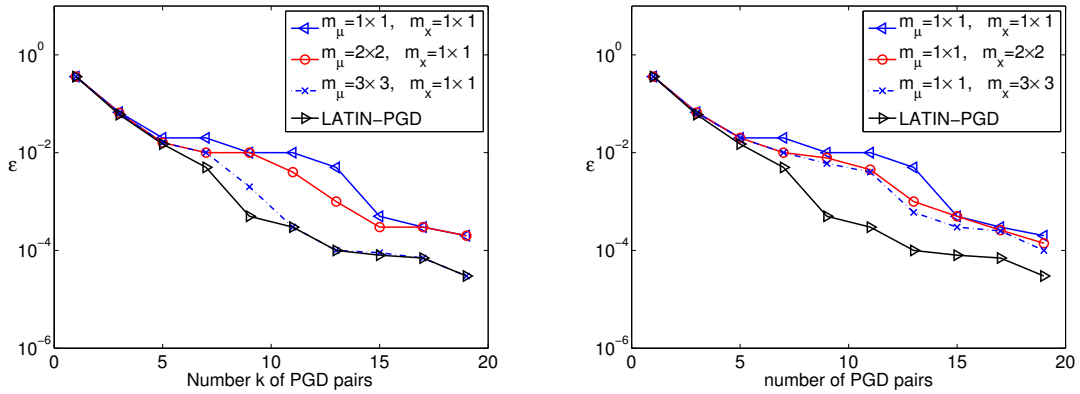
(a) $m_x = 1 \times 1$ reference space points (b) $m_\mu = 2 \times 2$ reference parameter points

Figure 13: Example of selection of reference points on a regular grid for space-parameter domain of **Problem 1**

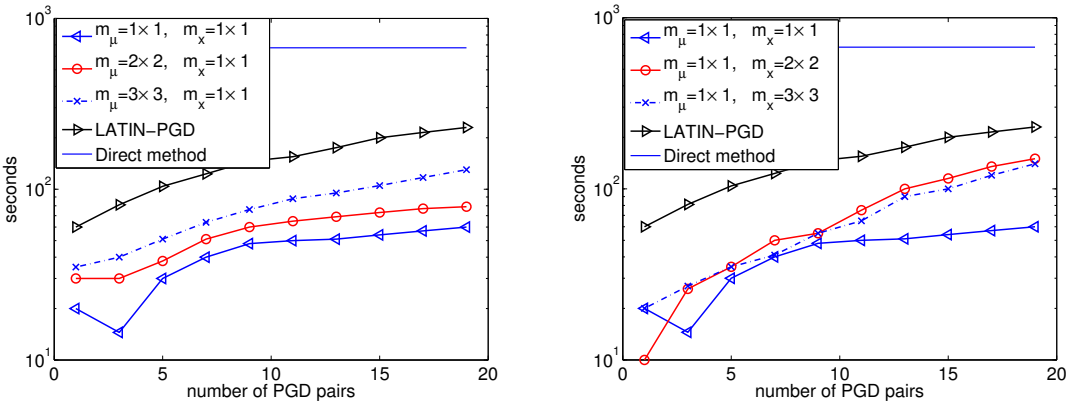
For all the simulations of **Section 4**, the finite element approximation space $\mathcal{V}_h$ is fixed and obtained by a finite element mesh of 50×50 bilinear quadrilateral elements leading to an approximation space of dimension $N = 2601$. The reference solution u_{ref} refers to the solution obtained by solving **Problem 2** with Newton-based nonlinear solution strategy (classical direct method without model reduction) for each parameter value (see **Figure 1**). A number of $p = 225$ values for parameter $\boldsymbol{\mu}$ is considered and chosen from a regular grid as depicted in **Figure 15(a)**. The following relative error ϵ is introduced:

$$\epsilon = 1/p \sum_{j=1}^{p} \frac{\| u_{ref}(\boldsymbol{x}, \boldsymbol{\mu}_j) - u_m(\boldsymbol{x}, \boldsymbol{\mu}_j)\|_{L^2}}{\| u_{ref}(\boldsymbol{x}, \boldsymbol{\mu}_j)\|_{L^2}} \tag{51}$$

where u_m refers to the solution obtained with the chosen model reduction strategy: LATIN-PGD, LATIN-PGD-RPM, RB or RB-EIM. The evolutions of ϵ with respect to the number of PGD pairs for increasing numbers of spatial and parameter reference points are plotted in **Figure 14**. Note that for the LATIN-PGD or the LATIN-PGD-RPM, the number of generated PGD pairs is comparable to the number of LATIN iterations since only one new PGD pair is generated per LATIN iteration at most.



(a) Error ϵ for $m_\mu = \{1\times 1, 2\times 2, 3\times 3\}$ and $m_x = 1\times 1$ (b) Error ϵ for $m_\mu = 1\times 1$ and $m_x = \{1\times 1, 2\times 2, 3\times 3\}$



(c) CPU time for $m_\mu = \{1\times 1, 2\times 2, 3\times 3\}$ and $m_x = 1\times 1$ (d) CPU time for $m_\mu = 1\times 1$ and $m_x = \{1\times 1, 2\times 2, 3\times 3\}$

Figure 14: Evolution of error ϵ (see (51)) and CPU time w.r.t. the number of PGD pairs for increasing numbers of spatial and parameter reference points to solve **Problem 1**

It can be seen on **Figure 14(a)** and **Figure 14(b)** that, by adding more reference points, the error curve of the LATIN-PGD-RPM converges to the one obtained with the LATIN-PGD. The CPU time also increases accordingly and tends to the LATIN-PGD computational cost (see **Figure 14(c)** and **Figure 14(d)**). Let recall that RPM technique is used only for the preliminary step. Thus, if the RPM approximation is not good enough, the generation of new PGD pairs as described in **Algorithm 2** can palliate this and makes the error monotonically decrease and the solution converge anyway. Obviously, this results into the generation of more PGD pairs compared with the LATIN-PGD curve for a given level of error.

It is worth noting at the fact that an error level of 10^{-3} can be reached with only one spatial reference point and a few parameter reference points (see dash-dot line with cross markers in **Figure 14(a)**). For this example, adding more parameter reference points seems to be better than adding more spatial reference points in order to converge to the optimal LATIN-PGD curve (see dash-dot lines with cross markers in **Figure 14(a)** and **Figure 14(c)**). Indeed, as it is often the case in structural mechanics, the spatial gradients of quantities are stronger than their variations in parameter variable for this nonlinear parametrized elliptic problem (see **Figure 1**). Since the parameter reference points $\{\boldsymbol{\mu}_a\}_{1\leqslant a\leqslant m_\mu}$ lead to the computation of m_μ integrals $\overline{\boldsymbol{K}}_b(\boldsymbol{\mu}_a)$ (see (42) and (47)) over the space domain, adding more parameter reference points is preferable to account for spatial gradients.

4.2. Computational gain analysis

As seen in the previous section, the more reference points are added, the more CPU time increases (see **Figure 14(c)** and **Figure 14(d)**). A compromise between the number of reference points – i.e. the CPU time – and the number of generated PGD pairs – i.e. number of LATIN iterations – has to be done in oder to reach a given level of error. For an error level of $\epsilon = 10^{-2}$, the CPU time gain compared with the direct method increases from 6 for the LATIN-PGD to 18 for the LATIN-PGD-RPM (see **Table 2**). For a very small number of reference points ($m_\mu = 1 \times 1$ and $m_x = 1 \times 1$) and a few more PGD pairs, the total CPU time gain is here multiplied by a factor of 3 thanks to the RPM applied to the preliminary step of the LATIN-PGD.

		LATIN-PGD-RPM		
	LATIN-PGD	$m_\mu = 1 \times 1$ $m_x = 1 \times 1$	$m_\mu = 2 \times 2$ $m_x = 1 \times 1$	$m_\mu = 3 \times 3$ $m_x = 1 \times 1$
Number of PGD pairs	7	9	7	7
Gain w.r.t. direct method	6	18	14.2	11.5

Table 2: Total CPU time gain with respect to the direct method for a given level of error ($\epsilon = 10^{-2}$)

Table 3 provides the gain in CPU time on the preliminary step thanks to the RPM. For a significant number of parameter reference points ($m_\mu = 3 \times 3$), the real gain is very close to the theoretical gain in computational complexity (50), that is to say approximately p/m_μ where p is the number of parameter values.

	LATIN-PGD-RPM		
	$m_\mu = 1 \times 1$ $m_x = 1 \times 1$	$m_\mu = 2 \times 2$ $m_x = 1 \times 1$	$m_\mu = 3 \times 3$ $m_x = 1 \times 1$
Real gain with respect to LATIN-PGD	90	39	25
Expected gain ($\approx \frac{p}{m_\mu}$)	225	56	25

Table 3: CPU time gain on the preliminary step with respect to the LATIN-PGD for a given level of error ($\epsilon = 10^{-2}$)

4.3. Comparison of RPM versus EIM: approximation versus interpolation techniques

In this section, a comparison of the performances of the RPM, described in **Section 3**, and the EIM is proposed. It has to be first noticed that, the two methods are here combined with different model reduction techniques and nonlinear strategies (see **Table 4**). As explained in **Section 3.1**, using EIM to solve the preliminary/update step of the PGD could be done but it may be less suitable when the reduced basis evolves. The interested reader could nevertheless refer to [44, 45] for one of the first applications of DEIM to PGD.

In the following, the more classical combination of EIM with reduced-basis techniques and offline-online procedure is considered and called RB-EIM. In this case, combined with Newton-based incremental solution strategy, the reduced-basis approximation associated to the space weak form **Problem 2** for each parameter value $\boldsymbol{\mu} \in \mathcal{D}$ is considered instead of the space-parameter weak form **Problem 3** used in the LATIN non-incremental solution strategy.

Moreover, RPM and EIM are based on different approaches. As seen in **Section 3.1**, RPM is based on an approximation technique of the integrals involved in the Galerkin projection onto an evolving reduced-basis whereas EIM is based on an interpolation procedure of the nonlinear terms (Jacobian and residue).

4.3.1. Summary of the RB-EIM procedure

The main steps of the online-offline procedure of the RB-EIM as described in [15] are here summarized. During the offline stage, the following steps are performed:

	Nonlinear solution strategy	Model order reduction technique	Comments
EIM	Newton method	Reduced-basis (RB)	Offline-online procedure [15]
RPM	LATIN	PGD	Evolving reduced basis

Table 4: Context of application of the RPM and the EIM: LATIN-PGD-RPM and RB-EIM

- Construction of a nested coarse sample set $\mathcal{S}_M^c = \{\boldsymbol{\mu}_m^M \in \mathcal{D}_c\}_{1\leqslant m\leqslant M}$ of magic parameter points thanks to a greedy selection process on coarse grid $\mathcal{D}_c \subset \mathcal{D}$ based on a $L^2(\Omega)$ norm. This requires the solution of M nonlinear problems for each parameter value of $\mathcal{D}_c$ whose computational cost scales with the size of $\mathcal{D}_c$ and N. This cost may be prohibitive in the parabolic case or when the nonlinear function g is time-dependent or implicitly dependent to the field variable u, which is the case here.

- Greedy selection process of a reduced-order basis $\mathcal{W}_k = \{\Phi_i\}_{i=1\leqslant i\leqslant k}$ for the solution u over the coarse sample set $\mathcal{D}_c$. Note that snapshots of the solution, $u(\boldsymbol{\mu}_m^M)$, computed for the elements of nested sample set $\mathcal{S}_M^c \subset \mathcal{D}_c$, are reused in (52).

- Determination of the associated approximation space (collateral reduced-basis):

$$\mathcal{W}_M^c = span\left\{\xi_m = g\left(u(\boldsymbol{\mu}_m^M); \boldsymbol{x}, \boldsymbol{\mu}_m^M\right)\right\}_{1\leqslant m\leqslant M} = span\{q_m(\boldsymbol{x})\}_{1\leqslant m\leqslant M} \tag{52}$$

  where the M functions $\{q_m(\boldsymbol{x})\}_{1\leqslant m\leqslant M}$ and interpolation magic points $\{\boldsymbol{x}_m^M\}_{1\leqslant m\leqslant M}$ are determined progressively by the recursive solution of M small linear systems.

Once order-k reduced basis $\mathcal{W}_k$, collateral reduced-basis $\mathcal{W}_M^c$ and interpolation magic points $\{\boldsymbol{x}_m^M\}_{1\leqslant m\leqslant M}$ have been determined once for all, one can proceed to the online stage on the fine parameter sample set $\mathcal{D}_f \subset \mathcal{D}$. For each parameter value $\boldsymbol{\mu} \in \mathcal{D}_f$ and each iterate $u^{(n)}$, one has the following steps:

- Update of coefficient functions $\{\varphi_m^M(\boldsymbol{\mu})\}_{1\leqslant m\leqslant M}$) for the reduced-order model of $u^{(n)}$ evaluated at interpolation "Magic" points $\{\boldsymbol{x}_m^M\}_{1\leqslant m\leqslant M}$;

- Update of tangent operator at cost $\mathcal{O}(Mk^2)$ in (19) and residue (i.e. Galerkin projection on the order-k reduced basis $\mathcal{W}_k$);

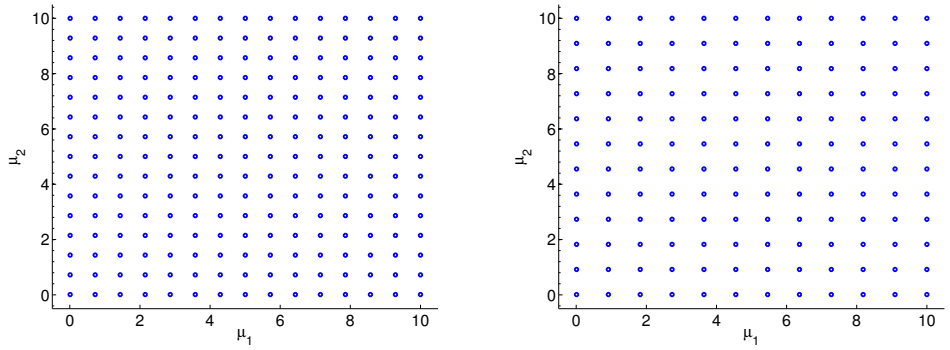
- Solution of the reduced-basis approximation on $\mathcal{W}_k$ associated to the space weak form **Problem 2** for parameter value $\boldsymbol{\mu} \in \mathcal{D}_f$ at cost $\mathcal{O}(k^3)$ (factorization of the full $k \times k$ matrice).

The online computational complexity depends only on k, M and the number of Newton iterations. The N independence of the online stage is recovered. As noticed previously, the costly part of the offline stage is mainly due to the construction of the coarse sample set $\mathcal{S}_M^c$ of magic parameter points $\{\boldsymbol{\mu}_m^M \in \mathcal{D}_c\}_{1\leqslant m\leqslant M}$.

4.3.2. Application of the LATIN-PGD-RPM and the RB-EIM to the reference problem

As in [15], a 15×15 regular grid of parameter values is used for the fine sample set $\mathcal{D}_f \subset \mathcal{D}$ (**Figure 15(a)**). This leads to a number $p = 225$ of parameter values for $\boldsymbol{\mu}$. For the offline stage of the RB-EIM, a 12×12 regular grid of parameter values is chosen for the coarse sample set $\mathcal{D}_c \subset \mathcal{D}$ (**Figure 15(b)**).

As explained at the end of **Section 4.1**, only one spatial reference point is used in the following for the LATIN-PGD-RPM. Thus, one sets $m_x = 1 \times 1$. Only the number m_μ of parameter reference points varies. For the RB-EIM, the influence on the convergence of the number M of magic points for a given size k of the reduced basis $\mathcal{W}_k$ generated offline is investigated. For the LATIN-PGD-RPM, the reduced basis $\mathcal{W}_k$ is progressively built and its size k increases throughout the LATIN iterations. As previously said, the number k of generated PGD pairs is comparable to the number of LATIN iterations since only one new PGD pair

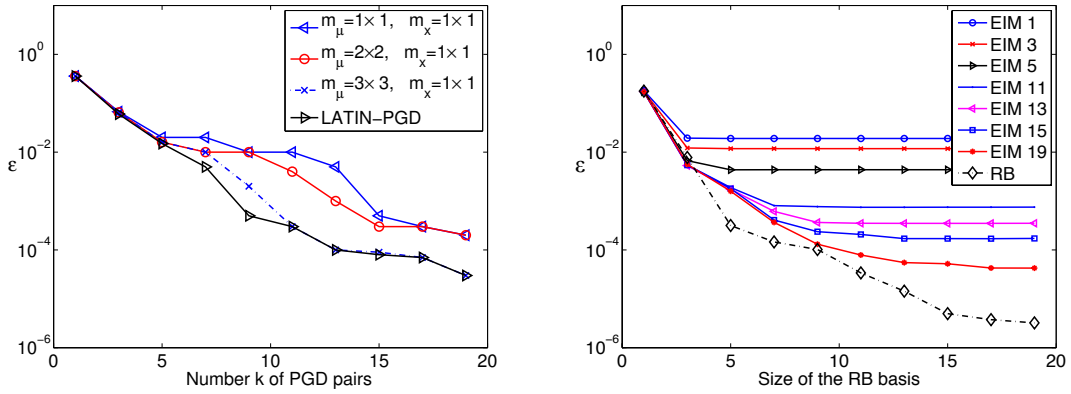


(a) Fine regular grid $\mathcal{D}_f$ of 15×15 points (b) Coarse regular grid $\mathcal{D}_c$ of 12×12 points

Figure 15: Fine and coarse grids, $\mathcal{D}_f$ and $\mathcal{D}_c$, for the parameter domain discretization considered for the RB-EIM [15]

is generated per LATIN iteration at most.

The error ϵ given by (51) for various numbers of reference and magic points and **Problem 1** is given in **Figure 16**. The error of the classical reduced basis approach without EIM (dash-dot line with diamond

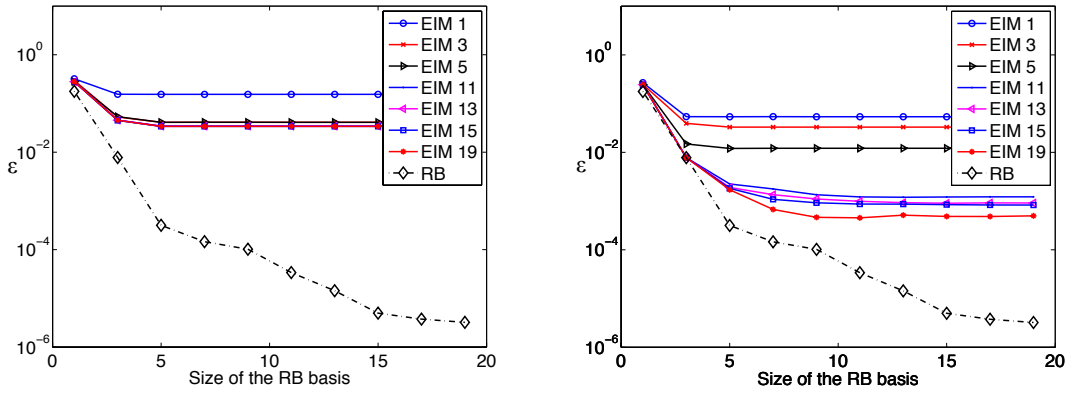


(a) Error ϵ for the LATIN-PGD-RPM with $m_x = 1 \times 1$ and $m_\mu = \{1 \times 1, 2 \times 2, 3 \times 3\}$ (b) Error ϵ for the RB-EIM and various numbers M of magic points from 1 to 19

Figure 16: Error ϵ given by (51) with respect to the size k of the reduced basis $\mathcal{W}_k$ for the LATIN-PGD-RPM and the RB-EIM and various numbers of reference/magic points for **Problem 1**

markers denoted by RB) is also given for comparison. Both the LATIN-PGD-RPM and the RB-EIM lead to the same level of error. However, some differences can be pointed out. For a given number M of magic points, the RB-EIM convergence curve first decreases until a plateau is reached once k slightly exceeds the value of M (see **Figure 16(b)**). Increasing M is necessary to make the error decrease. Indeed, for a fixed number M of magic interpolation points the error in the coefficient functions $\{\varphi_m^M(\boldsymbol{\mu})\}_{1 \leqslant m \leqslant M}$ used to interpolate the nonlinear terms (see (17)) ultimately dominates for increasing dimension k of the reduced basis $\mathcal{W}_k$. Increasing both M and k is necessary to lower the error. In the case of the LATIN-PGD-RPM (see **Figure 16(a)**), the reduced basis $\mathcal{W}_k$ being enriched throughout the iterations, the error decreases monotonically as expected by generating new PGD pairs. Increasing the number m_μ of parameter reference points improves the overall convergence rate. In any case, the LATIN-PGD-RPM solution converges.

As pointed out in **Section 4.3.1**, despite the fact that the computational cost of the online stage depends only on k, M and the number of Newton iterations and no more on N, the cost of the offline/learning stage can be very high for nonlinear problems in order to generate a pertinent reduced-basis $\mathcal{W}_k$ as well as a suitable coarse sample set $\mathcal{S}_M^c$ of magic parameter points. A *sufficiently fine* coarse grid $\mathcal{D}_c$ may be necessary. More precisely, for the 12×12 coarse grid $\mathcal{D}_c$ of **Figure 15(b)**, the offline stage of the RB-EIM accounts for 64% of the computational cost of the classic direct simulation performed on the 15×15 fine grid $\mathcal{D}_f$ of **Figure 15(a)**. Error ϵ given by (51) for the RB-EIM with various regular coarse grids $\mathcal{D}_c$ used during the offline stage is plotted in **Figure 17**. It can be seen on **Figure 17(a)** that a too coarse grid leads to a high level of error whatever the number M of magic points is. For a 5×5 grid, it can be seen on **Figure 17(b)** that error level is hardly improved for M higher than about 11.



(a) Coarse grid $\mathcal{D}_c$ of 2×2 points

(b) Coarse grid $\mathcal{D}_c$ of 5×5 points

(c) Coarse grid $\mathcal{D}_c$ of 10×10 points

(d) Coarse grid $\mathcal{D}_c$ of 12×12 points

Figure 17: Error ϵ given by (51) for the RB-EIM with various regular coarse grids $\mathcal{D}_c$ used during the offline stage

Finally, even though comparing a reduced basis approach with learning stage and a PGD-based model reduction may be discussable, one nevertheless provides, in **Table 5**, CPU time gains with respect to the direct simulation to reach an error level of $\epsilon = 10^{-2}$ for the LATIN-PGD-RPM and the RB-EIM with a comparable number of reference/magic points: $m_x = 1 \times 1$, $m_\mu = 2 \times 2$ and $M = 5$. As discussed previously, note that CPU time for the RB-EIM includes the time spent during the offline/learning stage, the online stage cost being negligible. It can be seen that the CPU time gains for the two approaches are quite similar

in the case of **Problem 1**. The size k of the reduced basis $\mathcal{W}_k$ generated by the LATIN-PGD-RPM is also comparable to the size of the reduced basis built by the RB-EIM during the offline stage.

	LATIN-PGD-RPM	RB-EIM
	$m_x = 1 \times 1$	$M = 5$
	$m_\mu = 4 \times 4$	
Size k of reduced basis $\mathcal{W}_k$	7	5
Gain w.r.t. direct method	14.2	18

Table 5: CPU time gain with respect to the direct simulation to reach an error level of $\epsilon = 10^{-2}$ for the LATIN-PGD-RPM and the RB-EIM with a comparable number of reference/magic points

5. Conclusion

In order to tackle the bottleneck of nonlinear model reduction techniques, a new approximation technique, called Reference Point Method, has been proposed to reduce the computational complexity of algebraic operations for constructing reduced-order models in the case of time dependent and/or parametrized nonlinear partial differential equations. It has been shown that this bottleneck is not specific to PGD-based model reduction but more generally to reduced-basis approximation in nonlinear model reduction. This problem arises due to the fact that at each iteration of the solution strategy: (i) Galerkin projection onto the ROB has to be performed, especially each time the reduced-order basis evolves which is the case for PGD, (ii) non-linear terms (Jacobian and residue) have to be evaluated for each new iterate which prevents pre-computations of operators. As a result, the complexity of these operations scales with the size of the original high-dimensional model.

Contrary to EIM which is based on an interpolation procedure of the nonlinear terms (Jacobian and residue), RPM is based on an approximation technique of the integrals involved in the Galerkin projection onto an evolving reduced-basis. For that two main features are introduced: (i) an approximation framework with the introduction of reference points to simplify the evaluation of integrands, (ii) a low-cost technique to reconstruct by explicit formulas an approximate low-rank separated representation by patch of operators without resorting to SVD-based techniques, which is a great advantage regarding integral computation by separation of variables.

An application of the RPM with the LATIN-PGD nonlinear solution strategy to a nonlinear parametrized elliptic PDE previously studied by other authors with reduced-basis method and EIM has been proposed. More precisely, the RPM has been used to approximate, at each iteration of the nonlinear solution strategy, the so-called *Preliminary step* which is nothing more than an update of the reduced model from the reduced basis obtained from the previous iterations.

It has been shown that the convergence of the whole strategy is ensured, whatever the number of reference points is (even if a very moderate increase of the number of the generated PGD modes is required to compensate a possible lack of accuracy of the RPM approximation). Increasing the number of reference points makes the convergence curve tend to the one obtained with the classical LATIN-PGD approach.

Moreover, computational complexity as well as CPU time to construct the reduced-order model of the preliminary step can be divided in practice by one order of magnitude. The implementation of RPM for more complex problems is currently in progress and one can expect higher gains.

Acknowledgements

This work was supported by the French *Agence Nationale pour la Recherche* through the SIM-DREAM project ANR-10-COSI-0006.

Appendix A. Nonlinear solution strategy by an alternating-direction scheme: the LATIN method

In order to solve **Problem 3**, the LATIN method is used as a nonlinear solution strategy. By a two-stage iterative scheme and two search directions, the LATIN method generates approximations of the solution that belong alternatively to two manifolds, $\mathbf{A}_d$ and $\mathbf{\Gamma}$ as represented in **Figure A.18**. Manifold $\mathbf{A}_d$ is defined by the linear, possibly global, equations of the problem whereas manifold $\mathbf{\Gamma}$ contains the local, possibly nonlinear, equations. Similarly to augmented Lagrangian techniques, the key point of the strategy consists, for **Problem 3**, in introducing manifold $\mathbf{\Gamma} = \{(u, w) \in \mathcal{V} \times \mathcal{W};\ w = g(u; \boldsymbol{\mu}), \forall \mu \in \mathcal{D}\}$ with $\mathcal{W} = L^2(\Omega)$. The problem solution is found by solving alternatively the *local stage* and the *global stage* as described hereafter.

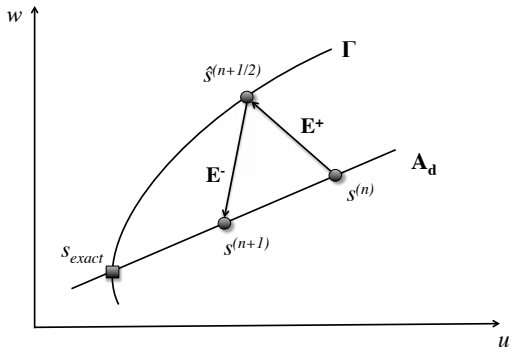


Figure A.18: The LATIN alternating-direction scheme: manifolds $\mathbf{A}_d$ and $\mathbf{\Gamma}$, search directions $\mathbf{E}^+$ and $\mathbf{E}^-$

.

Appendix A.1. Local stage at iteration $n+1$

Local stage consists in building solution $\hat{s}^{(n+1/2)} = (\hat{u}^{(n+1/2)}, \hat{w}^{(n+1/2)}) \in \mathbf{\Gamma}$, i.e. that verifies nonlinear equation $\hat{w}^{(n+1/2)} = g(\hat{u}^{(n+1/2)}; \boldsymbol{\mu})$ for all $\boldsymbol{\mu}$, knowing solution $s^{(n)} = (u^{(n)}, w^{(n)}) \in \mathbf{A}_d$ coming from the previous global stage thanks to search direction $\mathbf{E}^+$, verified by $\hat{s}^{(n+1/2)} - s^{(n)}$:

$$\mathbf{E}^+ : \forall\, \boldsymbol{\mu} \in \mathcal{D},\ \forall\, \boldsymbol{x} \in \Omega, \quad (\hat{w}^{(n+1/2)} - w^{(n)}) + \mathbf{H}^+(\hat{u}^{(n+1/2)} - u^{(n)}) \ = \ 0 \tag{A.1}$$

where $\mathbf{H}^+$ is a symmetric definite operator which is a parameter of the method. At this stage, the problem is nonlinear but local in space variable. A simple choice is to take a *stiff* ascending search direction ($\mathbf{H}^+ \to \infty$), which is equivalent to set $\hat{u}^{(n+1/2)} = u^{(n)}$. In this case, solution is found explicitly without resorting to a local nonlinear solver and solution $\hat{s}^{(n+1/2)} = (\hat{u}^{(n+1/2)}, \hat{w}^{(n+1/2)})$ is simply defined by:

$$\forall\, \boldsymbol{\mu} \in \mathcal{D},\ \forall\, \boldsymbol{x} \in \Omega, \quad \left\{ \begin{array}{rcl} \hat{u}^{(n+1/2)} & = & u^{(n)} \\ \hat{w}^{(n+1/2)} & = & g(\hat{u}^{(n+1/2)}; \boldsymbol{\mu}) = g(u^{(n)}; \boldsymbol{\mu}) \end{array} \right. \tag{A.2}$$

Appendix A.2. Global stage at iteration $n+1$

Global stage consists in building solution $s^{(n+1)} = (u^{(n+1)}, w^{(n+1)}) \in \mathbf{A}_d$, knowing solution $\hat{s}^{(n+1/2)} = (\hat{u}^{(n+1/2)}, \hat{w}^{(n+1/2)}) \in \mathbf{\Gamma}$ coming from the previous local stage thanks to search direction $\mathbf{E}^-$ satisfied by $s^{(n+1)} - \hat{s}^{(n+1/2)}$:

$$\mathbf{E}^- : \forall\, \boldsymbol{\mu} \in \mathcal{D},\ \forall\, \boldsymbol{x} \in \Omega, \quad (w^{(n+1)} - \hat{w}^{(n+1/2)}) - \mathbf{H}^-(u^{(n+1)} - \hat{u}^{(n+1/2)}) \ = \ 0 \tag{A.3}$$

Similarly to ascending search direction $\mathbf{H}^+$ of the local stage, the descending search direction $\mathbf{H}^-$ is a parameter of the method. It is shown in [22] that a well-suited choice is to choose the tangent operator for $\mathbf{H}^-$:

$$\mathbf{H}^- = \left.\frac{\partial g(u; \boldsymbol{\mu})}{\partial u}\right|_{u=\hat{u}^{(n+1/2)}} = \left.\frac{\partial g(u; \boldsymbol{\mu})}{\partial u}\right|_{u=\hat{u}^{(n+1/2)}=u^{(n)}} \tag{A.4}$$

Second equality is due to the choice made for search direction $\mathbf{E}^-$ (see (A.2)). A constant operator could be also chosen:

$$\mathbf{H}^- = \left.\frac{\partial g(u;\boldsymbol{\mu})}{\partial u}\right|_{u=\hat{u}^{(0)}} = \mathbf{H}_\mathbf{0}^- \tag{A.5}$$

In this case, the algorithm bears similarities to a quasi-Newton scheme. As shown in [22], the choice of search directions $\mathbf{H}^+$ and $\mathbf{H}^-$ does not affect the solution at convergence but it can change the convergence rate. **Figure A.19** depicts the evolution of LATIN convergence indicator δ_L (see (A.9) in **Appendix A.3**) as a function of the number of iterations for **Problem 1**. It can be seen that using a tangent operator (A.4) instead of a constant one (A.5) drastically improves the convergence rate. Using tangent operator (A.4) or, at least, updating search direction $\mathbf{H}^-$ is consequently recommended.

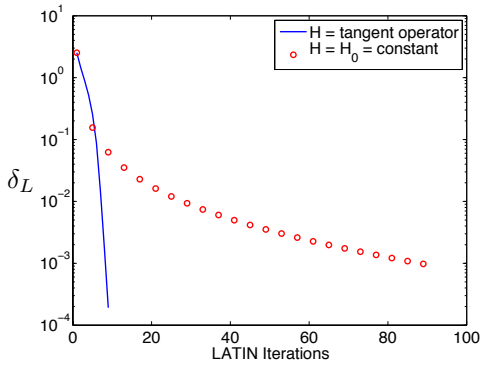


Figure A.19: Convergence of LATIN error indicator δ_L (A.9) for **Problem 1** and two choices of search direction $\mathbf{H}^-$: tangent operator (A.4) and constant operator (A.5)

The global stage problem reads as follows:

Problem 5 (Global stage). *Given* $\hat{s}^{(n+1/2)} = (\hat{u}^{(n+1/2)}, \hat{w}^{(n+1/2)})$, *find* $s^{(n+1)} = (u^{(n+1)}, w^{(n+1)}) \in \mathbf{A}_d$ *that satisfies search direction* (A.3) *and such that:*

$$\forall v \in \mathcal{S}, \quad \int_\mathcal{D} a(u^{(n+1)}, v)\,\mathrm{d}\mu + \int_{\mathcal{D}\times\Omega} w^{(n+1)}\ v\ \mathrm{d}x\,\mathrm{d}\mu = \int_{\mathcal{D}\times\Omega} f(\boldsymbol{x})\ v\ \mathrm{d}x\,\mathrm{d}\mu \tag{A.6}$$

By introducing search direction (A.3) in (A.6), one obtains:

$$\begin{aligned}\forall v \in \mathcal{S}, \quad \int_\mathcal{D} a(u^{(n+1)}, v)\,\mathrm{d}\mu + \int_{\mathcal{D}\times\Omega} \mathbf{H}^-\ u^{(n+1)}\ v\ \mathrm{d}x\,\mathrm{d}\mu = \ldots \\ \ldots \int_{\mathcal{D}\times\Omega} f(\boldsymbol{x})\ v\ \mathrm{d}x\,\mathrm{d}\mu - \int_{\mathcal{D}\times\Omega} \left(\hat{w}^{(n+1/2)} - \mathbf{H}^-\hat{u}^{(n+1/2)}\right)\ v\ \mathrm{d}x\,\mathrm{d}\mu\end{aligned} \tag{A.7}$$

The only unknown in this latter equation is $u^{(n+1)}$. This equation is global over the space-parameter domain but linear.

Due to the specific choices done for search directions $\mathbf{E}^-$ (see (A.2)) and $\mathbf{E}^+$ (see (A.4)) and by introducing the correction $\delta u^{(n+1)} = u^{(n+1)} - u^{(n)}$ between two consecutive global stages, **Problem 5** also reads:

Problem 6 (Global stage on correction). *Find* $\delta u^{(n+1)} = u^{(n+1)} - u^{(n)} \in \mathcal{S}$ *and* $w^{(n+1)} \in \mathcal{S}$ *such that:*

$$\forall v \in \mathcal{S}, \qquad \int_{\mathcal{D}} a(\delta u^{(n+1)}, v)\,\mathrm{d}\mu + \int_{\mathcal{D}\times\Omega} \mathbf{H}^{-}(u^{(n)};\boldsymbol{\mu})\ \delta u^{(n+1)}\ v\ \mathrm{d}x\,\mathrm{d}\mu = -\int_{\mathcal{D}} \mathcal{R}(u^{(n)}, v;\boldsymbol{\mu})\,\mathrm{d}\mu$$

$$\forall\, \boldsymbol{\mu} \in \mathcal{D},\ \forall\, \boldsymbol{x} \in \Omega, \qquad w^{(n+1)} = \hat{w}^{(n+1/2)} + \mathbf{H}^{-}\,\delta u^{(n+1)} \tag{A.8}$$

where :

$$\begin{cases} \mathcal{R}(u^{(n)}, v;\boldsymbol{\mu}) &= a(u^{(n)}, v) \ - \ \displaystyle\int_{\Omega} f(\boldsymbol{x})\ v\ \mathrm{d}x \ + \ \int_{\Omega} g(u^{(n)};\boldsymbol{\mu})\ v\ \mathrm{d}x \\ \mathbf{H}^{-}(u^{(n)};\boldsymbol{\mu}) &= \left.\dfrac{\partial g(u;\boldsymbol{\mu})}{\partial u}\right|_{u=u^{(n)}} \end{cases}$$

At each iteration, a local stage and a global stage have to be solved until convergence. It is clear that contrary to classical incremental solution strategies, one has to store the iterate on the whole space-parameter domain. Indeed, at each stage, one seeks a correction $\delta u^{(n+1)} \in \mathcal{S} = L^2(\mathcal{D}, \mathcal{V})$. In order to overcome this data storage issue, a dedicated representation of unknowns has to be proposed. In the LATIN method [22], Proper generalized decomposition of the iterates is classically used to the purpose as described in **Section 2.2**.

Appendix A.3. LATIN convergence indicator

In order to check the convergence of the nonlinear iterative scheme, the following error indicator is defined:

$$\delta_L^{(n+1)} = \frac{\|u^{(n+1)} - \hat{u}^{(n+1/2)}\|}{\frac{1}{2}\,\|u^{(n+1)} + \hat{u}^{(n+1/2)}\|} \quad \text{with} \quad \|\cdot\| = \int_{\mathcal{D}\times\Omega} \cdot^{\,2}\ \mathrm{d}x\,\mathrm{d}\mu$$

This error indicator measures, at iteration $n+1$, the distance between the iterate of $\boldsymbol{\Gamma}$ generated at the local stage and the subsequent iterate of $\mathbf{A}_d$ obtained at the global stage (**Figure A.18**). Due to the particular choice made for search direction $\mathbf{H}^{+}$ (see (A.2)), it also simply reads:

$$\delta_L^{(n+1)} = \frac{\|u^{(n+1)} - u^{(n)}\|}{\frac{1}{2}\|u^{(n+1)} + u^{(n)}\|} = \frac{\|\delta u^{(n+1)}\|}{\frac{1}{2}\|u^{(n+1)} + u^{(n)}\|} \tag{A.9}$$

Under the hypothesis of a monotonous operator $g(u;\boldsymbol{\mu})$, the iterative scheme converges to the reference solution $\mathbf{s}_{ref}$ for any choice of symmetric positive operator $\mathbf{H}^{-}$ [22].

Appendix B. LATIN-PGD final algorithm

The final algorithm of the LATIN-PGD nonlinear solution strategy is given in **Algorithm 2**.

Remark 4. *The computation of an admissible initial solution $s^{(0)} = (u^{(0)}, w^{(0)}) \in \mathbf{A}_d$ depends on the problem to solve. It can be obtained by solving the problem for a given value of parameter or a simplified problem by neglecting nonlinear terms. One can also solve **Problem 6** for $n = -1$ by a progressive Galerkin PGD with **Algorithm 1**. In this case, the first generated spatial function Φ_1 can be chosen as the first element of the reduced-order basis $\mathcal{W}_1$. For more details on the initialization of the algortihm, the interested reader can refer for example to [31].*

Algorithm 2 LATIN-PGD nonlinear solution strategy

$s^{(-1)} = (u^{(-1)}, w^{(-1)}) \leftarrow (0,0)$ ▷ Initialization
Compute $\hat{s}^{(-1/2)} = (\hat{u}^{(-1/2)}, \hat{w}^{(-1/2)})$ ▷ Solve local stage (A.2) for $n = -1$
Compute $s^{(0)} = (u^{(0)}, w^{(0)}) \in \mathbf{A}_d$ ▷ Solve **Problem 6** for $n = -1$ (see **Remark 4**)
Create reduced-order basis $\mathcal{W}_1 \leftarrow \{\Phi_1\}$ ▷ See **Remark 4**
for $n = 0$ to n_{max} **do** ▷ LATIN iterations
  procedure LOCAL STAGE
    Compute $\hat{s}^{(n+1/2)} = (\hat{u}^{(n+1/2)}, \hat{w}^{(n+1/2)}) \in \mathbf{\Gamma}$ ▷ Solve local stage (A.2)
  end procedure
  procedure GLOBAL STAGE ▷ Compute $s^{(n+1)} = (u^{(n+1)}, w^{(n+1)}) \in \mathbf{A}_d$
    Update parameter functions $\{\lambda_i \leftarrow \lambda_i + \delta\lambda_i\}_{1 \leqslant i \leqslant k}$ ▷ Solve preliminary step **Problem 4**
    $\breve{u}_k^{(n+1)} = u^{(0)} + \sum_{i=1}^{k} \Phi_i \, \lambda_i$
    Compute performance indicator η_0 (5)
    if $\eta_0 \leqslant tol$ **then**
      Generate a new PGD pair $(\lambda_{k+1}, \Phi_{k+1})$ with **Algorithm 1**
      procedure ROB UPDATE ▷ Use Gram-Schmidt process with inner product $(\cdot,\cdot)$
        Update parameter functions $\{\lambda_i \leftarrow \lambda_i + \lambda_{k+1}(\Phi_i, \Phi_{k+1})\}_{1 \leqslant i \leqslant k}$
        Orthogonalize space function $\Phi_{k+1} \leftarrow \overline{\Phi}_{k+1}$ w.r.t. current ROB $\mathcal{W}_k$
        if Φ_{k+1} is a "new" function **then**
          Update ROB $\mathcal{W}_k \leftarrow \mathcal{W}_{k+1}$ by adding new orthogonalized function
          $u^{(n+1)} = u^{(0)} + \sum_{i=1}^{k+1} \Phi_i \, \lambda_i$
        else
          $u^{(n+1)} = u^{(0)} + \sum_{i=1}^{k} \Phi_i \, \lambda_i$
        end if
      end procedure
    else
      $u^{(n+1)} = \breve{u}_k^{(n+1)}$
    end if
    $w^{(n+1)} = \hat{w}^{(n+1/2)} + \mathbf{H}^{-} \left(u^{(n+1)} - \hat{u}^{(n+1/2)}\right)$
  end procedure
  Check LATIN convergence indicator δ_L (A.9)
end for

Appendix C. Minimization of the RPM functional

Minimization of functional $J(\{(a_{ij}, b_{ij})\}_{1 \leqslant j \leqslant m_x})$ with $1 \leqslant i \leqslant m_\mu$ (see (27)) leads to:

$$\forall (\delta a_{ik}, \delta b_{ik}),$$
$$\sum_{k=1}^{m_x} \Bigg[\Delta\omega_k \; \lambda_{ik}^2 \int_{\mathcal{D}_i} \left(\delta a_{ik}(\mu) \; b_{ik}(x_k) + a_{ik}(\mu) \; \delta b_{ik}(x_k)\right) \left(\bar{A}_{ik}(\mu) - a_{ik}(\mu) b_{ik}(x_k)\right) d\mu \; + \ldots$$
$$\ldots \; \Delta\mu_i \int_{\Omega_j} \left(\delta a_{ik}(\mu_i) \; b_{ik}(x) + a_{ik}(\mu_i) \; \delta b_{ik}(x)\right) \left(\bar{B}_{ik}(x) - a_{ik}(\mu_i) b_{ik}(x)\right) dx \Bigg] = 0 \quad \text{(C.1)}$$

By considering variations $(\delta a_{ik}, \delta b_{ik})$ such that $\forall \; (i,k)$, $\delta b_{ik}(x) = 0$ and $\delta a_{ik}(\mu_i) = 0$, one gets from (C.1): for $1 \leqslant i \leqslant m_\mu$ and $1 \leqslant k \leqslant m_x$:

$$a_{ik}(\mu) = \frac{\sum_{k=1}^{m_x} \Delta\omega_k \; b_{ik}(x_k) \; \bar{A}_{ik}(\mu) \; \lambda_{ik}^2}{\sum_{k=1}^{m_x} \; \Delta\omega_k \; b_{ik}^2(x_k) \; \lambda_{ik}^2}, \quad \forall \; \mu \in \mathcal{D}_i \quad \text{(C.2)}$$

Similarly, by considering variations $(\delta a_{ik}, \delta b_{ik})$ such that $\forall\ (i,k)$, $\delta b_{ik}(x_j) = 0$ and $\delta a_{ik}(\mu) = 0$, (C.1) leads to: for $1 \leqslant i \leqslant m_\mu$ and $1 \leqslant k \leqslant m_x$:

$$b_{ij}(x) = \frac{\bar{B}_{ij}(x)}{a_{ij}(\mu_i)}, \quad \forall\ x\ \in \Omega_j \tag{C.3}$$

Equation (C.3) and (C.2) gives: for $1 \leqslant i \leqslant m_\mu$ and $1 \leqslant k \leqslant m_x$:

$$a_{ij}(\mu) = \left(\frac{\sum_{k=1}^{m_x} \Delta\omega_k\ \bar{A}_{ik}(\mu)\ \bar{B}_{ik}(\mu_i)\ \lambda_{ik}^2}{\sum_{k=1}^{m_x}\ \Delta\omega_k\ \bar{B}_{ik}(x_k)^2\ \lambda_{ik}^2} \right) a_{ik}(\mu_i), \quad \forall\ \mu \in \mathcal{D}_i \tag{C.4}$$

Since $\bar{B}_{ik}(x_k)\ =\ \bar{A}_{ik}(\mu_i)$ by definition of the generalized components (see (24)), this also reads:

$$a_{ij}(\mu) = \left(\frac{\sum_{k=1}^{m_x} \Delta\omega_k\ \bar{A}_{ik}(\mu)\ \bar{A}_{ik}(\mu_i)\ \lambda_{ik}^2}{\sum_{k=1}^{m_x}\ \Delta\omega_k\ \bar{A}_{ik}(\mu_i)^2\ \lambda_{ik}^2} \right) a_{ij}(\mu_i), \quad \forall\ \mu \in \mathcal{D}_i \tag{C.5}$$

Reconstruction $\bar{\bar{F}}$ of F (see (26)) is given by: for $1 \leqslant i \leqslant m_\mu$ and $1 \leqslant j \leqslant m_x$:

$$F(\mu, x) \approx \bar{\bar{F}}(\mu, x) = \left(\frac{\sum_{k=1}^{m_x} \Delta\omega_k\ \bar{A}_{ik}(\mu)\ \bar{A}_{ik}(\mu_i)\ \lambda_{ik}^2}{\sum_{k=1}^{m_x}\ \Delta\omega_k\ \bar{A}_{ik}(\mu_i)^2\ \lambda_{ik}^2} \right) \bar{B}_{ij}(x), \quad \forall (\mu, x) \in \mathcal{D}_i \times \Omega_j \tag{C.6}$$

Without changing the reconstruction, one can slightly modify the expressions (C.5) and (C.3) for functions a_{ij} and b_{ij} by scaling a_{ij} and multiplying b_{ij} by $a_{ik}(\mu_i)$, their product remaining unchanged. This leads to the following explicit formulas: for $1 \leqslant i \leqslant m_\mu$ and $1 \leqslant j \leqslant m_x$:

$$a_{ij}(\mu) = \frac{\sum_{k=1}^{m_x} \Delta\omega_k\ \bar{A}_{ik}(\mu)\ \bar{A}_{ik}(\mu_i)\ \lambda_{ik}^2}{\sum_{k=1}^{m_x}\ \Delta\omega_k\ \bar{A}_{ik}(\mu_i)^2\ \lambda_{ik}^2} \qquad \text{and} \qquad b_{ij}(x) = \bar{B}_{ij}(x) \tag{C.7}$$

References

[1] A. Chatterjee, An introduction to the proper orthogonal decomposition, Current Science 78 (7) (2000) 808–817.

[2] J. A. Atwell, B. B. Kings, Proper orthogonal decomposition for reduced basis feedback controllers for parabolic equations, Mathematical and computer modelling 33 (2001) 1–19.

[3] K. Kunish, L. Xie, Pod-based feedback control of the burgers equation by solving the evolutionary hjb equation., Computers and Mathematics With Applications 49(7-8) (2005) 5730 – 5742.

[4] K. Carlberg, C. Bou-Mosleh, C. Farhat, Efficient nonlinear model reduction via a least-squares petrov-galerkin projection and compressive tensor approximations., International Journal for Numerical Methods in Engineering 86 (1) (2010) 155–181.

[5] S. Zhu, L. Dedè, A. Quarteroni, Isogeometric analysis and proper orthogonal decomposition for parabolic problems, Numerische Mathematik 135 (2) (2017) 333–370.

[6] G. Kerschen, J. Golinval, A. Vakakis, L. Bergman, The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: An overview, Nonlinear Dynamics 1 (41) (2005) 147–169.

[7] P. Glüsmann, E. Kreuzer, On the application of karhunen–loève transform to transient dynamic systems, Journal of Sound and Vibration 328 (4-5) (2009) 507–519.

[8] L. Boucinha, A. Ammar, A. Gravouil, A. Nouy, Ideal minimal residual-based proper generalized decomposition for non-symmetric multi-field models – application to transient elastodynamics in space-time domain, Computer Methods in Applied Mechanics and Engineering 273 (1) (2014) 56–76.

[9] C. Prud'homme, D. Rovas, K. Veroyand, L. Machiels, Y. Maday, A. Patera, G. Turinici, Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods, Journal of Fluids Engineering 124 (1) (2002) 70–80.

[10] Y. Maday, E. M. Ronquist., The reduced-basis element method: application to a thermal fin problem, SIAM Journal on Scientific Computing 26(1) (2004) 240–258.

[11] K. Veroy, A. T. Patera, Certified real-time solution of the parametrized steady incompressible navier-stokes equations: rigorous reduced-basis a posteriori error bounds, International Journal for Numerical Methods in Fluids 47 (1) (2005) 773–788.

[12] N. Nguyen, K. Veroy, A. Patera, Certified real-time solution of parametrized partial differential equations, Vol. Handbook of materials modeling, Springer, Berlin, 2005, pp. 1523 – 1558.

[13] G. Rozza, Reduced-basis methods for elliptic equations in sub-domains with a posteriori error bounds and adaptivity, Applied Numerical Mathematics (55) (2004) 403–424.

[14] A. Quarteroni, G. Rozza, A. Manzoni, Certified reduced basis approximation for parametrized partial differential equations and applications, Journal of Mathematics in Industry 1 (1) (2011) 3.

[15] M. Grepl, Y. Maday, N. Nguyen, A. Patera, Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations, Modélisation mathématique et analyse numérique 41 (3) (2007) 575–605.

[16] G. Rozza, A. T. Patera, Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations, Archives of Computational Methods In Engineering (15) (2008) 229–275.

[17] J. Galvis, S. K. Kang, Spectral multiscale finite element for nonlinear flows in highly heterogeneous media: A reduced basis approach, Journal of Computational and Applied Mathematics 260 (1) (2014) 494–508.

[18] F. Chinesta, P. Ladevèze, E. Cueto, A short review on model order reduction based on proper generalized decomposition, Arch Comput Methods Eng (18) (2011) 395–404.

[19] F. Chinesta, P. Ladevèze, Separated Representations and PGD-Based Model Reduction: Fundamentals and Applications, CISM International Centre for Mechanical Sciences, Springer Vienna, 2014.

[20] F. Chinesta, R. Keunings, A. Leygue, The Proper Generalized Decomposition for Advanced Numerical Simulations: A Primer, SpringerBriefs in Applied Sciences and Technology, Springer International Publishing, 2013.

[21] P. Ladevèze, Sur une famille d'algorithmes en mécanique des structures, Comptes Rendus Académie des Sciences. Paris. Ser. II (300) (1985) 41–44.

[22] P. Ladevèze, Nonlinear Computational Structural Mechanics - new approaches and non-incremental methods of calculation, Mechanical Engineering Series, Springer New York, 1999.

[23] P. Boisse, P. Bussy, P. Ladevèze, A new approach in non-linear mechanics: the large time increment method, International Journal for Numerical Methods in Engineering 29 (1990) 647–663.

[24] D. Ryckelynck, Réduction a priori de modèles thermomécaniques, Compte Rendu Mecanique 330 (2002) 499–505.

[25] P. Ladevèze, A. Nouy, On a multiscale computational strategy with time and space homogenization for structural mechanics, Computational Methods Applied Mechanical Engineering (192) (2003) 3061–3087.

[26] P. Ladevèze, J.-C. Passieux, D. Néron, On multiscale computational mechanics with time-space homogenization, Oxford University Press, 2009, pp. 247–282.

[27] P. Ladevèze, J.-C. Passieux, D. Néron, The latin multiscale computational method and the proper generalized decomposition, Computational Methods Applied Mechanical Engineering (199) (2010) 1287–1296.

[28] P. Ladevèze, On reduced models in nonlinear solid mechanics, European Journal of Mechanics - A/Solids 60 (2016) 227–237.

[29] A. Nouy, A priori model reduction through proper generalized decomposition for solving time-dependent partial differential equations, Computational Methods Applied Mechanical Engineering 199 (23-24) (2010) 1603–1626.

[30] G. Bonithon, A. Nouy, A priori tensor approximations for the numerical solution of high dimensional problems: alternative definitions, in: 28th GAMM-Seminar Leipzig on Analysis and Numerical Methods in Higher Dimensions, Leipzig, Allemagne, 2012.

[31] D. Néron, P.-A. Boucard, N. Relun, Time-space pgd for the rapid solution of 3d nonlinear parametrized problems in the many-query context, International Journal for Numerical Methods in Engineering 103 (4) (2015) 275–292.

[32] D. Amsallem, J. Cortial, K. Carlberg, C. Farhat, A method for interpolating on manifolds structural dynamics reduced-order models, International Journal for Numerical Methods in Engineering 80 (1) (2009) 1241–1258.

[33] M. Barrault, Y. Maday, N. Nguyen, A. Patera, An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations, Comptes Rendus Académie des Sciences. Paris. Ser. I (339) (2004) 667–672.

[34] N. C. Nguyen, J. Peraire, An efficient reduced-order modeling approach for non-linear parametrized partial differential equations, International Journal for Numerical Methods in Engineering 76 (2008) 27–55.

[35] S. Chaturentabut, D. Sorensen, Nonlinear model reduction via discrete empirical interpolation, Society for Industrial and Applied Mathematics 32 (5) (2010) 2737–2764.

[36] D. Ryckelynck, A priori hypereduction method: an adaptive approach, International Journal of Computational Physics 202 (1) (2005) 346–366.

[37] P. Astrid, S. Weiland, K. Willcox, T. Backx, Missing point estimation in models described by proper orthogonal decomposition, IEEE Transactions on Automatic Control 53 (10) (2008) 2237 – 2251.

[38] K. Carlberg, C. Farhat, J. Cortial, D. Amsallem, The gnat method for nonlinear model reduction : Effective implementation and application to computational fluid dynamics and turbulent flows, Journal of Computational Physics (242) (2013) 623–647.

[39] C. Farhat, P. Avery, T. Chapman, J. Cortial, Dimensional reduction of nonlinear finite element dynamic models with finite rotations and energy-based mesh sampling and weighting for computational efficiency, International Journal for Numerical Methods in Engineering 98 (9) (2014) 625–662.

[40] P. Ladevèze, A computational technique for the integrals over the time-space domain in connection with the latin method (in french), Tech. Rep. 193, LMT Cachan (1997).

[41] R. Glowinski, P. Tallec, Augmented Lagrangian and operator-splitting methods in nonlinear mechanics, SIAM studies in applied mathematics, Society for Industrial and Applied Mathematics, 1989.

[42] C. Heyberger, P.-A. Boucard, D. Néron, Multiparametric analysis whitin the proper generalized decomposition framework, Computational Mechanics 49 (3) (2011) 277–289.

[43] N. Relun, D. Néron, P. Boucard, A model reduction technique based on the pgd for elastic-viscoplastic computational analysis, Computational Mechanics.

[44] J. V. Aguado, F. Chinesta, A. Leygue, E. Cueto, A. Huerta, Deim-based pgd for parametric nonlinear model order reduction, in: J. P. M. de Almeida, P. Diez, C. Tiago, N. Parés (Eds.), VI International Conference on Adaptive Modeling and Simulation, ADMOS, 2013.

[45] F. Chinesta, A. Leygue, F. Bordeu, J. V. Aguado, E. Cueto, D. Gonzalez, I. Alfaro, A. Ammar, A. Huerta, Pgd-based computational vademecum for efficient design, optimization and control, Archives of Computational Methods in Engineering 20 (1) (2013) 31–59.