Politecnico di Milano

Facoltà di Ingegneria Industriale e dell'Informazione
Dipartimento di Scienze e Tecnologie Aerospaziali

Corso di Laurea Magistrale in Ingegneria Aeronautica

Reduced Order Methods for PDEs: a comparison between Proper Orthogonal Decomposition and Proper Generalized Decomposition

Advisors:
Prof. Simona PEROTTO
Prof. Gianluigi ROZZA

Co-Advisor:
Dr. Francesco BALLARIN

Author:
Luca Marco VALSECCHI
Matr. 799957

Anno Accademico 2014–2015

Contents

List of Figures

List of Tables

Abstract

Despite the progressive improvements in terms of simulation capabilities and techniques, some challenging problems still remain infeasible, or at least too much expensive, in most of the real world applications. Many techniques were developed in the last decades in order to reduce the computational costs, such as the Reduced Basis (RB) or the Proper Orthogonal Decomposition (POD) method. In this thesis a third technique, known as Proper Generalized Decomposition (PGD) method, is analysed as an alternative Reduced Order Modelling (ROM) technique. The PGD method is based on the assumption of separability for the unknown field, and it has demonstrated its capability in dealing with high dimensional problems. In particular, it represents a highly-performing method in the context of an offline-online splitting scheme, without the necessity of an *a priori* knowledge of the solution. It allows to compute, once and for all, a general convolution solution of a multidimensional framework offline – where model parameters can be set as extra-coordinates – and then to evaluate a new solution online – for new specific values of the coordinates, physical or parameters – at very low computational cost, achievable even on average performance platforms.

Sommario

Malgrado i progressivi sviluppi in termini di capacità computazionali di simulazioni e delle relative tecniche di approssimazione, alcuni problemi complessi rimangono ancora irrisolvibili o troppo costosi nella maggioranza delle applicazioni realistiche. Molte tecniche numeriche sono state sviluppate negli ultimi decenni al fine di ridurre il costo computazionale, come il metodo Reduced Basis (RB) o Proper Orthogonal Decomposition (POD). In questo lavoro di tesi un terzo metodo, noto in letteratura come Proper Generalized Decomposition (PGD), viene analizzato come tecnica alternativa di Reduced Order Modelling (ROM). Il metodo PGD si basa sull'assuzione di separabilità del campo incognito, e ne sono state dimostrate le potenzialità nell'ambito di problemi ad alta dimensionalità. Ne risulta un metodo molto performante in un contesto di separazione offline-online dello schema numerico, senza avere la necessità di una conoscienza *a priori* della soluzione. Il metodo PGD permette di calcolare, una volta sola, offline una soluzione di convoluzione generale di un problema nel contesto multidimensionale – dove i parametri del modello sono visti come coordinate extra – e di valutare poi una nuova soluzione online – per specifici valori delle coordinate, siano esse fisiche, temporali o parametri – in tempo reale a basso costo, anche mediante l'utilizzo di piattaforme a medie prestazioni.

Acknowledgements

In primo luogo un grande ringraziamento va ai miei relatrori, la Professoressa Simona Perotto e il Professor Gianluigi Rozza, per la disponibilità dimostrata durante tutto lo sviluppo di questo lavoro di tesi. Inoltre, voglio ringraziare in modo particolare il mio correlatore, il Dott. Ing. Francesco Ballarin, per aver supportato tecnicamente lo sviluppo delle implementazioni e aver sopportato lo sviluppatore.

Un grande riconoscimento in generale a tutto lo staff del gruppo SISSA Mathlab per avermi offerto disponibilità e possibilità di crescita in un ambiente eccellente. In particolare voglio ringraziare Ale, mio ex coinquilino all'arrivo a Trieste e mio complice in questa esperienza alla SISSA, e non solo.

Saluto e ringrazio il PoliMi e tutti gli amici qui conosciuti, come Paul, Roby, Fra e Luca, che hanno contribuito in modo più o meno attivo al raggiungimento di un obiettivo che all'inizio sembrava tanto lontano a tutti.

Un ovvio ma non scontato grande ringraziamento va alla mia famiglia per avermi sempre supportato e soprattutto sopportato, e per avermi aiutato, ma non troppo, a trovare la mia strada. Un particolare grazie a mia sorella Chiara per avermi sempre e incondizionatamente supportato.

Ultimo ma non per importanza, un grazie per tutto quanto va alla mia ragazza Roberta, parte di qeuesta mia bella esperienza a Trieste ma non solo, con la quale ho condiviso tanto e con la quale condividerò ancora. Grazie del supporto e di aver condiviso con me un passo importante di un percorso di crescita professionale e personale.

Grazie a tutti coloro che hanno creduto in me e mi sono stati vicini, come Cio, Albo, Aurie e Dev. In generale grazie agli amici, vecchi o nuovi, lontani o vicini, per aver fatto parte di una conquista!

Introduction 1

1.1 Motivations

Scientific computing and numerical simulations have been gaining more and more importance during last decades. The progressive growth of modelling techniques, numerical analysis and discretization techniques leads to the possibility of simulating several phenomena in science and engineering. However, the main limitations of numerical simulations lie into the computational power despite of the increasing availability due to the complexity of many models or to specific constraints such as real-time processing.

The amount of applications where the increasing computational costs represent the main issue ranges over different applications. High dimensional models are increasingly common, and they are used in order to describe structure and mechanics of materials in quantum chemistry [10], or even in the kinetic theory description of complex fluids [7]. Indeed, modelling social dynamics and economic systems requires high dimensionality, as well as the description vehicular traffic or congested and panic flows phenomena. Surely, there are many applications where complex models could be defined, even if the practical applications are limited by the so-called *curse of dimensionality* which characterizes the high dimensional models. For example, if a D-dimensional problem is considered, wherein $\mathcal{M}$ nodes are used in order to discretize each coordinate space, the resulting number of degrees of freedom reaches the number of $\mathcal{M}^D$. In case of a problem with $\mathcal{M} \approx 10^3$ – a very coarse description – and $D \approx 30$ – relatively simple parametric models – the numerical complexity results 10^{90} which is a very large number, specially considering that the total number of elementary particles in the universe is presumed to be on the order of 10^{80}. Even if the model dimensions could be considered as one of the main issue related to not affordable computational costs, the growing necessity

of interfacing with *real-time* applications is more and more frequent thanks to the increasing technology developments. In these cases it is evident the necessity of quick response and then high computational performances, even using low-performance platforms such as smartphones. Moreover, interfacing with problem sensitivity analysis with respect to parameters or considering optimization problems under constrains – such as in the case of optimal control or shape optimization – leads to the necessity of the direct computation of a huge number of solutions – i.e. *many-query* simulations – of the concerned model for particular values of the problem parameters [31]. Overall, there are different more scenarios where the computational effort appears to be a discriminant for the feasibility of the considered approach. A particular attention is deserved to the numerical methods for Computational Fluid Dynamics (CFD) which are essential in many engineering applications [30]. These range among different applications, from aerodynamics in the aeronautical or in the automotive fields, to hydrodynamic in the naval or in the industrial fields, coming to the more recent studies of physiological flows in the health care field [4].

Despite the progressive improvements attained by simulation capabilities and techniques, some challenging problems remain infeasible, or at least too much expensive for most of applications, possibly involving up to $\mathcal{O}(10^6\text{-}10^9)$ degree of freedom and several days of CPU time also on parallel hardware. Indeed, classical numerical approaches require too much computational efforts making both *real-time* and *many-query* simulations unaffordable. For these reasons, it has become necessary to develop suitable ***Reduced Order Models*** (ROM) with the aim to reduce both CPU efforts and storage capacity in order to open the horizons to new possibilities in the field of numerical simulations.

1.2 ROM: State of the Art

Models defining physical problems are often represented by a set of Partial Differential Equations (PDEs) with parametric dependence, which can either be physical or geometrical, leading to high computational cost simulations. The goal of ROM techniques is to capture the main behaviour of a model while reducing the computational effort with the constrain of controlling the propagation of the error between the reduced-order approximation and the full-order solution [31, 23]. Several techniques were developed in order to reduce the computational cost needed by simulations, such as greedy-based Reduced Basis (RB) methods or Proper Orthogonal Decomposition (POD) [13].

The earliest attempt of reduction approximation can be considered the

truncated Fourier series (1807) for the purpose of solving the heat equation in a metal plate [19], where a small number of trigonometric functions or modes were used in order to approximate the solution. During the last three decades, several investigations on theoretical approaches and numerical methods in the context of ROMs have allowed to approach many fluid dynamics problems with the aim of bypassing the computational effort arising from growing demand of applications and the complexity of the involved models. The main idea standing at the basis of the reduction strategies is that the behaviour of a system is often well described by a small number of dominant *modes*. A general solution is then computed by several methods as a *convolution solution* involving different coordinates and/or parameters and obtained retaining the dominant modes. Subsequently this general solution is particularized specifying the values of the coordinates and/or parameters. This kind of approach allows the *offline-online splitting* of the solution process and the consequent low-cost computation in the (online) evaluation process, giving the chance to meet the requests of high-response algorithms in the real-time context as well as permit several multi-query applications. The consequent necessity of a reduced computational cost not only leads to an academic interest on this methods but also should lead to an engineering pragmatic interest addressed to industrial applications.

The POD can be considered as the best known ROM technique, where the dominant modes retained are selected applying a Singular Value Decomposition (SVD) to a snapshot matrix containing the solutions of the system for different values of the parameters the model depends on. This method is usually applied to choose a new basis to build the ROMs typically for time-dependent problems, transforming the original problem unknowns into a set of variables with a reduced dimensionality – the POD principal components – which retain the most of the system energy. Firstly introduced in statistics as *principal component analysis* by Pearson (1901) [35], it is possible to appropriately refer to POD after the advent of electronic computers, when it was proposed as computational reduction tool by Lumley (1967) and Sirovich (1987) in the context of turbulent flows [45]. Considering last decades studies, it is interesting to recall the Kunisch et al. works [25, 26] and the Ravindran paper [40] in the context of parabolic equations, general equations in fluid dynamics and optimal control of fluids respectively, which are particularly significant in the aeronautical field.

During last decade an increasing effort has been aimed to the development of *a posteriori* error estimation procedures, based for instance on the well-known greedy algorithms. In this class we can classify the RB method which is usually applied to build bases for parameter-dependent problems. This popular strategy is based on the idea of selecting, at each step, the locally optimal element (refer to [30, 23] for more details on the algorithm). This approach

is less expensive compared with the POD method. Other methods for the reduction of the computational cost are base on low rank, balanced truncation, and other approaches (see [42, 6]).

A new generation of ROM simulation strategies consists in the ***Proper Generalized Decomposition*** (PGD) method. This method follows the idea of the *separated representation*, based on finite sum decomposition which leads to a drastic reduction of the number of degrees of freedom [14]. In the context of offline-online splitting and considering a generic modelled problem dependent on different parameters, the PGD method can be easily used to obtain a convolution solution for different values of the parameters during the offline stage. The parameters, which could either be space and time or any kind of model and geometrical parameters, are treated as generic coordinates for which a specific space is defined, allowing this approach to investigate on *combined time and parameter*-dependent problems. Separated representation was firstly introduced in the 80s by P. Ladevèze who proposed a first space-time separation for strong nonlinear problems [27, 29]. Later, the method was used in the context of stochastic high-dimensional problems [34].

1.3 Thesis Structure

This work investigates the performance of the PGD method. In particular, we develop a double comparison between the PGD solution, a reference full-order solution and a reduced-order convolution solution obtained with a POD-Galerkin method. A finite element approximation is implemented for the chosen benchmark problems using the *libMesh* open-source library [2] which provides a framework for numerical simulation on parallel architectures. The POD method implementation considered consists in a library[1] able to directly interface with the libMesh library. Similarly, the PGD method was implemented from scratch as an extension of libMesh, allowing to use the built in Finite Element Method (FEM) tools.

In Chapter 2 the POD-Galerkin method is described in order to define the reference reduced-order solution. In particular, a general approach is described to set a comparison tool for a generic problem, emphasizing the construction of the method instead of the application of the method to a particular problem (for which it is possible to find several different articles in the literature, e.g., [4]).

[1] Implemented by Francesco Ballarin, SISSA mathLab, International School for Advanced Studies, http://mathlab.sissa.it/, in the framework of the new open access library *ITHACA*

In Chapter 3 the PGD method is described. Firstly, the PGD offline stage is defined for a generic problem in weak formulation with the aim of providing a general description in order to underline the strengths of such an approach. After the definition of the separated representation which is the starting point of the method, the PGD progressive construction iterative routine is described as well as each of the issues related to it. Afterwards, boundary and initial conditions are treated in a general way, defining a problem which combines Boundary Value Problems (BVPs) and Initial Value Problems (IVPs), to outline the PGD approach for a generic space-time problem. Then, the possible parameter dependence is tackled. Finally, the PGD offline modulus is generalized to define the separated form of an analytic function in a multi-dimensional space, which represents one of the pragmatic key points in applying the method.

Then, the benchmark problems used as comparison tests are defined in Chapter 4. The choice is of defining preparatory problems with an increasing degree of complexity starting from a parametric transient diffusion equation in Section 4.2 which can be considered as one of the simplest time and parametric-dependent problems. For this problem, homogeneous boundary conditions are considered, without lack of generality, in order to focus on the construction steps of the PGD solver for the defined problem. Then, parametric Graetz problem is considered in Section 4.3. This increases the computational effort adding to the previous equation the term of transport. In the resulting equation no attention is focused on stability issues. Such issues can be considered by the reader as the object for a future investigation on the PGD method. Finally, in Section 4.4, a linear bidimensional vectorial problem is considered in order to take into account problems with a dimension greater than 1. This problem, can be seen as a simple problem in the context of the structural analysis, which represents an interesting field of research in the aeronautical engineering. For each of the chosen benchmark problems, the implementations of the PGD, POD and standard FE methods are performed and the results are compared to establish the performance of the PGD method.

In Chapter 5 the results of the analysed problems are commented to draw some conclusion on the thesis investigation and the detected issues. Moreover, possible future development are suggested.

This thesis has been developed in the framework of a collaboration between SISSA, International School for Advanced Studies, mathlab laboratory (Prof. G. Rozza's group) and Politecnico di Milano, MOX (Prof. S. Perotto's group). An internship at SISSA has been spent in 2015, thanks to the contribution of the Mathematics Area of SISSA.

Proper Orthogonal Decomposition 2

2.1 Introduction

The Proper Orthogonal Decomposition (POD) method is based on the idea that, in many cases, a problem solution can be defined in a subspace of smaller dimension with respect to the space dimension of the original full-order model. Indeed, it is possible to identify a certain number of dominant modes associated with most of the energy of the system. For this aim, the POD begins with the construction of a *snapshot matrix* generated by a discrete evaluation of the problem solution for different values of the problem parameters, as well as for the time coordinate. The snapshot set is determined using a costly, large-scale, high-fidelity method [9], e.g., finite elements, with the result of a reduced-order model built *a posteriori* by means of an already-computed discrete field. The original problem unknowns are transformed in a reduced dimensionality set of variable called *POD principal components*, where the reduced basis generated is obtained applying a SVD to the snapshot matrix. The POD basis is then given by the left singular vectors corresponding to the most meaningful singular values of the snapshot matrix.

2.2 POD-Galerkin Constructor: Offline Phase

In this work a POD-Galerkin ROM is considered as reference reduced-order approximation and it will be used as comparison.

Consider a generic differential problem defined by the operator $\mathcal{A}$ such that:

$$\mathcal{A}\Big(\mathbf{u}(\mathbf{x},\boldsymbol{\mu})\Big) = \mathbf{f}(\mathbf{x},\boldsymbol{\mu}) \quad \text{in } \Omega(\boldsymbol{\mu}) \tag{2.1}$$

where $\mathbf{f}$ is a given distributed source term. In (2.1) the domain Ω is defined as the Cartesian product between the spatial domain $\Omega_x \subset \mathbb{R}^d$, with d spatial dimension, and the domain associated with a generic set of parameters the spatial domain depends on. Denoting this set by $\boldsymbol{\mu} \in \Omega_{\boldsymbol{\mu}} \subset \mathbb{R}^P$, with P the number of parameters, any parameter may characterize either geometrical configurations or physical properties of the problem. The set space is in turn the Cartesian product between the spaces associated to each parameter. The domain is then defined as:

$$\Omega(\boldsymbol{\mu}) = \Omega_{\mathbf{x}}(\boldsymbol{\mu}) \times \Omega_{\boldsymbol{\mu}} \,. \tag{2.2}$$

It is evident the privileged role of the spatial coordinate $\mathbf{x}$ with respect to the parameter set $\boldsymbol{\mu}$, highlighted also in the expression of the unknown field $\mathbf{u}$. Indeed, in a POD framework the spatial coordinates can be viewed as "master" coordinates and the corresponding dominant modes are sought in order to retain the most energetic information. As explained in Chapter 3, in the PGD there are no privileged coordinates and each variable – either spatial, temporal or a parameter – is considered as a "generic" coordinate.

In order to derive the algebraic formulation of (2.1) we firstly write the weak form of the problem. To do this, we define the unknown space V over a reference spatial domain $\Omega_{\mathbf{x}}$ such that the parametrized space domain $\Omega_{\mathbf{x}}(\boldsymbol{\mu})$ can be obtained as the image of $\Omega_{\mathbf{x}}$ through a parametrized map $T(\cdot;\boldsymbol{\mu}) : \mathbb{R}^d \to \mathbb{R}^d$, i.e., $\Omega_{\mathbf{x}}(\boldsymbol{\mu}) = T(\Omega_{\mathbf{x}};\boldsymbol{\mu})$. The space V is a Hilbert space equipped with a norm associated with its inner product $(\cdot,\cdot)_V$.

The weak formulation is then obtained by multiplying problem (2.1) by a test function $\mathbf{v} \in V$, integrating over the spatial domain – eventually integrating by parts the diffusion term in order to reduce the solution regularity order required – and then tracing back to the reference domain $\Omega_{\mathbf{x}}$. For example, if a linear operator is considered, then a bilinear form is associated with the operator, as well as a functional is associated with the source term, leading to the weak formulation of the problem:

$$\text{find } \mathbf{u} \in V \text{ such that } a(\mathbf{u},\mathbf{v};\boldsymbol{\mu}) = F(\mathbf{v};\boldsymbol{\mu}) \quad \forall \mathbf{v} \in V \,. \tag{2.3}$$

Introducing then a finite-dimensional space $V_h \subset V$ of dimension N^h, being $h > 0$ the computational mesh size, a Galerkin-Finite Element (FE) approximation is considered. Then, denoting by $\{\varphi_i^h\}_{i=1,\dots,N^h}$ a (Lagrangian)

basis of V_h, the FE approximation of (2.3) reads

$$\text{find } \mathbf{u}_h \in V_h \text{ such that } a(\mathbf{u}_h, \mathbf{v}_h; \boldsymbol{\mu}) = F(\mathbf{v}_h; \boldsymbol{\mu}) \quad \forall \mathbf{v}_h \in V_h \,. \tag{2.4}$$

Problem (2.4) can be written as the equivalent algebraic linear system

$$A(\boldsymbol{\mu})\, \underline{\mathbf{u}}(\boldsymbol{\mu}) = \underline{\mathbf{f}}(\boldsymbol{\mu}) \tag{2.5}$$

for the vector of the unknowns $\underline{\mathbf{u}}(\boldsymbol{\mu}) = (u_h^{(1)}, \dots, u_h^{(N^h)})^T$ and where

$$\Big(A(\boldsymbol{\mu})\Big)_{ij} = a(\varphi_j^h, \varphi_i^h; \boldsymbol{\mu})\,; \qquad \Big(\underline{\mathbf{f}}(\boldsymbol{\mu})\Big)_i = F(\varphi_i^h; \boldsymbol{\mu})\,. \tag{2.6}$$

Then, we introduce the inner product matrix X for the space V_h whose elements are given by

$$\Big(X\Big)_{ij} = (\varphi_j^h, \varphi_i^h)_{V_h} \quad \in \mathbb{R}^{N^h \times N^h}\,, \tag{2.7}$$

$(\cdot,\cdot)_{V_h}$ denoting the discrete inner product defined over the space V_h.

The POD here described aims to find a reduced order space $V_N \subset V_h$ collecting a combination of "snapshots", i.e., of FE solutions. We denote by $\Sigma = \boldsymbol{\mu}^1, \dots, \boldsymbol{\mu}^n \subset \Omega_{\boldsymbol{\mu}}$ a training sample of n points in $\Omega_{\boldsymbol{\mu}}$ such that it is possible to define the ***snapshot matrix*** as:

$$S = [\underline{\mathbf{u}}_h(\boldsymbol{\mu}^1) | \dots | \underline{\mathbf{u}}_h(\boldsymbol{\mu}^n)] \in \mathbb{R}^{N^h \times n} \tag{2.8}$$

which demands the computation of n FE problems, where $n < N^h$.

A POD basis for the unknown field $\mathbf{u}$ can be obtained considering the Singular Value Decomposition (SVD) of the following matrix:

$$X^{1/2} S = U \Sigma W^T\,, \tag{2.9}$$

where $U \in \mathbb{R}^{N^h \times n}$ is a matrix containing the first n left singular vectors, $W \in \mathbb{R}^{n \times n}$ is an orthogonal matrix containing the right singular vectors, and $\Sigma \in \mathbb{R}^{n \times n}$ is a diagonal matrix containing the *singular values* of S. For any $N < n$ the POD basis is given by the functions $\boldsymbol{\varphi}_i$ for $i = 1 : N$ which are the with the first N columns of U (left singular vectors). We underline that the left multiplication by $X^{1/2}$ guarantees the basis to be orthogonal with respect to the inner product $(\cdot,\cdot)_{V_h}$, while considering only the snapshot matrix S would lead to a basis orthogonal with respect to the product defined in $\mathbb{R}^{N^h}$. Then, it is possible to define the POD basis matrix as

$$Z = [\underline{\boldsymbol{\varphi}}_1 | \dots | \underline{\boldsymbol{\varphi}}_N] \in \mathbb{R}^{N^h \times N} \tag{2.10}$$

and the basis functions of the space are $V_N = span\varphi_1, ..., \varphi_N$.

A different approach – which however might conduce to a worst conditioning – leads to solve eigenproblem for the two-point correlation matrix

$$C = S^T X S \in \mathbb{R}^{n \times n} \tag{2.11}$$

and to define the POD basis for the space V_N as the set constituted by the first N eigenvectors of (2.11) expressed as

$$\underline{\boldsymbol{\varphi}}_j = \frac{1}{\sqrt{\lambda_j}} S \underline{\boldsymbol{\psi}}_j \quad j = 1, \ldots, N \tag{2.12}$$

being

$$C \underline{\boldsymbol{\psi}}_j = \lambda_j \underline{\boldsymbol{\psi}}_j \quad j = 1, \ldots, N \tag{2.13}$$

and

$$\lambda_j = (\sigma_j)^2 . \tag{2.14}$$

It is easy to verify that the basis functions are by construction orthonormal. For more detailes the readers can refer to [45].

Notice that the dimension N of the reduced space is chosen as the smallest integer for which the "energy" of the retained modes

$$E(\phi_1, \ldots, \phi_N) = \frac{\sum_{j=1}^{N} (\sigma_j)^2}{\sum_{j=1}^{n} (\sigma_j)^2} \tag{2.15}$$

is greater than $1 - \epsilon_{tol}$, for a prescribed (small) tolerance ϵ_{tol}.

The interested reader can refer to [23] and [22] for more details about the method.

2.3 Affine Parametric Dependence

A key point for an efficient ROM evaluation is the capability to decouple the construction stage of the reduced-order space (once and offline) from the parametric evaluation stage (online), i.e., the so called offline/online decomposition. In order to reach this goal, further assumptions on the linear algebraic system (2.5) are needed, requiring that matrices and vectors fulfil the assumption of ***affine parametric dependence*** [41]. Thus, they can be written in a form

like

$$A(\boldsymbol{\mu}) \;=\; \sum_{q=1}^{Q_A} \Theta_q^A(\boldsymbol{\mu})\, A^q\,; \quad \underline{\mathbf{f}}(\boldsymbol{\mu}) \;=\; \sum_{q=1}^{Q_f} \Theta_q^f(\boldsymbol{\mu})\, \underline{\mathbf{f}}^q \tag{2.16}$$

and similarly for the other terms. We remark that the parametric dependence is only in the $\Theta_q^A(\boldsymbol{\mu}) \in \mathbb{R}$ and $\Theta_q^A(\boldsymbol{\mu}) \in \mathbb{R}$ functions while the coefficients $A^q \in \mathbb{R}^{N^h \times N^h}$ and $\underline{\mathbf{f}}^q \in \mathbb{R}^{N^h}$ are computed once. When dealing with affine parametrization, expressions like (2.16) are straightforward to be obtained, while, in case of nonaffine parametrizations, an appropriate affine expansion can be recovered by means of the Empirical Interpolation Method (EIM) [5].

2.4 POD Online Phase

As shown in section Section 2.2, an orthonormal set of basis functions for the space V_N is obtained performing a POD over each set of snapshots. Then, a reduced-order approximation for the unknown field $\underline{\mathbf{u}}_N$ is obtained such that

$$\underline{\mathbf{u}}(\boldsymbol{\mu}) \;\approx\; Z\, \underline{\mathbf{u}}_N(\boldsymbol{\mu}) \tag{2.17}$$

where the reduced-order approximation $\underline{\mathbf{u}}_N \in \mathbb{R}^N$ is determined performing a ***Galerkin projection***. In order to achieve this projection, the *residual* associated with $\underline{\mathbf{u}}_N$ is imposed to be orthogonal to the columns of the basis matrix (2.10), i.e.,

$$\left[Z^T\right] \left[\underline{\mathbf{f}}(\boldsymbol{\mu}) - A(\boldsymbol{\mu})\, Z\, \underline{\mathbf{u}}_N(\boldsymbol{\mu})\right] \;=\; \left[\underline{\mathbf{0}}\right]. \tag{2.18}$$

Thus, once the reduced basis is built – during the *offline stage* – for any new value $\boldsymbol{\mu}$ of the parameter in the set $\Omega_{\boldsymbol{\mu}}$ the reduced-order approximation is obtained solving – at the ***online stage*** – the following reduced-order problem

$$A_N(\boldsymbol{\mu})\, \underline{\mathbf{u}}_N(\boldsymbol{\mu}) \;=\; \underline{\mathbf{f}}_N(\boldsymbol{\mu}) \tag{2.19}$$

where the N-th reduced terms are defined as

$$A_N(\boldsymbol{\mu}) \;=\; Z^T\, A(\boldsymbol{\mu})\, Z \quad \in \mathbb{R}^{N \times N} \qquad\qquad \underline{\mathbf{f}}_N(\boldsymbol{\mu}) \;=\; Z^T\, \underline{\mathbf{f}} \quad \in \mathbb{R}^N\,. \tag{2.20}$$

Proper Generalized Decomposition 3

3.1 Introduction

The Proper Generalized Decomposition (PGD) method can be viewed as an *a priori* reduced-order approximation, i.e. which does not need to know the solution of the complete problem. Ideally, one would be able to assess the accuracy of the reduced-order solution and to define if a successive enrichment of the reduced approximation basis is needed. An approximated solution of a field can be often written via a separated representation involving few terms (e.g. a combination of shape functions in the framework of a finite element method). Reduced order methods, such as POD, demonstrate how a small number of basis functions $\Phi(\mathbf{x})$ could lead to a sufficiently accurate numerical solution. These functions are obtained by solving an eigenvalue problem associated to the *snapshots* of the complete solution computed for different values of the coordinates – such as physical and/or geometrical quantities. In particular, the more the field evolves smoothly, the more the magnitude of the ordered eigenvalues α_i decreases fast with the index i, and the fewer are the modes needed to approximate the evolution of the field with a desired accuracy. Defining a cutoff value ϵ in order to decide what eigenvalues α_i (and so the corresponding modes) are relevant, only a small number N of modes are usually retained (with $N \ll \mathcal{M}$, where $\mathcal{M}$ is the number of the discretization mesh nodes), such that $\alpha_i \geq \epsilon$, for $i \leq N$. Thus, in the case of a general space and a time dependent scalar field $u(\mathbf{x}, t)$, it is possible to write

$$u(\mathbf{x}, t) \approx \sum_{i=1}^{N} \Phi_i(\mathbf{x}) T_i(t) \,. \tag{3.1}$$

Equation (3.1) represents a natural *separated representation* and consists in

a finite sum decomposition: the solution is approximated by a sum of function products where each function depends on only one of the variables involved in the model. The use of this separated representation is the core of the PGD method where the solution is computed via an ***enrichment loop*** of the sum. It results that the n-th *enrichment step* provides the n-th term of the sum, considering known the previous $(n-1)$ terms. Then, the n-th approximation of the unknown field is

$$u^n(\mathbf{x},t) \;=\; \Phi_n(\mathbf{x})T_n(t) + \sum_{i=1}^{n-1} \Phi_i(\mathbf{x})T_i(t) \tag{3.2}$$

where it is clear that the unknown functions identify a ***nonlinear problem***. Ideally, the functions involved should be computed simultaneously by applying a suitable algorithm to guarantee robustness and optimality [14]. A suitable iterative process is then needed in order to compute each function of the nonlinear problem. This issue persists even in the case when the original problem is linear.

The first advantage in terms of performances is appreciable when considering, for example, a transient problem defined in a d-dimensional physical space. Use of a common incremental strategy with T time steps requires the solution of T d-dimensional problems, one for each time step. On the other hand, a space-time separated representation as in (3.1) leads to the solution of N d-dimensional problems for each nonlinear iteration p needed to compute each term of the finite sum (3.1). With the assumption of a constant number Q of nonlinear iteration for each term of the sum we have a total of $N \cdot Q$ d-dimensional problems to be solved. For many problems this total number is significantly less compared to the total number of problems to be solved with a common incremental method, i.e., $T \gg N \cdot Q$ (see Section 3.2.4).

This different approach allows to split the space-time problem into a series of decoupled Boundary Values Problems (BVP's) and Initial Value Problems (IVP's) thanks to the separated representation of the solution (see Section 3.2.4). This leads to different decoupled problems for each coordinate to be solved with different and independent strategies. Thus, the PGD can be viewed as an *outer shell* which organizes a series of solvers for the decoupled sub-problems allowing the use of the preferred – or most convenient – solver available for the particular BVP or IVP. For example, it is possible to solve a transient parametric diffusion equation by the use of first order finite differences for the IVP's and by the use of finite elements for solving the BVP's (see Section 4.2).

The optimality of the PGD method is not expected when the differential operator defining the model is non-symmetric, for example in the transient diffusion equation – as it contains a first order time derivative – in particular

with small diffusivity k. Indeed, the number of modes involved in the separated representation could increase significantly. Alternative PGD strategies can be used in order to alleviate this difficulty, as expressed in [14, p. 104]. Moreover, a possible simple approach consists in a *data post-compression* of the computed PGD solution by invoking once again the PGD, as described in Section 3.3.1. The reader can find more details on the separated representation in [20].

The PGD approach is developed in a context of *online-offline* splitting of a high dimensional model. In order to do that a *convolution solution* is computed once and offline, with respect to the variation of the value of each parameter, considered as a possible coordinate moving in its own space. Then, a particular solution – for specific values of the parameters – can be evaluated at low cost – online – allowing the computation on any also low performing platform – such as smartphones or tablets – or in high frequency applications – such as real time simulations – as described in [8, 33].

3.2 PGD Offline Phase

In order to provide a PGD constructor which computes a convolution solution of a problem, we firstly need to define the solution dependencies. Consider a problem dependent on a certain number of *physical coordinates* (i.e. space and time) and on a certain number of *parameters*, either geometrical or physical properties of the problem. In the PGD framework all the physical coordinates and parameters are denoted as generic ***coordinate***.

Thus, a generic linear problem is dependent on N_C coordinates defined in the spatial coordinate space Ω_x and time Σ_t, and in the space of a generic set of parametric coordinates $\boldsymbol{\mu}$, each defined in its own space Π_k, with $k = 1 : N_k$ being N_k the number of the components of $\boldsymbol{\mu}$. The solution $\mathbf{u}$ is thus defined in the space $\Omega = \Omega_x \times \Sigma_t \times \prod_{k=1}^{N_k} \Pi_k$, where the total number of coordinates is $N_C = N_k + 2$. For a more compact notation, the whole coordinates are grouped into the generic coordinate vector $\mathbf{x}$ where each coordinate is defined in its own space Ω_j, with $j = 1 : N_C$. Accordingly with the connotation of coordinates in the PGD context, each one of the N_C coordinates can either be physical or parametric, as well as scalar or vectorial. Indeed the physical space coordinate is denoted by $\mathbf{s}$ and it coincides with the first PGD coordinate, i.e., $\mathbf{x}_1 = \mathbf{s}$ and $\Omega_1 = \Omega_s$, while the time coordinate is $x_2 = t$ such that $\Omega_2 = \Sigma_t$, and so on.

Defining the problem operator $\mathcal{A}$ as a sum of N_L linear operators and

denoting by $\mathbf{f}$ the source term, it is possible to write the problem as:

$$\mathcal{A}(\mathbf{u}) = \sum_{l=1}^{N_L} \mathcal{A}^l(\mathbf{u}) = \mathbf{f} \quad \text{in } \Omega \tag{3.3}$$

with the addition of the desired boundary and initial conditions.

A weak formulation of problem (3.3) is firstly needed in order to derive its algebraic formulation. Denoting by U the unknown space defined over Ω, such that

$$U = \mathbf{H}^1_{0,\partial\Omega_s}(\Omega_s) \times \mathbf{L}^2(\Sigma_t \times \prod_{k=1}^{N_k} \Pi_k) \,, \tag{3.4}$$

the weak formulation can be obtained multiplying (3.3) for the test function $\mathbf{u}^*$ and integrating (by parts) over the whole domain Ω. The following weak formulation of the problem is obtained:

$$\text{find } \mathbf{u} \in U \text{ such that:} \quad \sum_{l=1}^{N_L} a^l(\mathbf{u}, \mathbf{u}^*) = F(\mathbf{u}^*) \tag{3.5}$$

where a^l represent the bilinear forms associated to the linear operators $\mathcal{A}^l$.

3.2.1 Separated Representation

The separated form of the solution consists in a finite sum of products of N_C functions, each one dependent on only one coordinate. Considering the solution approximation as a N-term finite sum, the PGD yields the separated form for the solution $\mathbf{u}^N$

$$\mathbf{u}^N(\mathbf{x}) = \sum_{i=1}^{N} \mathbf{U}^i_s(\mathbf{s}) \circ \mathbf{U}^i_t(t) \circ \cdots = \sum_{i=1}^{N} \bigodot_{j=1}^{N_C} \mathbf{U}^i_j(x_j) \,, \tag{3.6}$$

where, for the generic vectorial unknown field $\mathbf{u}$, the product sequence is defined as a sequence of Hadamard products (known as Schur product [17]), which consists in an element-wise product. In this way the representation of $\mathbf{u}$ results in a separation of each element of the unknown vector. With $\mathbf{U}^i_j$ we denote the functions of the separated representation depending on the j-th coordinate associated with the i-th term of the sum.

Then, the generic linear operator $\mathcal{A}$ can be defined in a separated form. Each one of the N_L linear operators in (3.3) is written as a product sequence which involves N_C linear operators $\mathcal{L}^l_j$ dependent on the corresponding coordinates,

with $j = 1 : N_L$. Then, it is possible to rewrite (3.3) as:

$$\sum_{l=1}^{N_L} \bigodot_{j=1}^{N_C} \mathcal{L}_j^l(\mathbf{U}_j) \;=\; \mathbf{f} \quad \text{in } \Omega \tag{3.7}$$

where the products involved in the sequences are defined as Hadamard products.

Consider a problem defined by the separated operator (3.7) and consider the unknown field, approximated with the separated representation (3.6). In virtue of the linearity of the bilinear forms of the weak problem (3.5), it is possible to write the bilinear form associated with the whole operator $\mathcal{A}$ for the N-term approximation $\mathbf{u}^N$ as:

$$a(\mathbf{u}^N, \mathbf{u}^*) \;=\; \sum_{i=1}^{N} \sum_{l=1}^{N_L} a^l\left(\bigodot_{j=1}^{N_C} \mathbf{U}_j^i(x_j), \bigodot_{j=1}^{N_C} \mathbf{U}_j^*(x_j) \right). \tag{3.8}$$

where the bilinear form a^l can be expressed by the means of the bilinear forms related to the whole linear operators $\mathcal{L}_j^l$, called a_j^l. Thus, we can define a product sequence intended as a sequence of convolution products, i.e., tensor products, such that

$$\bigotimes_{j=1}^{N_C} a_j^l(\mathbf{U}_j^i(x_j), \mathbf{U}_j^*(x_j)) = a^l\left(\bigodot_{j=1}^{N_C} \mathbf{U}_j^i(x_j), \bigodot_{j=1}^{N_C} \mathbf{U}_j^*(x_j) \right), \tag{3.9}$$

and we can rewrite (3.8) as

$$a(\mathbf{u}^N, \mathbf{u}^*) \;=\; \sum_{i=1}^{N} \sum_{l=1}^{N_L} a^l\left(\bigodot_{j=1}^{N_C} \mathbf{U}_j^i(x_j), \bigodot_{j=1}^{N_C} \mathbf{U}_j^*(x_j) \right). \tag{3.10}$$

The test function $\mathbf{u}^*$ can be defined in the general separated form, accordingly to (3.6), and it results

$$\mathbf{u}^*(\mathbf{x}) \;=\; \bigodot_{j=1}^{N_C} \mathbf{U}_j^*(x_j)\,. \tag{3.11}$$

The problem unknowns are represented by all the $\mathbf{U}_j$ each of one multiplied by the associated test function $\mathbf{U}_j^*$, as well as multiplied together, resulting in a nonlinear problem to be solved.

3.2.2 Progressive Construction

The separated approximation (3.6) is a sum of N function products, each involving N_C functions $\mathbf{U}_j^i(x_j)$, unknown *a priori*. The construction of the approximation is made by a ***successive enrichment loop*** where each unknown function product at the current step is computed, and the process continues until a defined accuracy is reached [15]. Thus, at the n-th enrichment step, the functions $\mathbf{U}_j^i(x_j)$ are known for $i < n$, from the previous steps, while $\mathbf{U}_j^n(x_j)$ are the unknown of the current step, for $j = 1 : N_C$. The solution (3.6) can be expressed in a form which underlines the n-th unknown terms:

$$\mathbf{u}^n(\mathbf{z}) \;=\; \prod_{j=1}^{N_C} \circ \mathbf{U}_j^n(x_j) \;+\; \sum_{i=1}^{n-1} \prod_{j=1}^{N_C} \circ \mathbf{U}_j^i(x_j)\,. \tag{3.12}$$

The nonlinear character of this unknown function products is the price to pay for the separated representation and it requires a ***nonlinear loop*** which implies m iterations in order to compute the unknown functions $\mathbf{U}_j^n(x_j)$, for each coordinate j, i.e., it requires a number of m iterations in order to reach convergence. It is important to point out that the nonlinearity resulting from the separated approach is independent from the nonlinear character of the operator $\mathcal{A}$, so that we have a nonlinear PGD problem even if the original problem is linear.

The main advantage of this construction process in terms of costs consists in the *splitting* of the problem into a number of sub-problems equal to the number of coordinates N_C, so that each problem is a low-dimensional problem defined in its own coordinate space and thus much less expensive. In fact, even if the computation of a solution is needed for each nonlinear step and for each of the N enrichment steps, the computational reduction is due to the fact that the number of the enrichment steps is typically low. Moreover, most of sub-problems are one-dimensional – for each parameter coordinate and for the time coordinate – and the d-dimensional space problem is time independent, resulting in a set of low-cost subproblems. Sometimes, it could be convenient to separate also the physical space [14, p. 15], such as in case of plate, shell or extruded geometries, as well as in case of composite laminate analysis or in the general case of dominant-direction contexts. A mere quantification of cost reduction can be obtained in the simple case of the same number of nodes $\mathcal{M}$ used to discretize each coordinate with a total number of N enrichment steps. The PGD leads to a total number of $N \cdot \mathcal{M} \cdot N_C$ unknowns, instead of the $\mathcal{M}^{N_C}$ degrees of freedom – the exponential dimensionality growing is the so-called *curse of dimensionality* – involved in standard mesh-based discretizations. Thus, for low N and high-dimensional N_C and high-resolution $\mathcal{M}$ approximations in the computational cost is significantly reduced.

A simple nonlinear iterative method to be used during the computation of the n-th enrichment step consists in a ***fixed point alternating direction strategy***, as proposed in [14, p. 27]. Accordingly to this algorithm, each unknown function $\mathbf{U}_j^n(x_j)$ is separately computed in the space of its coordinate, considering known and fixed the other coordinates.

> **Remark**
> *The advantage of applying the PGD separated approach to the problem resides in the ability to* split *the original problem with the consequent possibility to use different numerical methods for each subproblem.*

During the generic p-th nonlinear step each subproblem has to be sequentially solved considering the other subproblems at the last computed value, i.e. the value at the p-th step for the just solved subproblems and the value at the $(p-1)$-th step for the not-yet solved subproblems. The loop starts from an appropriate initial guess which must be chosen accurately. In fact, a null j-th solution, chosen in order to evaluate the corresponding problem, would lead to a null coefficient that multiplies the bilinear form of the considered subproblem, resulting into a singular problem. One possibility resides in choosing the initial condition as unitary for each known-considered function at the first enrichment step, and then use the separated solution of the previous enrichment $(n-1)$-th step as initial guess for the current n-th nonlinear loop. The practice suggests that this choice leads to a faster convergence of the nonlinear loop with respect to the choice of a unitary initial guess for each nonlinear loop. On the other hand, in case of strong non-symmetric operators this could lead to a worse result.

At the n-th enrichment step and p-th nonlinear step, in virtue of the linearity of the bilinear forms (3.10) and considering the solution representation (3.12), the associated bilinear form can be defined as:

$$a(\mathbf{u}^{n,p}, \mathbf{u}^*) \;=\; F(\mathbf{u}^*) \;-\; \sum_{i=1}^{n-1} a(\mathbf{u}^i, \mathbf{u}^*)\,. \tag{3.13}$$

which consists, at the p-th linearization step, of the nonlinear problem associated with the n-th enrichment step.

Every nonlinear loop consists in the computation of the whole unknowns, one for each coordinate subproblem. Thus, during each subproblem solution the bilinear form (3.13) has to be specified to the related coordinate. Then, the

test function associated to the j-th subproblem, i.e., $\bar{\mathbf{u}}_j^*$, is the multiplication between the test function $\mathbf{U}_j^*$ and the trial functions $\mathbf{U}_k^q$, for $k = 1 : N_C$ with $k \neq j$. The trial functions are solutions of the associated suproblems computed during th q-th nonlinear step. Note that with $\bar{\cdot}_j$ are denoted the quantities specified to the j-th subproblem, for which only the j-th component is considered unknown, as said just before. There are two possibilities for the choice of $\bar{\mathbf{u}}_j^*$:

- consider the solution of each subprobem at the previous $(p-1)$-th nonlinear step:

$$\bar{\mathbf{u}}_j^*(x_j) \;=\; \bigotimes_{\substack{k=1\\k\neq j}}^{N_C} \left(\mathbf{U}_k^{n,p-1}(x_k)\right) \circ \left(\mathbf{U}_j^*(x_j)\right); \tag{3.14}$$

- consider the solution associated to the previous nonlinear step for the not-yet-solved problems and the current p-th nonlinear step solution the just-solved problems:

$$\bar{\mathbf{u}}_j^*(x_j) \;=\; \bigotimes_{m=1}^{j-1} \mathbf{U}_m^{n,p}(x_m) \circ \mathbf{U}_j^*(x_j) \circ \bigotimes_{M=j+1}^{N_C} \mathbf{U}_M^{n,p-1}\,. \tag{3.15}$$

The two choices share the same spirit as the Jacobi and the Gauss-Seidel method, respectively, where in the first case the computation is based on all the previous values, while in the second one the just-computed values are exploited[32]. Approach (3.15) has been chosen resulting in a fast convergence of the method. Although this approach may not be easily parallelizable, the most expensive part from the computational point of view resides in solving each single subproblem, operation which can be often parallelized. For these reasons the second choice seems to be the smart one. Anyway, in both cases the only unknown resulting is the one associated with the n-th enrichment step and the j-th coordinate function, $U_j^n(x_j)$.

Similarly to (3.10), omitting the dependency coordinate x_j of the j-th coordinate functions for a more compact notation, the bilinear form can be written as:

$$a(\mathbf{u}^n, \mathbf{u}^*) \;=\; \sum_{l=1}^{N_L} \bigotimes_{j=1}^{N_C} a_j^l(\mathbf{U}_j^n, \mathbf{U}_j^*) \;+\; \sum_{i=1}^{n-1} \sum_{l=1}^{N_L} \bigotimes_{j=1}^{N_C} a_j^l(\mathbf{U}_j^i, \mathbf{U}_j^i) \tag{3.16}$$

where the only unknown is related to the n-th step. Thus, considering known the bilinear form evaluated at the previous enrichment steps, i.e., for $i < n$, and

underlining that the only unknown is related to the j-th coordinate problem of the p-th nonlinear step, it is possible to specify the bilinear form (3.16) with respect to the j-th subproblem. This provides a form $\bar{a}_j$ defined as:

$$\begin{aligned}\bar{a}_j(\mathbf{U}_j^{n,p}, \mathbf{U}_j^*) \;=\; & \sum_{l=1}^{N_L} \bigboxtimes_{m=1}^{j-1} a_m^l(\mathbf{U}_m^{n,p}) \otimes a_j^l(\mathbf{U}_j^{n,p}, \mathbf{U}_j^*) \otimes \bigboxtimes_{M=j+1}^{N_C} a_M^l(\mathbf{U}_M^{n,p-1}) \\ & + \sum_{i=1}^{n-1} \sum_{l=1}^{N_L} \bigboxtimes_{j=1}^{N_C} a_j^l(\mathbf{U}_j^i, \mathbf{U}_j^i)\end{aligned} \tag{3.17}$$

where the abuse of notation $a_k^l(\mathbf{U}_k^{n,q})$ is used to indicate bilinear forms with repeated input arguments in order to simplify the notation. Note, once again, that the bilinear form in (3.17) only depends on the j-th test function component, considering unknown the only $\mathbf{U}_j^{n,p}$.

Similarly to the separated solution in (3.6) and to the separated bilinear form in (3.16), the source function has to be considered in separated form as well (see Section 3.3), and analogously any other function involved in the problem, such as Dirichlet data or Neumann fluxes for the space subproblem and initial condition function for the time subproblem (see Section 3.2.5). For example, a source function which consists of a constant part $\mathbf{f}_C$ and of a non-constant part $\mathbf{f}_{NC}$ known via a N_F-terms of a finite sum

$$\mathbf{f}_{NC}(\mathbf{x}) \;=\; \sum_{i=1}^{N_F} \bigodot_{j=1}^{N_C} \mathbf{F}_j^i(x_j)\,, \tag{3.18}$$

leads to the separated functional:

$$F(\mathbf{u}^*) \;=\; \mathbf{f}_C \cdot \bigodot_{j=1}^{N_C} \mathbf{c}_j(\mathbf{U}_j^*) \;+\; \sum_{i=1}^{N_F} \bigboxtimes_{j=1}^{N_C} F_j^{NC,i}(\mathbf{U}_j^*) \tag{3.19}$$

where:

$$\mathbf{c}_j(\mathbf{U}_j^*) \;=\; \int_{\Omega_j} \mathbf{U}_j^* \mathrm{d}x_j\,; \qquad\qquad F_j^{NC,i}(\mathbf{U}_j^*) \;=\; \int_{\Omega_j} \mathbf{U}_j^* \cdot \mathbf{F}_j^i \mathrm{d}x_j\,. \tag{3.20}$$

Analogously to the bilinear form in (3.17), the functional specification for

the j-th subproblem becomes:

$$\begin{aligned}\bar{F}_j(\mathbf{U}_j^*) \;=\; & \mathbf{f}_C \cdot \Big(\prod_{m=1}^{j-1} \circ \mathbf{c}_m(\mathbf{U}_m^{n,p}) \circ \mathbf{c}_j(\mathbf{U}_j^*) \circ \prod_{M=j+1}^{N_C} \circ \mathbf{c}_M(\mathbf{U}_M^{n,p-1}) \Big) \\ & + \sum_{i=1}^{N_F} \Big(\prod_{m=1}^{j-1} \boxtimes F_m^{NC,i}(\mathbf{U}_m^{n,p}) \otimes F_j^{NC,i}(\mathbf{U}_j^*) \otimes \prod_{M=j+1}^{N_C} \boxtimes F_M^{NC}(\mathbf{U}_M^{n,p-1}) \Big)\end{aligned} \tag{3.21}$$

where the product sequences related to the constant source term are intended as a sequence of Hadamard products to be scalar multiplied for the constant source term $\mathbf{f}_C$, in order to obtain a correct reconstruction of the separated functional $F(\mathbf{u}^*)$.

The single p-th nonlinear step could be generalized as the solution of the whole N_C subproblems, where the j-th problem related to the coordinate x_j, is defined as:

$$\bar{a}_j(\mathbf{U}_j^{n,p}, \mathbf{U}_j^*) \;=\; \bar{F}_j(\mathbf{U}_j^*) \;-\; \sum_{i=1}^{n-1} \bar{a}_j(\mathbf{U}_j^i, \mathbf{U}_j^*)\,. \tag{3.22}$$

Remark
The total amount of problems is equal to the number of coordinates D. The dimension of each j-th subproblem is equal to the dimension of the considered coordinate space, $\dim(\Omega_j)$. *Therefore, it is evident that the maximum dimension of the single sub-problem results lower than the one obtained in the case of the full-order problem.*

The process continues in an iterative loop until a defined accuracy is reached. The evaluation of the accuracy is performed by a specific *nonlinear stopping criterion* and the end of the fixed point loop determines also the end of the n-th enrichment iteration, which ends with the assignments $U_j^n(x_j) \leftarrow U_j^{n,p}(x_j)$ for each $j \leq D$. The enrichment loop continues until a chosen *enrichment stopping criterion* is satisfied and its end determines also the end of the PGD offline phase.

However the optimality of the described process is not guaranteed because of the general presence of non-symmetric differential operators involved in the model and the presence of strong nonlinearities, advection terms or extra coordinates, which may cause the fixed point alternating direction strategy to converge too slowly or fail. In this case it is necessary to consider a different

strategy such as a *residual minimization*, as described in [16, 15], where an iterative alternating direction scheme is applied in order to minimize the square of the norm of the residual in a strong form. This approach was firstly described in the context of the LATIN method for structural problem [29, 15] and was successively treated via a more efficient strategy [36] and in a generalized procedure for multidimensional models [12]. Despite the fact the residual minimization approach is more efficient when the problem concerns non-symmetric operators, it is far from being optimal and the number of terms computed in the separated representation of the solution contains more terms than strictly needed.

3.2.3 Stopping Criteria

The PGD constructor leads to the solution of a nonlinear problem for each enrichment loop. Thus a nonlinear loop proceed until reaching a fixed point within a user-specified tolerance ϵ^{fp}. The fixed point loop ends when a stopping criterion error $\varepsilon^{fp}(p)$ is small enough, i.e. $\varepsilon^{fp}(p) < \epsilon^{fp}$. A possible ***nonlinear stopping criterion*** proposed in [14, p. 27] is to consider the error defined by:

$$\varepsilon_I^{fp}(p) = \frac{\| \bigodot_{j=1}^{N_C} \mathbf{U}_j^{n,p}(x_j) - \bigodot_{j=1}^{N_C} \mathbf{U}_j^{n,p-1}(x_j) \|}{\| \bigodot_{j=1}^{N_C} \mathbf{U}_j^{n,p}(x_j) \|} \tag{3.23}$$

where $\| \cdot \|$ is a suitable norm, for example the L^2-norm.

This particular stopping criterion underlines another motivation why a null initial guess for the fixed point loop is not suitable. In fact, in that case the residual computed by criterion (3.23) would not be defined due to the null denominator.

As mentioned in [20], another possibility lies in computing the stopping criterion error as:

$$\varepsilon_{II}^{fp}(p) = \sum_{j=1}^{N_C} \| \bar{\mathbf{U}}_j^{n,p}(x_j) - \bar{\mathbf{U}}_j^{n,p-1}(x_j) \|_j^2 \tag{3.24}$$

where $\bar{\mathbf{U}}_j^n$ denotes the normalization of the j-th function. This criterion allows to define a residual which is more sensitive to the local variations of the j-th coordinate, for each coordinate, instead of a global behaviour stopping criterion as in (3.23).

The experience suggests that the second stopping criterion is more accurate and it is monotonic, while the first is only decreasing in average but oscillating, leading to a slower convergence, less accuracy and less stability. In fact, we

can notice that the fist stopping criterion involves the normalized difference between two consecutive composed solutions (multiplication of the functions of each coordinate). Thus, this quantity is sensitive to the decrease of a product of more terms and the high difference – in norms – between two multiplied terms leads to high numerical errors.

On the other hand, the PGD constructor produces an enrichment sequence of function products until a defined accuracy is reached. The evaluation of the accuracy is performed by an appropriate ***enrichment stopping criterion*** which measures the n-step enrichment error $\varepsilon^{en}(n)$, so that the enrichment loop ends when $\varepsilon^{en}(n)$ is smaller than a user-defined tolerance ϵ^{en}, i.e., $\varepsilon^{en}(n) < \epsilon^{en}$. Following the possible choices proposed in [14, p. 30], some stopping criteria are proposed. Each of them is based on a suitable norm $\|\cdot\|$. Without loosing generality, in the following we adopt the L^2-norm – as in the case of the stopping criterion associated to the fixed point loop – leading to the evaluation of integrals of dimension less than or equal to three, as described in Section 3.2.

A first stopping criterion could be the relative weight of the last-computed mode with respect to the entire PGD expansion. Thus

$$\varepsilon_I^{en}(n) \;=\; \frac{\| \boxdot_{j=1}^{N_C} \mathbf{U}_j^n(x_j)\|}{\| \sum_{i=1}^{n} \boxdot_{j=1}^{N_C} \mathbf{U}_j^i(x_j)\|} \,. \tag{3.25}$$

Since $\mathbf{u}^n(\mathbf{x})$ is expressed in a separated form, its square will be also expressed in a separated form having $\frac{n(n+1)}{2}$ terms, involving the evaluation of $D + \frac{n(n+1)}{2} D$ integrals.

A similar, but less expensive, criterion is based on the relative weight of the last-computed mode with respect to the first one, i.e.

$$\varepsilon_{II}^{en}(n) \;=\; \frac{\| \boxdot_{j=1}^{N_C} \mathbf{U}_j^n(x_j)\|}{\| \boxdot_{j=1}^{N_C} \mathbf{U}_j^1(x_j)\|} \,. \tag{3.26}$$

This second criterion requires the computation of only $2 \cdot D$ integrals.

More accurate error estimators can be defined in terms of the residual $\mathcal{R}_n(\mathbf{u}^*)$ obtained from the bilinear form (3.10), i.e. considering unknown all the N_C coordinates x_j. This leads to the weak form of the residual:

$$\mathcal{R}_n(\mathbf{u}^*) \;=\; \sum_{i=1}^{n} \left(\sum_{l=1}^{N_L} \left(\bigotimes_{j=1}^{N_C} a_j^l(\mathbf{U}_j^i(x_j), \mathbf{U}_j^*(x_j)) \right) \right) \;-\; \bar{F}_j(\mathbf{U}_j^*(x_j)) \,. \tag{3.27}$$

3.2.4 Non-Incremental Time

As said, the PGD constructor allows to approximate the field $u(x_1, \ldots, x_D)$ with the separated representation (3.6) leading to D independent sub-problems. A space-time problem is considered as defined in the coordinates $\mathbf{s} \in \Omega_s$ and $t \in \Sigma_t$ in order to investigate the time-dependence approximation of the PGD problem. This approach leads to a ***non-incremental*** algorithm where the time dependence is decoupled from the space dependence, in contrast to a standard *incremental* method where the temporal evolution is associated to the spatial solution increment. The IVPs defined over the time space Σ_t have a computational complexity which is negligible compared to that of the BVPs associated with the physical space Ω_s, even if small time steps are considered. Moreover, in contrast to a standard *incremental* solution procedure, the separated representation allows to consider also big time steps without causing significant stability issues on the numerical scheme [14, p. 62].

In case of Q_n nonlinear iterations for each enrichment step, we have a total number of $Q = Q_1 + \cdots + Q_N$ decoupled problems to identify the N-term approximation (3.6). Numerical experiments suggest that Q_n rarely exceed ten while N is $\mathcal{O}(10)$. Thus, the complexity of the complete PGD solution results in few hundreds of BVP's in Ω_x. This is generally several orders of magnitude less than the total number of BVP's obtained with a standard incremental procedure.

It is important to emphasize that the time step choice for the IVPs integration is *independent* from the mesh size used in the BVPs problems. Indeed, the non-incremental approach leads to the possibility to bypass the stability CFL condition [43] that links the time step with the space mesh size. Obviously, if the space and/or time resolution of these decoupled problems is too poor, the PGD will compute an inaccurate solution, but this issue regards the convergence rather than stability on the whole procedure.

In conclusion, we can state that the non-incremental time approach leads to the possibility of choosing the desired numerical method for the treatment of the time dependence without any constrain related to the spatial discretization. This freedom in selecting the numerical schemes leads to a reduction of several orders of the number of BVPs.

3.2.5 Boundary and Initial Conditions

The separated nature of the PGD approach leads to consider decoupled problems with respect to each of the D coordinates. Analysing the spatial BVPs defined in Ω_s and the IVPs defined in Σ_t, it turns out to be not trivial to deal with the boundary and initial conditions associated with the original model.

They have to be properly related to the sub-problems generated by the PGD process.

In the following, a space-time problem with generic boundary and initial condition is considered. We introduce the Dirichlet data in a set of functions $\mathbf{g}_d$ defined on $\Gamma_{D_d} \times \Sigma_t$ with $\Gamma_{D_d} \subseteq \partial\Omega_s$, for each $d = 1 : N_D$, the Neumann fluxes in a set of functions $\mathbf{q}_h$ defined on $\Gamma_{N_h} \times \Sigma_t$ with $\Gamma_{N_h} \subseteq \partial\Omega_s$, for each $h = 1 : N_Q$. We consider a source term $\mathbf{f}$ defined in $\Omega_s \times \Sigma_t$ an initial condition $\mathbf{u}_0$ defined in Ω_s. The problem can be written as:

$$\begin{cases} \mathcal{A}\Big(\mathbf{u}(\mathbf{s},t)\Big) = \mathbf{f}(\mathbf{s},t) & \text{in } \Omega_s \times \Sigma_t & \\ \mathbf{u}(\mathbf{s},t) = \mathbf{g}_d(\mathbf{s},t) & \text{on } \Gamma_{D_d} \times \Sigma_t & \forall d = 1 : N_D \\ a^{ell}_{\partial\Omega_s}\big(\mathbf{u}(\mathbf{s},t)\big) = \mathbf{q}_h(\mathbf{s},t) & \text{on } \Gamma_{N_h} \times \Sigma_t & \forall h = 1 : N_Q \\ \mathbf{u}(\mathbf{s},0) = \mathbf{u}_0(\mathbf{s}) & \text{in } \Omega_s & \end{cases}, \tag{3.28}$$

where $a^{ell}_{\partial\Omega_s}$ denotes the elliptic part of the bilinear form – where the elliptic character is intended to be with respect to the spatial coordinate $\mathbf{s}$ – restricted to the boundary of the spatial domain Ω_s, i.e., after integrating by parts. Notice that the possible presence of more coordinates – such as problem parameters – is irrelevant for the assumptions we are going to formulate. In order to take into account non-homogeneous Dirichlet boundary conditions, a *lifting* approach is considered. Thus, the requirement on the Dirichlet data is that the functions $\mathbf{g}_d$ – which have to be strictly defined on the Dirichlet boundary Γ_{D_d} for each $d = 1 : N_D$ – have to be regular enough to guarantee the existence of the operator $\mathcal{A}(\mathbf{g}_d, t)$ – defined on the whole domain $\Omega = \Omega_s \times \Sigma_t$ – and thus have to be defined on the whole domain Ω as well. For the same reason, we require that they are regular enough to guarantee the same values at the interfaces between Dirichlet boundaries, if any. In other words, the functions $\mathbf{g}_d$ have to be extended to the whole domain Ω with continuity, i.e., $\mathbf{g}_d \in \mathbb{C}^0(\Omega)$. At the same time, the requirements on the initial condition is that, at the Dirichlet boundaries, it has the same value of the Dirichlet data, i.e. $\mathbf{u}_0(\mathbf{s} \in \Gamma_{D_d})$ which consists again in a continuity requirement. It follows that a ***regularity***

condition is demanded on $\mathbf{g}$ and $\mathbf{u}_0$ so that we have:

$$\begin{aligned} &\exists\, \mathbf{g}(\mathbf{s},t) \in \Omega \quad \text{such that} \\ &\begin{cases} \mathbf{g}(\mathbf{s},t) = \mathbf{g}_d(\mathbf{s},t) \\ \mathbf{u}_0(\mathbf{s}) = \mathbf{g}(\mathbf{s},0) \end{cases} \\ &\text{for } \mathbf{s} \in \Gamma_D \text{ with} \quad \Gamma_D = \bigcup_{d=1}^{N_D} \Gamma_{D_d} \end{aligned} \quad . \tag{3.29}$$

Note that this condition implies that the "extended Dirichlet boundary function" $\mathbf{g}(\mathbf{s},t)$ has to properly extend the Dirichlet data $\mathbf{g}_d$. Such function is the *lifting function*, and the simplest way to define it consists in computing the solution of a simple Laplace problem completed with the same boundary conditions as the original problem, in order to find a function regular enough to satisfy many linear operators. A possibility is to consider the solution of the following problem:

$$\begin{cases} \nabla^2 \mathbf{g} \;=\; 0 & \text{in } \Omega_s \times \Sigma_t \\ \mathbf{g}(\mathbf{s},t) \;=\; \mathbf{g}_d(\mathbf{s},t) & \text{on } \Gamma_{D_d} \times \Sigma_t \end{cases} \quad . \tag{3.30}$$

The regularity of the function $\mathbf{g}$ is subordinated to the existence of $\mathcal{A}(\mathbf{g}_d)$, so that problem (3.30) may differ to guarantee the constrain of regularity or to be less computationally expensive as possible.

Experience suggests that the assumptions (3.29) are sufficiently general to describe most of the problems of physical interest in engineering and industrial processes.

Is now possible to apply the ***lifting*** with a simple variable change, by rewriting $\mathbf{u}(\mathbf{s},t)$ as $\mathbf{u}(\mathbf{s},t) = \mathbf{v}(\mathbf{s},t) + \mathbf{g}(\mathbf{s},t)$, so that we obtain a problem similar to (3.28) but with homogeneous initial and boundary conditions, i.e.,

$$\begin{cases} \mathcal{A}\Big(\mathbf{v}(\mathbf{s},t)\Big) = \mathbf{f}(\mathbf{s},t) - \mathcal{A}\Big(\mathbf{g}(\mathbf{s},t)\Big) & \text{in } \Omega_s \times \Sigma_t \\ \mathbf{v}(\mathbf{s},t) = 0 & \text{on } \Gamma_{D_d} \times \Sigma_t \qquad \forall d = 1 : N_D \\ a^{ell}_{\partial\Omega_s}(\mathbf{v}(\mathbf{s},t)) = \mathbf{q}_h(\mathbf{s},t) & \text{on } \Gamma_{N_h} \times \Sigma_t \qquad \forall h = 1 : N_Q \\ \mathbf{v}(\mathbf{s},0) = 0 & \text{in } \Omega_s \end{cases} . \tag{3.31}$$

Note that the lifting approach guarantees that the initial solution is null, allowing to assume a null initial condition for the IVPs independently to the initial condition of the original problem.

The solution of the original problem (3.28) is then reconstructed as

$$\mathbf{u}(\mathbf{s},t) = \mathbf{v}(\mathbf{s},t) + \mathbf{g}(\mathbf{s},t) \tag{3.32}$$

$\mathbf{v}(\mathbf{s},t)$ being the solution to the homogeneous problem (3.31).

Seeking for a separated representation of the function $\mathbf{g}(\mathbf{s},t)$ with N_G terms (as in Section 3.3) and applying the PGD solver to compute the unknown field $\mathbf{v}(\mathbf{s},t)$ allow us to achieve an approximated solution of the original problem of the form

$$\mathbf{u}^N(\mathbf{s},t) \;=\; \sum_{i=1}^{N_G} \boxtimes_{j=1}^{D} \mathbf{G}^i_s(\mathbf{s}) \circ \mathbf{G}^i_t(t) \;+\; \sum_{i=1}^{N} \mathbf{V}^i_s(\mathbf{s}) \circ \mathbf{V}^i_t(t) \tag{3.33}$$

where the terms $\mathbf{G}^i_j(x_j)$ associated with the function $\mathbf{g}(\mathbf{s},t)$ ensure the satisfaction of the Dirichlet boundary conditions of the original problem (3.28), while the functions $\mathbf{V}^i_j(x_j)$ – coming from the enrichment process of the homogeneous problem (3.31) and thus null on the Dirichlet boundary $\Gamma_D(\Omega_s)$ – allow the complete solution (3.33) to verify both the partial differential equation and the natural (Neumann) boundary conditions (*cf.* [14, p. 40]). Then, the lifting function $\mathbf{g}$ coming from the non-homogeneous Dirichlet boundary conditions can be treated as part of the enrichment process and the reconstructed N-term solution can be written as:

$$\mathbf{u}^N(\mathbf{s},t) \;=\; \sum_{i=1}^{G+N} \mathbf{U}^i_s(\mathbf{s})\, \mathbf{U}^i_t(t) \tag{3.34}$$

If the problem depends on more coordinates and not only on the spatial and temporal ones, the considerations are the same with the only difference

that each term of the function products in the previous separated forms are multiplied by a function dependent on each other coordinate x_j, with $j = 3 : N_C$. Considering, for example, (3.34) the summed products are $\prod_{j=1}^{N_C} \mathbf{U}_j^i$ for $i = 1 : G + N$. In particular, if the lifting function $\mathbf{g}(\mathbf{s}, t)$ is not dependent on x_j, the term related to that coordinate does not provide any contribution to the definition of the function itself, and thus it is possible to take into account that coordinate setting $\mathbf{G}_i^j(x_j) = 1$, for each finite sum term.

When Neumann boundary conditions are specified, we integrate by parts the bilinear form and introduce the Neumann fluxes in the formulation. Seeking for a separated representation of the functions $\mathbf{q}_h$ in (3.28) (as in Section 3.3) each defined by N_{Q_h} terms, a linear form q associated with $a_{\partial\Omega_s}^{ell}$ is defined similarly to (3.10) via a separated representation:

$$q(\mathbf{u}^*) = \sum_{h=1}^{N_Q} \sum_{i=1}^{N_{Q_h}} \prod_{j=1}^{D} q_j^{h,i}(\mathbf{U}_j^*) \tag{3.35}$$

where:

$$q_j^{h,i}(\mathbf{U}_j^*) = \int_{\Omega_j} \mathbf{U}_j^* \mathbf{Q}_j^{h,i} \mathrm{d}x_j \tag{3.36}$$

and $\mathbf{Q}_j^{h,i}$ are the functions defining the separated representation of the Neumann flux functions $\mathbf{q}_h$.

In case of a generic differential operator defining the problem, the process is the same with only additional care to treat all the boundary terms as shown before.

3.2.6 *Extra Coordinates*

In the PGD framework the notion of *coordinate* is enlarged to any parameters which may characterize the problem, either in terms of geometrical configuration or physical properties. In this work some property parameters are taken into account in the benchmark tests. This is the case of the diffusivity parameter k, the reaction constant c, and the transport velocity vector $\mathbf{b}$ characterizing for instance a standard advection-diffusion-reaction problem. The parameters are set as new coordinates of the problem. It is also possible to introduce boundary and initial conditions as new coordinates, as well as the geometrical domain, by means of different approaches. In [15, Chap. 2.3] it is presented an application of the *finite element interpolation* in order to compute convolution solutions with respect to boundary and initial conditions. This approach is suitable for frameworks where the space subproblem solution is computed

by a finite element solver and there is a coupling between different problems defined on the same mesh discretizing the domain. Indeed, in this case we have that the "function-coordinate" of a problem is the output of another problem, giving the exact nodal values needed by the finite element interpolation. In [15, Chap. 19] it is presented an approach to change the computational domain, where the spaces on which to consider geometrical parametrizations are referred to reference domains by means of a coordinate transformation. This approach results suitable for different geometrical parametrization methods defining the deformation of the domain – such as Free-Form Deformation [44, 24], Radial Basis Functions [18], Inverse Distance Weighting [46, 47] or its reduced counterpart – and this could be useful in the context of the Fluid Structure Interaction (FSI) and shape optimization problems.

3.3 Separated Approximations of Functions

In order to take into account a non-constant source term or, generically, a non-constant function – such as non-homogeneous essential boundary conditions and initial condition, or natural boundary conditions (see Section 3.2.5) – it is possible to use PGD as a tool for finding a separated representation of the function.

The procedure explained in the following is referred to a scalar problem without lack of generality. Consider the simple algebraic problem for the unknown scalar field u defined as

$$u(x_1, \dots, x_D) = f(x_1, \dots, x_D) \tag{3.37}$$

defined in $\Omega = \Omega_1 \times \cdots \times \Omega_D$, and apply the PGD constructor as in Section 3.2 to (3.37) in order to obtain a separated approximation of a given function $f(x_1, \cdots, x_D)$.

The resulting N-term approximation of the function $f(x_1, \cdots, x_D)$ reads

$$f^N(x_1, \cdots, x_D) = \sum_{i=1}^{N} \prod_{j=1}^{D} F_j^i(x_j)\,. \tag{3.38}$$

In this way, it is possible to define a integral form of the subproblem to be solved for each coordinate involved in the approximation process. The bilinear form associated with the linear operator defining the integral formulation of

the problem reduces to

$$a^N(u^N, u^*) = \sum_{i=1}^{N} \prod_{j=1}^{N_D} \int_{\Omega_j} U_j^N U_j^* \mathrm{d}x_j \tag{3.39}$$

and the functional of the right hand side of the formulation becomes

$$F(u^*) = \int \_\Omega_1 cdots \int_{\Omega_D} f(x_1, \ldots, x_{N_D})\, \mathrm{d}x_1 \ldots \mathrm{d}x_{N_D} \,. \tag{3.40}$$

Note that the functional requires a multi-space integration over the whole domain Ω. This cause the evaluation of a huge number of integrals, which becomes computationally expensive when the problem dimensionality increases. For example, for a space-time separated problem the functional specified to the space subproblem is

$$\bar{F}_x(U_x^*) = \int_{\Omega_x} U_x^* \int_{\Sigma_t} U_t^{n,p-1} f(\mathbf{x}, t)\, \mathrm{d}t\, \mathrm{d}\mathbf{x} \tag{3.41}$$

which implies to evaluate the function $f(\mathbf{x}, t)$ and integrate it over the Σ_t domain for each of the $\mathbf{x}_k$ points in the Ω_x domain.

Then, it is possible to define the j-th subproblem in the weak form in order to apply the PGD strategy and compute the separated form of function (3.22). It is easy to understand that the number of integral evaluations increases exponentially with the increase of the number of coordinates involved in the separation process [14, Sec. 3.3].

It is possible to assume that, in many cases, the function which requires to be defined in a separated form in order to define the PGD constructor can be considered dependent only on a certain number of coordinates. For example, in case of a transient diffusion problem with the diffusivity parameter as extra coordinate, it is reasonable to fix a context where the source function can be considered only dependent on space and time. If a separated representation depending on n_D variables is computed with this kind of assumption, with n_D less than the number of coordinates for the PGD framework, N_D, then it suffices to impose unitary value for the component of the sum (3.38) associated with the missing $N_D - n_D$ coordinates, which the considered function was not considered to depend on. In this way, it is possible to reduce the number of integral evaluations and thus the numerical complexity.

Another possibility lies in considering a framework where the behaviour of the function to be approximated in separated form is assumed to be a combination of some separated functions, in a FEM enrichment-like procedure as explained in [15, Cap. 8.1]. These functions $f_k(x_1, \cdots, x_D)$ are depen-

dent on all the coordinates but are supposed known in their separated form. Their combination, for some unknown coefficients α^k, leads to the following approximation of the function

$$f(x_1, \cdots, x_{N_D}) \approx f^{sep}(x_1, \cdots, x_{N_D}) = \sum_{k=1}^{N_A} \alpha^k \sum_{i=1}^{N_{F,k}} \prod_{j=1}^{N_D} F_j^{k,i}(x_j) \tag{3.42}$$

where $F_j^{k,i}$ are the components of the known separated form of the functions f_k. Once the functions are chosen, i.e., fixed the number N_A of functions and their separated form $N_{F,k}$ components, with $k = 1 : N_A$, it is possible to define the best N_A coefficients α_k which minimize a certain error, for example by the mean of a *least square minimization* of the residual. Moreover, it is possible to consider an approach such as (3.42) where the separated functions are defined as function series – for example, polynomial functions – truncated at the term N_A a priori unknown. Even if this approach is not suitable for generic functions – particularly if not enough smooth – it could be versatile for some cases where the function to be approximated can be considered as non-arbitrarily complex.

Finally, it is interesting to note that some techniques like *Empirical Interpolation Method* (EIM) [5] or its discrete counterpart, the Discrete Empirical Interpolation Method (DEIM) [11], can be useful in order to reduce the computational effort for the definition of a separated approximation of a function.

3.3.1 *"A Posteriori" Data Compression*

The PGD solver allows the construction of an approximated solution in separated form (3.1) and can be viewed as an *on-the-fly* compressed representation of the model solution. In case of a 2D model involving a symmetric and coercive differential operator, the number N of modes produced by the PGD constructor corresponds to the number obtained by applying a *singular value decomposition* (SVD) which consists in an optimal separated representation [14, p. 47]. In particular, when the differential operator involved in the model is non symmetric, the separated approximation obtained is suboptimal and the number of terms in the finite sum is higher than the one obtained by applying the SVD (in 2D). So that an *"a posteriori" decomposition* is needed in order to gain a "better optimality" of the solution, even if the idea of the offline computation and online evaluation makes the optimality not crucial. Thus, with respect of an hypothetical optimal solution involving N^{opt} terms with a computed number N of terms, with $N \gg N^{opt}$, one can compute an "a posteriori" data compression in order to reduce the computational cost of the online phase as much as possible. This may be crucial in real-time applications

where high response frequency can be required. Then, the solution of the problem can be approximated by

$$u(x_1, \ldots, x_D) \approx \sum_{i=1}^{N} \prod_{j=1}^{D} U_j^i(x_j) \tag{3.43}$$

so that one looks for an enhanced representation $u^{enh}(x_1, \ldots, x_D)$

$$u^{enh}(x_1, \ldots, x_D) \approx \sum_{i=1}^{\tilde{N}} \prod_{j=1}^{D} \tilde{U}_j^i(x_j) \tag{3.44}$$

that verifies

$$u^{enh}(x_1, \ldots, x_D) = u(x_1, \ldots, x_D)\,. \tag{3.45}$$

This is done by applying the PGD to problem (3.45). This approach leads to a more compact representation of the solution with $\tilde{N} \leq N$ allowing a substantial storage saving and a significant CPU saving when performing online evaluations or post-processing tasks. However, the order of magnitude of this reduction in terms of enrichment steps, is difficult to predict because it strongly depends on the problem definition, similarly to the optimality trend.

3.4 PGD Online Phase

Once the complete convolution of solutions is computed during the previous offline phase, then the evaluation for particular values of the coordinates is performed in an ***online phase***. In this case the computational saving is the main target of the phase, in order to allow the computation on low performance devices, or even more in order to be able to compute quickly a solution for real-time applications – such as training surgery in the medical field [33] – or for optimization processes – where a huge number of solutions is required in order to find an optimal result [8, 28].

3.4.1 Online Solution Reconstruction

The online evaluation of a separated approximation of the solution is performed via the evaluation of the functions $\mathbf{U}_i^j(\mathbf{x}_j)$ related to the associated coordinate $\mathbf{x}_j$, at each enrichment step $i = 1 : N$. Thus, imposing the defined values $\bar{\mathbf{x}}_j$

for each coordinate $j = 1 : D$, we obtain the evaluated solution

$$\bar{\mathbf{U}}^N(\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_D) = \sum_{i=1}^{N} \bigotimes_{j=1}^{D} \mathbf{U}_i^j(\bar{\mathbf{x}}_j)\,. \tag{3.46}$$

In contrast to the POD approach, where a Galerkin projection is needed (see Chapter 2), with the PGD approach the computational cost is reduced because only the evaluations of functions $\mathbf{U}_i^j(\mathbf{x}_j)$ is needed. Since a total number of $N(D-1)$ products and $N-1$ sums is demanded, the online phase turns out to be very computationally cheap and suitable for portable platforms and high frequency applications.

3.5 Conclusion

Summarizing the PGD algorithm phases, we can firstly divide the procedure into the main phases, offline and online. This division allows us to build an affordable tool in terms of computational costs distribution, via the offline-online splitting.

The offline phase starts with the choice of the problem, the definition of its dependencies, and the selection of the PGD coordinates. These define the desired separation of the original problem variables. Sometimes it should be convenient to separate also the physical space as in [8]. Once the coordinates are defined, the PGD algorithm leads to the construction of a nonlinear loop – containing one different subproblem for each selected coordinate – whose solution define one step of the offline enrichment loop presented in Section 3.2.2. The selection of appropriate stopping criteria for these loops identify the reduced order basis.

Then, once the information of the convolution solution are obtained in the offline phase, the online phase simply consists in the (computationally cheap) reconstruction of the desired solution – for a new value of the coordinates defined – which, in case of the PGD, it reduces to a sum between different terms (see Section 3.4).

Implementation and Benchmark Tests 4

4.1 Introduction

In order to investigate the performances of the PGD method as proposed in Chapter 3, a twofold comparison with a reference full order solution and a reference reduced order solution is performed on different problems.

The implementation of the different problems is performed with the aid of the open-source C++ library libMesh [2]. This software library provides a framework for the numerical simulation of PDEs using arbitrary unstructured discretizations on both serial and parallel platforms, using the finite element method in 2D and in 3D, with steady and unsteady settings. Moreover, this library makes use of advanced and efficient computational libraries such as PETSc, LASPack, and SLEPc. This is crucial in view of a general MOR framework where the cost reduction is the main issue. In the same way, the PGD implementation developed in this work has led to a library extension, with the target of being a useful and well-organized tool to be interfaced with different kinds of solvers, underlining the *outer shell* role of the PGD method (see Section 3.1).

The chosen test problems include a first pilot example in order to better explain the PGD constructor action. This first problem consists in a transient two-dimensional diffusion parametric problem where the temporal variable and the diffusivity parameter are considered as extra coordinates, as described in Section 3.2.6. Afterwards, a generic heat diffusion equation is taken into account with the addition of linear transport. Finally, a two-dimensional elasticity parametric problem is considered where the elasticity parameters are defined as extra coordinates, in the context of a structural analysis.

4.2 A Parametric Transient Diffusion Equation

4.2.1 Problem Definition

We consider as first test the parametric transient heat equation, where the time coordinate t and the diffusivity parameter k, are considered extra coordinates, so that the unknown field $u(\mathbf{x}, t, k)$ is defined in $\Omega = \Omega_x \times \Sigma_t \times \Pi_k$, with $\Omega_x \subset \mathbb{R}^2$, $\Sigma_t \subset \mathbb{R}$ and $\Pi_k \subset \mathbb{R}$, where the problem is defined in a 2-dimensional space. Without loss of generality, only Dirichlet boundary conditions are considered on the whole boundary. The mathematical problem can be written as

$$\begin{cases} \frac{\partial u}{\partial t} - k\nabla^2 u = f & \text{in } \Omega_x \times \Sigma_t \times \Pi_k \\ u(\mathbf{x}, t, k) = g(\mathbf{x}, t, k) & \text{on } \partial\Omega_x \times \Sigma_t \times \Pi_k \\ u(\mathbf{x}, 0, k) = u_0(\mathbf{x}, k) = g(\mathbf{x}, 0, k) & \text{on } \Omega_x \times \Pi_k \end{cases}, \tag{4.1}$$

whose associated integral formulation reads as

$$\int_\Omega u^* \left(\frac{\partial u}{\partial t} - k\nabla^2 u - f \right) \mathrm{d}\mathbf{x}\, \mathrm{d}t\, \mathrm{d}k = 0 \tag{4.2}$$

for any suitable test function u^*.

According to Section 3.2.5 non homogeneous Dirichlet data results in the lifting process which, in turn, leads to a homogeneous Dirichlet problem, while the Neumann fluxes results in a functional to be added at the right hand side of weak formulation. However, the flux functions have to be known in a separated form. Analogously, in case of a non-constant source term it is necessary to define a separated form of such a function, as described in Section 3.3. In order to simplify the explanation of the PGD constructor homogeneous initial and boundary conditions and a constant source term f are considered, i.e., $g(\mathbf{x}, t, k) = 0$ on $\partial\Omega_x \times \Sigma_t \times \Pi_k$ and $f = 0$.

The computational domain coincides with the L-shaped in Figure 4.1. The mesh is generated with Gmsh, an open-source finite element grid generator [1]. For each of the considered solving strategies we adopt the same mesh in order to be unbiased by the mesh when comparing the performances. The FEM approximation is based on $P2$ elements combined with a fifth-order quadrature approximation for the space integral evaluation, resulting in a total of 1641 degrees of freedom for the considered mesh. The time coordinate is uniformly discretized in the range $[0; 0.1]s$ with 40 time steps and the time dependence is discretized via a theta method for $\theta = 0.5$, i.e., via the second order Crank-Nicolson finite difference scheme [39, p. 237–265]. The diffusivity

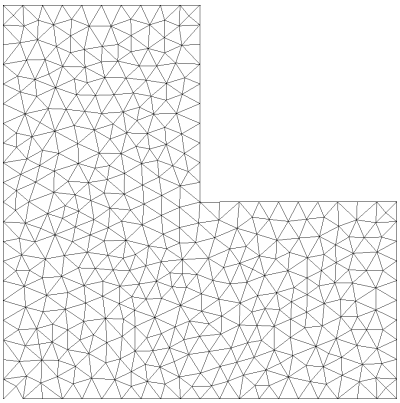


Figure 4.1: *2-D mesh generated with Gmsh*

parameter coordinate k is defined in the range 1 to 5 and uniformly discretized via 10 steps.

4.2.2 Full-Order Solution

The discretization of the coordinates for the full-order solution is defined in Section 4.2.1. In order to compare the full-order solution with the reduced-order ones, the considered parametric unsteady problem is evaluated for each different values of the diffusivity parameter k in the considered range. We obtain a space-time solution for each value of k, which can be compared with the reduced-order solutions evaluated for the same values of the parameter. As described before, the time dependence is managed with a theta method, while the physical space is discretized by the means of FEM. Accordingly to [37], a low Courant Friedrichs Lewy (CFL) number is not required for the stability of the numerical scheme. However, it is required for numerical accuracy. During this work, no attention was paid to the optimality of the solution in terms of accuracy, ignoring issues such as stabilization. The reason is that the full order solution is considered as reference solution in order to investigate the behaviour of the POD and the PGD methods, and we do not investigate the approximation quality of the single numerical scheme. So, we refer to the full-order solution as to the *truth solution*, in analogy with the Reduced Basis context [38].

4.2.3 POD Reduced-Order Solution

In the POD context, the solutions constituting the snapshot set are computed for different value of the parameters. In particular, the space coordinate is the privileged one, the time variation is discretized again through the θ-method used in Section 4.2.2, and the only parameter considered is the diffusivity coefficient. To obtain comparable solutions, not only the time dependence but also the diffusivity parameter discretization coincide with ones employed on the full order problem.

It is simple to verify that the eigenvalue magnitude decreases fast in correspondence with the first 4 POD basis functions, namely the most of the system energy is retained by this limited number of modes. This is the number of basis functions used in order to reconstruct the reduced solution during the POD online phase.

4.2.4 PGD Reduced-Order Solution

The PGD constructor leads to an approximated solution for the field $u(\mathbf{x}, t, k)$ in a separated form (see (3.1)), which reads as:

$$u(\mathbf{x}, t, k) \approx \sum_{i=1}^{N} U_x^i(\mathbf{x})\, U_t^i(t)\, U_k^i(k)\,. \tag{4.3}$$

Following the process described in Section 3.2.2, at the n-th enrichment step, the approximated solution reads

$$u^n(\mathbf{x}, t, k) = \sum_{i=1}^{n-1} U_x^i(\mathbf{x})\, U_t^i(t)\, U_k^i(k) + U_x^n(\mathbf{x})\, U_t^n(t)\, U_k^n(k)\,. \tag{4.4}$$

The alternated direction strategy (fixed point loop) provides the computation of the unknown functions starting from initial guesses for the t coordinate and for the k coordinate, say $U_t^{n,0}(t)$ and $U_k^{n,0}(k)$, and computing $U_x^n(\mathbf{x})$. Then, the scheme computes alternately the unknown associated to the other coordinates from the already known functions (as described in Section 3.2.2).

Each iteration of the fixed point loop consists in the following three steps (one for each coordinate):

- Computation of the unknown $U_x^{n,p}(\mathbf{x})$ from $U_t^{n,p-1}(t)$ and $U_k^{n,p-1}(k)$. The simplest choice of the test function to be used in (4.2) is

$$u^*(\mathbf{x}, t, k) = U_x^{n,*}(\mathbf{x})\, U_t^{n,p-1}(t)\, U_k^{n,p-1}(k)\,. \tag{4.5}$$

where $U_x^{n,*}(\mathbf{x})$ is the only test function while $U_t^{n,p-1}(t)$ and $U_k^{n,p-1}(k)$ are the trial functions known from the previous $(p-1)$-th nonlinear iteration. Substituting (4.3) and (4.5) in (4.2) yields

$$\begin{aligned}\int_\Omega U_n^{x,*}\, U_n^{t,p-1}\, U_n^{k,p-1}\left(U_n^{x,p}\,\frac{\partial U_n^{t,p-1}}{\partial t}\, U_n^{k,p-1} - k\,\nabla^2 U_n^{x,p}\, U_n^{t,p-1}\, U_n^{k,p-1}\right)\mathrm{d}\mathbf{x}\,\mathrm{d}t\,\mathrm{d}k \\ = -\int_\Omega U_n^{x,*}\, U_n^{t,p-1}\, U_n^{k,p-1}\,\mathcal{R}^{n-1}\,\mathrm{d}\mathbf{x}\,\mathrm{d}t\,\mathrm{d}k\end{aligned}, \tag{4.6}$$

where $\mathcal{R}^{n-1}$ is the residual related to $(n-1)$-th enrichment step, defined as

$$\mathcal{R}^{n-1} = \sum_{i=1}^{n-1} U_x^i\,\frac{\partial U_t^i}{\partial t}\,U_k^i - \sum_{i=1}^{n-1} k\,\nabla^2 U_x^i\,U_t^i\,U_k^i - f\,. \tag{4.7}$$

Since all t-dependent and k-dependent functions are known, it is possible to integrate (4.6) over the spaces Σ_t and Π_k. This process can be achieved integrating separately the functions depending on the single coordinates on the correspondent spaces due to the separated nature of the solution (4.3). Defining the known quantities integrated over the time domain Σ_t as

$$\begin{aligned} K_1^t &= \int_{\Sigma_t}\left(U_t^{n,p-1}\right)^2\mathrm{d}t \\ K_2^t &= \int_{\Sigma_t} U_t^{n,p-1}\,\frac{\partial U_t^{n,p-1}}{\partial t}\mathrm{d}t \\ K_3^t &= \int_{\Sigma_t} U_t^{n,p-1}\mathrm{d}t \\ K_{4,i}^t &= \int_{\Sigma_t} U_t^{n,p-1}\,\frac{\partial U_t^i}{\partial t}\mathrm{d}t \\ K_{5,i}^t &= \int_{\Sigma_t} U_t^{n,p-1}\,U_t^i\mathrm{d}t \end{aligned}\quad, \tag{4.8}$$

and similarly the known quantities integrated over the diffusivity domain

Π_k as

$$\begin{aligned} K_1^k &= \int_{\Omega_k} \left(U_k^{n,p-1}\right)^2 \mathrm{d}k \\ K_2^k &= \int_{\Omega_k} k \left(U_k^{n,p-1}\right)^2 \mathrm{d}k \\ K_3^k &= \int_{\Omega_k} U_k^{n,p-1} \mathrm{d}t \quad , \\ K_{4,i}^k &= \int_{\Omega_k} U_k^{n,p-1}\, U_k^i \mathrm{d}k \\ K_{5,i}^k &= \int_{\Omega_k} k\, U_k^{n,p-1}\, U_k^i \mathrm{d}k \end{aligned} \tag{4.9}$$

equation (4.6) becomes

$$\begin{aligned} &\int_{\Omega_x} U_x^{n,*} \Big(U_x^{n,p}\, K_2^t\, K_1^k - \nabla^2 U_x^{n,p}\, K_1^t\, K_2^k \Big) \mathrm{d}\mathbf{x} \\ = -&\int_{\Omega_x} U_x^{n,*} \Bigg(\sum_{i=1}^{n-1} \Big(U_x^i\, K_{4,i}^t\, K_{4,i}^k - \nabla^2 U_x^i\, K_{5,i}^t\, K_{5,i}^k \Big) - K_3^t\, K_3^k\, f \Bigg) \mathrm{d}\mathbf{x}\,. \end{aligned} \tag{4.10}$$

It is then possible to numerically solve the sub-problem (4.10) with the selected discretization technique, i.e, via a finite element method.

- Computation of the unknown $U_t^{n,p}(t)$ from $U_x^{n,p}(\mathbf{x})$ and $U_k^{n,p-1}(k)$.
  The test function to be used in (4.2) becomes

$$u^*(\mathbf{x}, t, k) = U_t^{n,*}(t)\, U_x^{n,p}(\mathbf{x})\, U_k^{n,p-1}(k)\,. \tag{4.11}$$

Replacing (4.3) and (4.11) in (4.2) yields

$$\begin{aligned} \int_{\Omega} U_t^{n,*}\, U_x^{n,p}\, U_k^{n,p-1} \left(U_x^{n,p}\, \frac{\partial U_t^{n,p}}{\partial t}\, U_k^{n,p-1} - k\, \nabla^2 U_x^{n,p}\, U_t^{n,p}\, U_k^{n,p-1} \right) \mathrm{d}\mathbf{x}\, \mathrm{d}t\, \mathrm{d}k \\ = -\int_{\Omega} U_t^{n,*}\, U_x^{n,p}\, U_k^{n,p-1}\, \mathcal{R}^{n-1}\, \mathrm{d}\mathbf{x}\, \mathrm{d}t\, \mathrm{d}k\,. \end{aligned} \tag{4.12}$$

As in the previous step, all the $\mathbf{x}$-dependent and k-dependent functions are known so it is possible to integrate (4.12) over the spaces Ω_x and Π_k, separately. Defining the known quantities integrated over the spatial domain

Ω_x as

$$\begin{aligned} K_1^x &= \int_{\Omega_x} \left(U_x^{n,p}\right)^2 \mathrm{d}\mathbf{x} \\ K_2^x &= \int_{\Omega_x} U_x^{n,p}\, \nabla^2 U_x^{n,p} \mathrm{d}\mathbf{x} \\ K_3^x &= \int_{\Omega_x} U_x^{n,p} \mathrm{d}\mathbf{x} \\ K_{4,i}^x &= \int_{\Omega_x} U_x^{n,p}\, \nabla^2 U_x^{i} \mathrm{d}\mathbf{x} \\ K_{5,i}^x &= \int_{\Omega_x} U_x^{n,p}\, U_x^{i} \mathrm{d}\mathbf{x} \end{aligned} \tag{4.13}$$

and exploiting quantities in (4.9), equation (4.12) becomes

$$\begin{aligned} &\int_{\Sigma_t} U_t^{n,*} \left(K_1^x \, \frac{\partial U_t^{n,p}}{\partial t} \, K_1^k - K_2^x \, U_t^{n,p} \, K_2^k \right) \mathrm{d}t \\ = &- \int_{\Sigma_t} U_t^{n,*} \left(\sum_{i=1}^{n-1} \left(K_{5,i}^x \, \frac{\partial U_t^{i}}{\partial t} \, K_{4,i}^k - K_{4,i}^x \, U_t^{i} \, K_{5,i}^k \right) - K_3^x \, K_3^k \, f \right) \mathrm{d}t \, . \end{aligned} \tag{4.14}$$

This is the integral form of an IVP for the unknown $U_t^{n,p}$ that can be numerically solved via different discretization techniques. It could be useful to consider the corresponding strong form

$$\begin{aligned} &K_1^x \, \frac{\partial U_t^{n,p}}{\partial t} \, K_1^k - K_2^x \, U_t^{n,p} \, K_2^k \\ = &- \left(\sum_{i=1}^{n-1} \left(K_{5,i}^x \, \frac{\partial U_t^{i}}{\partial t} \, K_{4,i}^k - K_{4,i}^x \, U_t^{i} \, K_{5,i}^k \right) - K_3^x \, K_3^k \, f \right) \end{aligned} \tag{4.15}$$

and solve it via any suitable method for ordinary differential equations, such as a theta-method [37].

- Computation of the unknown $U_k^{n,p}(k)$ from $U_t^{n,p}(t)$ and $U_x^{n,p}(\mathbf{x})$.
  The test function to be used in (4.2) becomes

$$u^*(\mathbf{x}, t, k) = U_k^{n,*}(k)\, U_x^{n,p}(\mathbf{x})\, U_t^{n,p}(t) \, . \tag{4.16}$$

Replacing (4.3) and (4.16) in (4.2) yields

$$\begin{aligned}\int_{\Omega} U_k^{n,*}\, U_x^{n,p}\, U_t^{n,p}\Big(U_x^{n,p}\,\frac{\partial U_t^{n,p}}{\partial t}\,U_k^{n,p} - k\,\nabla^2 U_x^{n,p}\, U_t^{n,p}\, U_k^{n,p}\Big)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t\,\mathrm{d}k \\ = -\int_{\Omega} U_k^{n,*}\, U_x^{n,p}\, U_t^{n,p}\,\mathcal{R}^{n-1}\,\mathrm{d}\mathbf{x}\,\mathrm{d}t\,\mathrm{d}k\,.\end{aligned} \tag{4.17}$$

Once again all the $\mathbf{x}$-dependent and t-dependent functions are known and we can integrate (4.17) over the domains Ω_x and Σ_t, separately. Defining the known quantities integrated over the time space Σ_t as in (4.8) using the just computed functions, i.e., replacing with the index p the old index $p-1$, and exploiting quantities in (4.13), equation (4.17) becomes

$$\begin{aligned}\int_{\Omega_k} U_k^{n,*}\Big(K_1^x\,K_2^t\,U_k^{n,p} - K_2^x\,K_1^t\,k\,U_k^{n,p}\Big)\mathrm{d}k \\ = -\int_{\Omega_k} U_k^{n,*}\Bigg(\sum_{i=1}^{n-1}\Big(K_{5,i}^x\,K_{4,i}^t - k\,K_{4,i}^x\,K_{5,i}^t\Big)U_k^i - K_3^x\,K_3^t\,f\Bigg)\mathrm{d}k\,.\end{aligned} \tag{4.18}$$

This equation does not involve any differential operator since the model of the problem (4.2) does not contain any derivative with respect to the diffusivity parameter k. So that the corresponding strong form

$$\Big(K_1^x\,K_2^t - K_2^x\,K_1^t k\Big)U_k^{n,p} = -\Bigg(\sum_{i=1}^{n-1}\Big(K_{5,i}^x K_{4,i}^t - K_{4,i}^x\,K_{5,i}^t\Big)U_k^i - k\,K_3^x\,K_3^t\,f\Bigg) \tag{4.19}$$

leads to an algebraic equation that can be simply solved.

Iterating the above procedure until reaching a fixed point for the nonlinear loop by the satisfaction of a specific stopping criterion (see Section 3.2.3) leads to the solution of the n-th enrichment functional product $U_x^n(\mathbf{x})\,U_t^n(t)\,U_k^n(k)$ by the assignment of the last non-linear iteration solution, i.e., $U_x^n(\mathbf{x}) \leftarrow U_x^{n,p}(\mathbf{x})$, $U_t^n(t) \leftarrow U_t^{n,p}(t)$ and $U_k^n(k) \leftarrow U_k^{n,p}(k)$.

4.2.5 A Cross Comparison

The parametric transient diffusion problem (4.1) is solved considering a constant source term $f = 1$, while the lifting function (accordingly to Section 3.2.5) is defined by:

$$g(\mathbf{x}, t, k) \;=\; 0 \tag{4.20}$$

corresponding to homogeneous Dirichlet boundary conditions and null initial data.

The full-order solutions require to solve different problem, one for each value of the discretized diffusivity coefficient. In order to compare the time needed to compute the full-order solutions with the time needed by the reduced-order methods, we reconstruct the online solutions for each value of the (diffusivity) parameter used during the full-order computation. In this way we compare the time needed by the three different approaches during the computation of the same number of solutions.

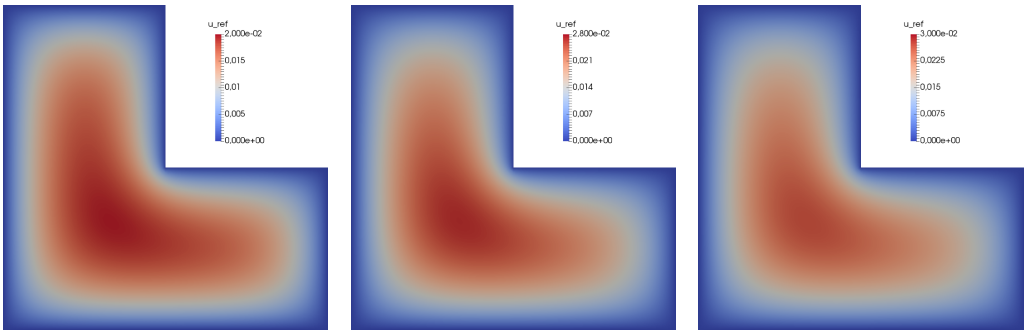
In Table 4.1 the results obtained via serial simulations are presented. The full-order model performance is evaluated via the total time required by all the simulations, as well as the average time required by the 11 simulations needed to discretize the diffusivity range into 10 steps. The POD performance indices considered are the time required by the POD offline stage, the time required by the POD online reconstruction, the ratio between the total time needed by the POD and the full-order FEM to compute the same number of simulations. The same performance indices are considered for the PGD approach with additional details on the number of enrichment steps and average nonlinear steps needed during the PGD offline stage.

During the PGD offline computation of the solution the tolerances used for both the non-linear loop and the enrichment loop are $1e-4$ (see Section 3.2.3 for more details on the stopping criteria).

The computational time needed for the PGD offline construction and the online evaluation has an order of the magnitude which is 20% of the total time needed to perform the classical incremental simulations, for the same number of simulations, while the average computational time needed to reconstruct online the solution is 15% of the average time demanded by the full-order simulations.

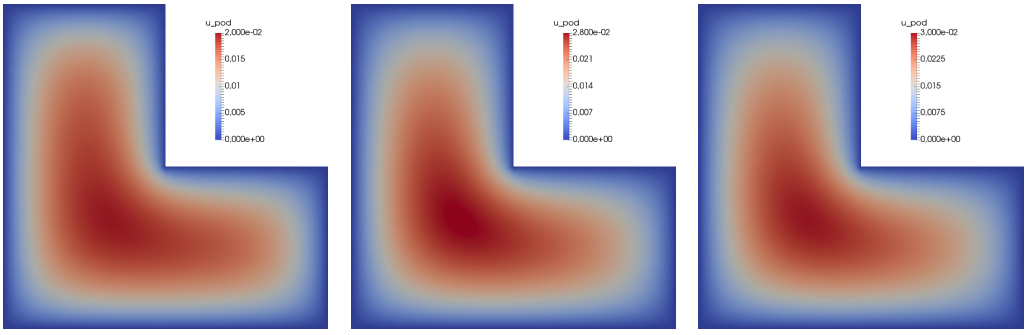
The computational saving is evident with a low number of degrees of freedom due to the non-incremental nature of the PGD method in tackling the time dependence (see Section 3.2.4) and to the generic decoupled nature of the diffusivity sub-problems. In fact, even if the nonlinear nature of the PGD leads to a priori coupled problems, it is evident that the physical nature of the transient diffusion equation – considering the diffusivity as parameter – leads to an algebraic dependence with respect to the diffusivity coefficient, as detailed in Section 4.2.4.

The results of the numerical simulations are now qualitatively compared at 3 different times and for $k = 5$. The solutions obtained with the full-order FEM, the reduced-order POD solutions and the corresponding pointwise errors with respect to the reference solution, and the reduced-order PGD solutions with the associated error with respect to the reference solution, are plotted at times



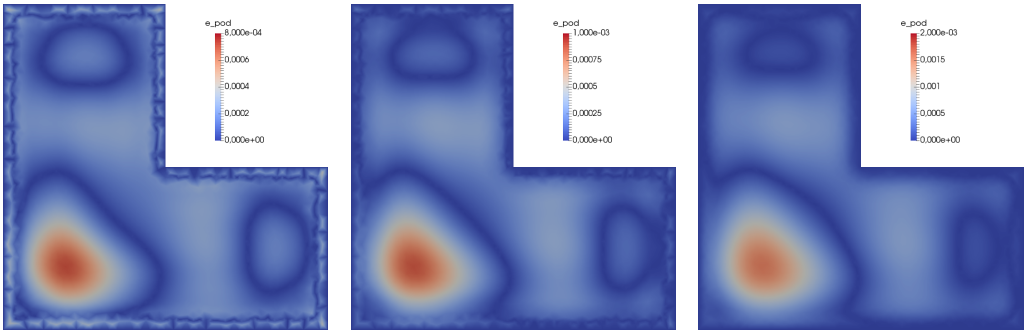
(a) $t = 0.025$s **(b)** $t = 0.0625$s **(c)** $t = 0.1$s

Figure 4.2: *Test 1, full-order solutions for* k=5



(a) $t = 0.025$s **(b)** $t = 0.0625$s **(c)** $t = 0.1$s

Figure 4.3: *Test 1, POD solutions for* k=5



(a) $t = 0.025$s **(b)** $t = 0.0625$s **(c)** $t = 0.1$s

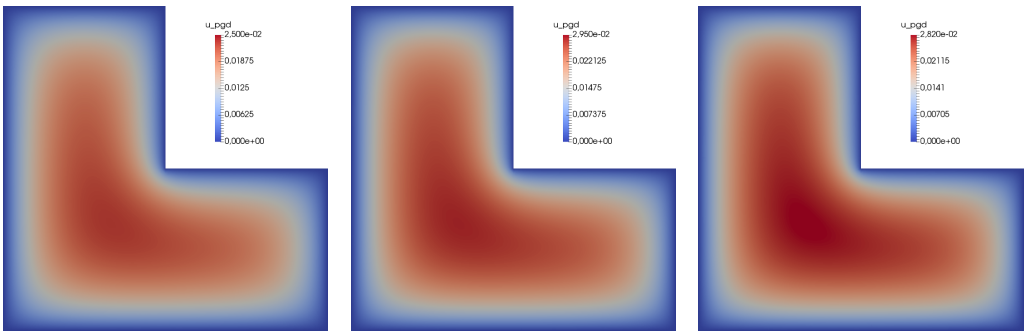
Figure 4.4: *Test 1, POD pointwise errors for* k=5

Table 4.1: *Test 1 performance comparison among full-order solution, POD reduced-order solution, and PGD reduced-order solution. Results on Linux 4.2.0-27-generic x86_64 on Intel® Core™i5-2500K CPU @ 3.30 GHz.*

Full-order single simulation average time		1.3727	s
Full-order total time		15.0993	s
POD reduced-order simulation	*offline time*	24.9106	s
	online total time	1.1796	s
	online average time	0.1072	s
POD reduced-order total time		26.0902	s
PGD reduced-order simulation	*offline time*	0.6658	s
	online total time	2.6340	s
	online average time	0.2395	s
PGD reduced-order total time		3.2998	s
PGD number of enrichment steps		3	
PGD average number of nonlinear steps		1	
PGD total time over full-order total time ratio		0.2185	
PGD average online time over full-order average time ratio		0.1745	

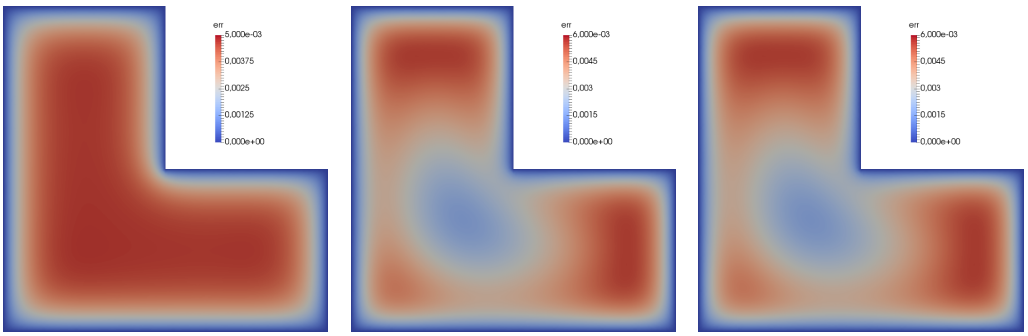
$t = 0.025$s, $t = 0.0625$s, and $t = 0.1$s. The pointwise error defined is simply the absolute value of the difference between the full order solution and the reduced-order solutions. We consider the final time and two intermediate times between the final and the initial time, because the solutions at the initial time are all coincident with a null solution in the whole space domain, being assigned homogeneous initial and boundary conditions. In particular, in Figure 4.2 the full order solutions are presented, in Figure 4.3 and in Figure 4.4 the POD reduced-order solutions and their errors are represented, while in Figure 4.5 and in Figure 4.6 the PGD reduced-order solutions and their errors are plotted.

From the obtained results, we can assert that the PGD solution leads to an approximation error – with respect to the full-order solution – which is greater with respect to the POD counterpart, even if is more homogeneously distributed in the spatial domain. However, the computational saving in computing the PGD reduced order solution is clearly evident, and sometimes it can be more convenient a PGD approach at the expense of a more accurate POD solution.



(a) $t = 0.025$s **(b)** $t = 0.0625$s **(c)** $t = 0.1$s

Figure 4.5: *Test 1, PGD solutions for k=5*



(a) $t = 0.025$s **(b)** $t = 0.0625$s **(c)** $t = 0.1$s

Figure 4.6: *Test 1, PGD pointwise errors for k=5*

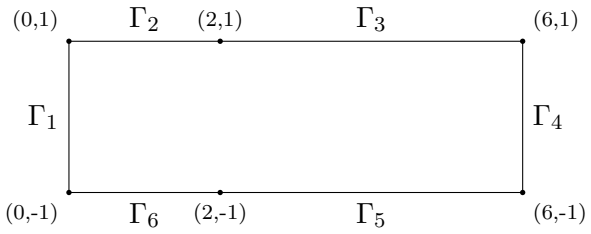


Figure 4.7: *Duct domain for the considered Graetz problem*

4.3 Graetz Problem

4.3.1 Problem Definition

The second benchmark test considered is a classical problem in the literature [21] and consists in a steady advective-convective problem modelling heat conduction in a duct where the walls are at different temperatures. A first segment considered "cold" with a zero temperature is followed by a second "hot" segment where a higher temperature is imposed, i.e., we impose $u = c$ on Γ_C portion of the boundary with null temperature and $u = h$ on Γ_H the second portion of the boundary. The flow has an imposed cold temperature at the inlet, and is subject to a known convection field, directed as the duct axis and defined by a parabolic velocity profile with the maximum value on the axis and vanishing at the walls. This problem models many aspects of steady convection-diffusion phenomena such as heat transfer into a channel, forced convection with an imposed velocity profile, heat conduction through walls, insulation [3]. From a physical point of view, the Péclet number is the dimensionless number relevant in the study of transport phenomena and it coincides with the ratio between the advective and the diffusive transport rate, being defined as

$$\mathrm{Pe} = \frac{L\|\mathbf{b}\|}{2\,k} \tag{4.21}$$

where L is a characteristic length of the problem, $\mathbf{b}$ is the local advective field, and k is the diffusivity coefficient.

In this problem the physics is complex and it changes with the Péclet number, starting with the presence of thermal boundary layers, which are firstly separated and then interact each other, and arriving to their absence.

We considered a problem where the transport velocity profile is known while the diffusivity coefficient is a problem parameter. It follows that the Péclet number changes due to the diffusivity variation. The considered spatial domain and the subdivision of the boundary are defined in Figure 4.7.

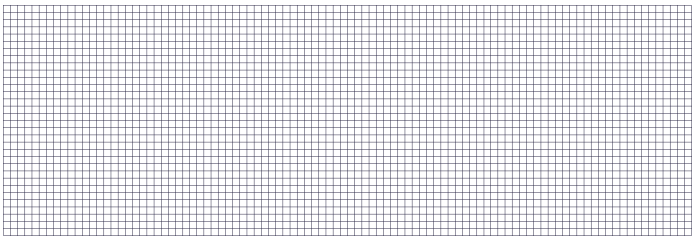


Figure 4.8: *2-D mesh generated with Gmsh for the considered Graetz problem*

The governing equation the a transport-diffusion equation

$$-k\nabla^2 u(\mathbf{x},k) + \mathbf{b}\cdot\nabla u(\mathbf{x},k) = 0 \quad \text{in } \Omega_{\mathbf{x}}\times\Omega_k \tag{4.22}$$

where the diffusivity coefficient is a problem parameter and the domain is $\Omega = \Omega_{\mathbf{x}} \times \Omega_k$, with $\Omega_{\mathbf{x}}$ spatial domain and Ω_k the domain associated with the parameter k. The considered boundary conditions are the imposed "cold" temperature on the first segment of the duct, i.e., on $\Gamma_C = \Gamma_1 \cup \Gamma_2 \cup \Gamma_6$, the imposed "hot" temperature on $\Gamma_H = \Gamma_3 \cup \Gamma_5$, and null flux on Γ_4. The problem can be stated in a generic integral formulation by means of integration by parts, thus obtaining

$$\int_\Omega k\nabla u^* \cdot \nabla u + \int_\Omega u^* \mathbf{b}\cdot\nabla u = 0 \tag{4.23}$$

for any suitable test function u^*, where

$$\begin{aligned} &u \in U = \left\{ u \in H^1(\Omega_{\mathbf{x}}) \quad \text{such that } u(\mathbf{x}\in\Gamma_C) = c\,,\ u(\mathbf{x}\in\Gamma_H) = h \right\} \\ &u^* \in H^1_{0,\Gamma_C\cup\Gamma_H}(\Omega_{\mathbf{x}}) \end{aligned} \tag{4.24}$$

The 2D spatial domain is discretized via Gmsh, obtaining the mesh in Figure 4.8, the diffusivity domain is the range $[1;5]$ uniformly discretized in N_K steps, and the (constant) transport velocity is defined by the parabolic profile with a maximum value equal to 10. Note that none of the problems associated with the different values of k is dominated by the transport term, i.e., we have low Péclet number. This avoids any investigation on stability to more deeply analyze the robustness of the reduced order models.

Table 4.2: *Greater POD eigenvalues in decreasing order for the considered Graetz problem, with $N_K = 100$ snapshots computed.*

$\lambda_1 = 9.88e-01$
$\lambda_2 = 5.74e-03$
$\lambda_3 = 6.84e-05$
$\lambda_4 = 1.26e-06$
$\lambda_5 = 1.49e-08$
$\lambda_6 = 2.22e-10$
$\lambda_7 = 2.03e-12$
$\vdots$

4.3.2 Full-Order Solution

In order to compute the full-order solution, the parametric problem is evaluated for each value in the range $[1; 5]$ after a subdivision in N_K steps. We analyze the solver performances for different values of N_K – as detailed in Section 4.3.5 – obtaining a set of transport-diffusion problems where the diffusion coefficient is constant – and consequently the Péclet number – for each choice of k. Even if the considered problems are characterized by different Péclet numbers, none of the considered value requires a stabilization. For this reason, the implementation of the full-order solution is performed by the means of a standard FEM, computed on the mesh in Figure 4.8. We choose $P2$ finite elements combined with a fifth order quadrature rule, dealing to a total number of 12545 degrees of freedom.

4.3.3 POD Reduced-Order Solution

The POD reduced-order solution is computed by assuming the diffusivity coefficient the only parameter of the problem. In order to compare the reduced solutions obtained with the PGD method, the POD is performed using N_K *training samples* to compute the basis functions during the offline phase. Looking at the eigenvalue magnitude in Table 4.2, we can recognize that it decreases very fast, i.e., most of the energy of the system is retained by the first modes. Indeed, once we define a tolerance of $10e-12$, the number of the POD basis functions to be considered during the online phase results to be 4 (see Section 2.2 for the tolerance definition).

4.3.4 PGD Reduced-Order Solution

The PGD reduced-order solution is obtained for a discretization of the diffusivity parameter range defined by N_K steps. Accordingly to (3.1), the PGD constructor leads to an approximated solution for the field $u(\mathbf{x}, k)$ in a separated form, which reads:

$$u(\mathbf{x}, k) \approx \sum_{i=1}^{N} U_x^i(\mathbf{x})\, U_k^i(k)\,. \tag{4.25}$$

Following the process described in Section 3.2.2, at the n-th enrichment step the approximated solution is

$$u^n(\mathbf{x}, k) = \sum_{i=1}^{n-1} U_x^i(\mathbf{x})\, U_k^i(k) + U_x^n(\mathbf{x})\, U_k^n(k)\,. \tag{4.26}$$

The alternated direction strategy (fixed point loop) provides the computation of the unknown function starting from initial guesses for the k coordinate, say $U_k^{n,0}(k)$. Then, the scheme computes, alternately, the unknown from the already known functions.

Each iteration of the fixed point loop consists in the following steps, one for each coordinate:

- Computation of the unknown $U_x^{n,p}(\mathbf{x})$ from $U_k^{n,p-1}(k)$.
  The simplest choice of the test function to be used in (4.23) is

$$u^*(\mathbf{x}, k) = U_x^{n,*}(\mathbf{x})\, U_k^{n,p-1}(k)\,. \tag{4.27}$$

  Substituting (4.25) and (4.27) in (4.23), yields

$$\begin{aligned}&\int_\Omega \left(k\, \nabla U_x^{n,*} \cdot \nabla U_x^{n,p}\, U_k^{n,p-1}\, U_k^{n,p-1} + U_x^{n,*}\mathbf{b} \cdot \nabla U_x^{n,p}\, U_k^{n,p-1}\, U_k^{n,p-1} \right) \mathrm{d}\mathbf{x}\, \mathrm{d}k \\ &= -\sum_{i=1}^{n-1} \int_\Omega \left(k\, \nabla U_x^{n,*} \cdot \nabla U_x^{i}\, U_k^{n,p-1}\, U_k^{i} + U_x^{n,*}\mathbf{b} \cdot \nabla U_x^{i}\, U_k^{n,p-1}\, U_k^{i} \right) \mathrm{d}\mathbf{x}\, \mathrm{d}k x\end{aligned}\,, \tag{4.28}$$

  where the right-end side is the residual related to the $(n-1)$-th enrichment step. Since all k-dependent functions are known, it is possible to integrate (4.28) over the space Π_k. This process can be achieved integrating separately the functions depending on single coordinate on their own space due to the separated nature of the solution (4.25). Defining the known quantities

integrated over the diffusivity domain Π_k by

$$\begin{aligned} K_1^k &= \int_{\Omega_k} \left(U_k^{n,p-1}\right)^2 \mathrm{d}k \\ K_2^k &= \int_{\Omega_k} k \left(U_k^{n,p-1}\right)^2 \mathrm{d}k \\ K_{3,i}^k &= \int_{\Omega_k} U_k^{n,p-1}\, U_k^i \mathrm{d}k \\ K_{4,i}^k &= \int_{\Omega_k} k\, U_k^{n,p-1}\, U_k^i \mathrm{d}k \end{aligned} \quad , \tag{4.29}$$

equation (4.28) becomes

$$\begin{aligned} &\int_{\Omega_x} \left(\nabla U_x^{n,*} \cdot \nabla U_x^{n,p}\, K_2^k \,+\, U_x^{n,*} \mathbf{b} \cdot \nabla U_x^{n,p}\, K_1^k \right) \mathrm{d}\mathbf{x} \\ = &- \sum_{i=1}^{n-1} \int_{\Omega} \left(\nabla U_x^{n,*} \cdot \nabla U_x^i\, K_{4,i}^k \,+\, U_x^{n,*} \mathbf{b} \cdot \nabla U_x^i\, K_{3,i}^k \right) \mathrm{d}\mathbf{x} \end{aligned} \quad . \tag{4.30}$$

It is then possible to numerically solve the sub-problem (4.30) with the preferred discretization technique.

- Computation of the unknown $U_n^{k,p}(k)$ from $U_n^{x,p}(\mathbf{x})$.
  The test function to be used in (4.23) becomes

$$u^*(\mathbf{x}, k) \;=\; U_k^{n,*}(k)\, U_x^{n,p}(\mathbf{x})\,. \tag{4.31}$$

Using (4.25) and (4.31) in (4.23), yields

$$\begin{aligned} &\int_{\Omega} \left(k\, \nabla U_x^{n,p} \cdot \nabla U_x^{n,p}\, U_k^{n,*}\, U_k^{n,p} \,+\, U_x^{n,p} \mathbf{b} \cdot \nabla U_x^{n,p}\, U_k^{n,*}\, U_k^{n,p} \right) \mathrm{d}\mathbf{x}\, \mathrm{d}k \\ = &- \sum_{i=1}^{n-1} \int_{\Omega} \left(k\, \nabla U_x^{n,p} \cdot \nabla U_x^i\, \hat{U}_k^{n,*}\, U_k^i \,+\, U_x^{n,p} \mathbf{b} \cdot \nabla U_x^i\, U_k^{n,*}\, U_k^i \right) \mathrm{d}\mathbf{x}\, \mathrm{d}k \end{aligned} \tag{4.32}$$

As in the previous step all the $\mathbf{x}$-dependent functions are known. So it is possible to integrate (4.32) over the space Ω_x. Defining the known quantities

integrated over the spatial domain Ω_x by

$$\begin{aligned} K_1^x &= \int_{\Omega_x} \nabla U_x^{n,p} \cdot \nabla U_x^{n,p} \mathrm{d}\mathbf{x} \\ K_2^x &= \int_{\Omega_x} U_x^{n,p} \mathbf{b} \cdot \nabla U_x^{n,p} \mathrm{d}\mathbf{x} \\ K_{3,i}^x &= \int_{\Omega_x} \nabla U_x^{n,p} \cdot \nabla U_x^{i} \mathrm{d}\mathbf{x} \\ K_{4,i}^x &= \int_{\Omega_x} U_x^{n,p} \mathbf{b} \cdot \nabla U_x^{i} \mathrm{d}\mathbf{x} \end{aligned}, \tag{4.33}$$

equation (4.32) becomes

$$\begin{aligned} &\int_{\Omega} \left(k\, U_k^{n,*}\, U_k^{n,p}\, K_1^x \,+\, U_k^{n,*}\, U_k^{n,p}\, K_2^x \right) \mathrm{d}k \\ &= -\sum_{i=1}^{n-1} \int_{\Omega} \left(k\, U_k^{n,*}\, U_k^{i}\, K_{3,i}^x \,+\, U_k^{n,*}\, U_k^{i}\, K_{4,i}^x \right) \mathrm{d}k \end{aligned}. \tag{4.34}$$

This equation does not involve any differential operator since the model of problem (4.23) does not contain any derivative with respect to the diffusivity parameter k. So, the corresponding strong form

$$\left(K_2^x \,+\, k\, K_1^x \right) U_k^{n,p} = -\sum_{i=1}^{n-1} \left(K_{4,i}^x \,+\, k\, K_{3,i}^x \right) U_k^{i} \tag{4.35}$$

leads to an algebraic equation for the unknown $U_k^{n,p}$ that is simply solvable.

Iterating the above procedure until reaching a fixed point for the nonlinear loop by the satisfaction of a specific stopping criterion (see Section 3.2.3) leads to identify the n-th enrichment function product $U_x^n(\mathbf{x})\, U_k^n(k)$ by the assignment of the last nonlinear iteration solution, i.e. $U_x^n(\mathbf{x}) \leftarrow U_x^{n,p}(\mathbf{x})$ and $U_k^n(k) \leftarrow U_k^{n,p}(k)$.

4.3.5 Performance Comparisons

The considered Graetz parametric problem is a transport-diffusion problem with non-homogeneous boundary conditions. Accordingly to Section 3.2.5, the lifting function has to be a function which satisfy the problem boundary conditions and smooth enough in the physical space domain $\Omega_{\mathbf{x}}$. For this purpose, we choose to define the lifting function $g(\mathbf{x}, k)$ as the solution of

the Laplace problem associated with the domain $\Omega_{\mathbf{x}}$ with the same boundary conditions as the Graetz problem. Since the boundary conditions do not depend on any parameter, the lifting function results to be the solution of the problem

$$\begin{cases} \nabla^2 u_g = 0 & \text{in } \Omega_{\mathbf{x}} \\ u(\mathbf{x}) = 0 & \text{on } \Gamma_1 \cup \Gamma_2 \cup \Gamma_6 \\ u(\mathbf{x}) = 1 & \text{on } \Gamma_3 \cup \Gamma_5 \end{cases} \tag{4.36}$$

obtaining

$$g(\mathbf{x}, k) = u_g(\mathbf{x}) \,. \tag{4.37}$$

The full-order solutions require to solve different problems, one for each value of the discretized diffusivity coefficient. In order to compare the time needed to compute the full-order solutions with the time needed by the reduced-order methods, we reconstruct the online solutions for each (diffusivity) parameter value used during the full-order computation. In this way we compare the time needed by the three different approaches during the computation of the same number of solutions.

In Table 4.3 the results obtained for $N_K = 100$ via a sequence of serial simulations are presented. The full-order performance is evaluated with the total time required by the simulations, together with the average time required by the 101 simulations needed to discretize the diffusivity range into N_K steps. The POD performance indices are the time required by the POD offline construction, the time required by the POD online reconstruction, the ratio between the total time needed by the POD and the full-order FEM to compute the same number of simulations. The same performance indices are considered for the PGD with the addition of details on the number of enrichment steps and average nonlinear steps needed by the PGD offline phase.

During the PGD offline computation, the tolerances used for both the nonlinear loop and the enrichment loop are set to $1e-4$ (see Section 3.2.3 for more details on the stopping criteria).

The computational time needed to perform the PGD offline construction and the online evaluation turns out to be approximately 40% of the total time needed by the classical FEM. It is reasonable to tribute this behaviour to the nonlinear nature of the PGD problem. The PGD online average computational time is about the 1% of the average time needed by the full-order simulations. Comparing the PGD performances with the reference reduced-order POD simulation, whose total time is about the same as the one demanded by FEM

Table 4.3: *Performances comparison among full-order solution, POD reduced-order solution, and PGD reduced-order solution for the considered Graetz problem with $N_K = 100$. Results on Linux 4.2.0-27-generic x86_64 on Intel® Core™i5-2500K CPU @ 3.30 GHz.*

Full-order simulation	*average time on* 101	0.47211	s
Full-order total time		47.68280	s
POD reduced-order	*offline time*	45.61237	s
	online average time	0.00259	s
	online total time	0.26169	s
POD reduced-order total time		45.87406	s
POD total time over full-order total time ratio		96.21	%
POD average online time over full-order average time ratio		0.55	%
PGD reduced-order	*offline time*	17.40050	s
	online total time	0.25910	s
	online average time	0.00257	s
PGD reduced-order total time		17.65960	s
PGD number of enrichment steps		5	
PGD average number of nonlinear steps		2	
PGD total time over full-order total time ratio		37.04	%
PGD average online time over full-order average time ratio		0.54	%

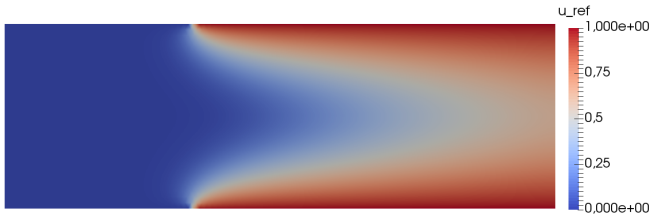
performance, we observe that the computational saving of the PGD method is significant. The online time to reconstruct both the POD and the PGD solutions exhibits, as expected in the view of the offline-online separation, a drastic reduction of the "on-the-fly" computational cost.

The qualitative results of the numerical simulations are proposed for two different values of the diffusivity coefficient k, in particular, we pick the extremes of the considered range, i.e., $k = 1$ and $k = 5$. The corresponding Péclet number is computed considering the total length of the duct $L = 6$ and the maximum of the transport velocity $\mathbf{b} = 10$, located along the centerline. For the considered values of the diffusivity parameter, via (4.21), we obtain $Pe_{K=1} = 60$ and $Pe_{k=5} = 12$, respectively.

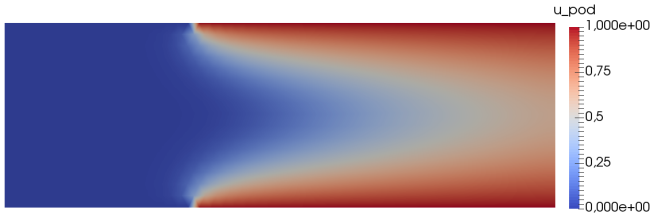
Comparing the performance indices when the number N_K increases, we can draw some conclusions on the performance of the PGD method for the

examined problem. We performed the computation with five different number of the parameter discretization steps: 100, 169, 225, 289 and 400. We present the solutions for the only two extreme values, i.e., 100 and 400. In Figure 4.9 and in Figure 4.10 we show the solutions of the computations for $N_K = 100$ and for the chosen values of the parameter and their corresponding pointwise errors, simply computed as the absolute value of the difference between the full order solution and the reduced-order solutions. Then, in Table 4.4 the results of the same computation for $N_K = 400$ are presented. The computational time needed to perform the PGD offline construction and the online evaluation is approximately the 10% of the total FEM time, and the PGD online average computational time is less then the 1% of the average time needed by the full-order simulations. The computed solutions and the reduced-order model errors are presented in Figure 4.11 and Figure 4.12. We notice that the PGD reduced order solution is affected by an error which is more then twice the error produced by the POD approximation. In Figure 4.13 we plot in the same graph the total time needed by the two reduced-order methods divided by the time required by full-order method. Notice that, for the POD solutions, the total time needed is of the same order of magnitude as for the reference one, while it decreases considerably for the PGD solutions, at the expense of the solution accuracy.

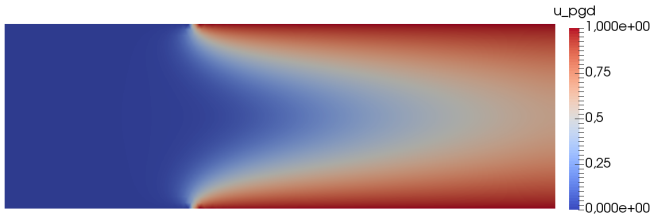
From the obtained results, it is evident that the PGD algorithm is more convenient compared to the proposed POD reduced-order counterpart. The fact that the computational time of the PGD results lower than the POD counterpart for a high number of discretization steps of the parameter is due to the nonlinear nature of the PGD method which adds to the original problem a computational cost negligible only for a high dimension of the discretized parameter space. Indeed, when we choose $N_K = 400$ the POD method results more expensive from a computational point of view, because of the nature of POD which requires to evaluate offline a number of problem equal to the (high) number of snapshots. In case we have only one parameter this number coincides with the number of the considered values of the parameter. The computational cost reduction given by the PGD – with respect to the POD – is clear even in case of one parameter, and it is reasonable to expect an increase of the computational saving when dealing with high-dimensional models and with more complex problems. In fact, considering, for example, $N_K = 100$ where the POD requires the solution of 100 problems, we have that the PGD requires the solution of a number of problems of one order of magnitude less, namely 30 problems. If the complexity of these problems increases we have that the PGD nonlinear additional cost is very little compared to the cost of each problem resolution. For example, this is the case of time dependence (see Section 4.2.5) which considerably increases the computational cost of each



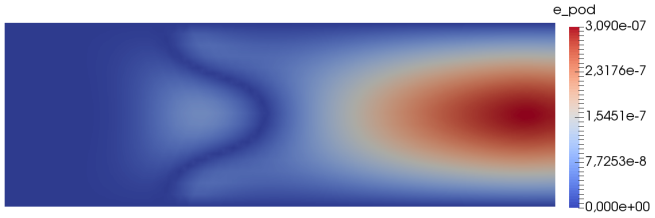
(a) *Reference full-order solution*



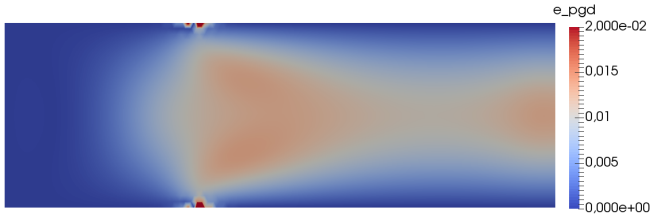
(b) *Reference POD reduced-order solution*



(c) *PGD reduced-order solution*

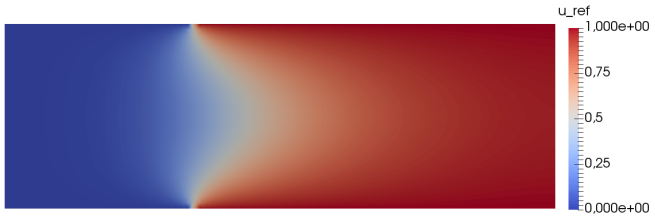


(d) *POD reduced-order solution pointwise error*

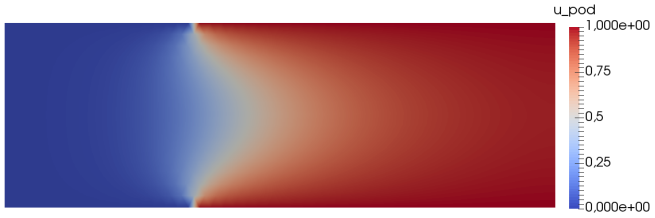


(e) *PGD reduced-order solution pointwise error*

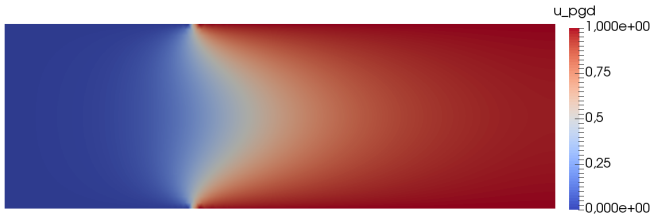
Figure 4.9: *Graetz problem numerical solutions for k=1 with $N_K = 100$.*



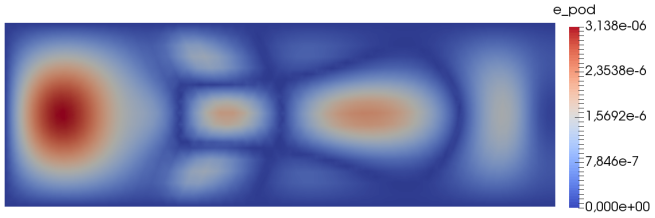
(a) *Reference full-order solution*



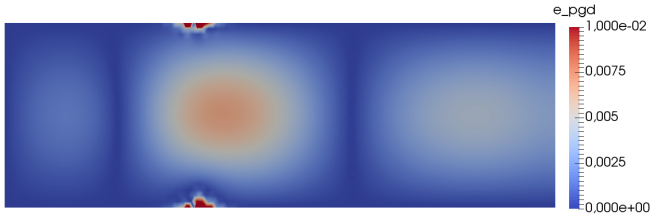
(b) *Reference POD reduced-order solution*



(c) *PGD reduced-order solution*

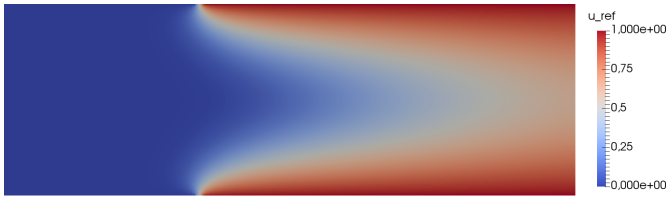


(d) *POD reduced-order solution pointwise error*

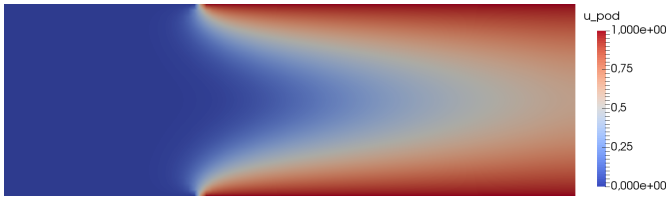


(e) *PGD reduced-order solution pointwise error*

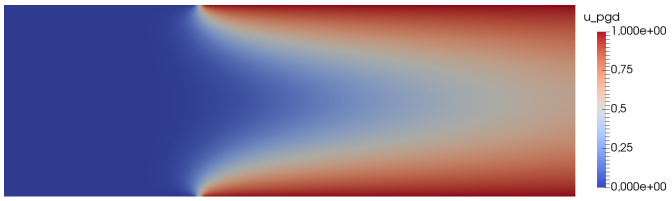
Figure 4.10: *Graetz problem numerical solutions for* k=5 *with* $N_K = 100$.



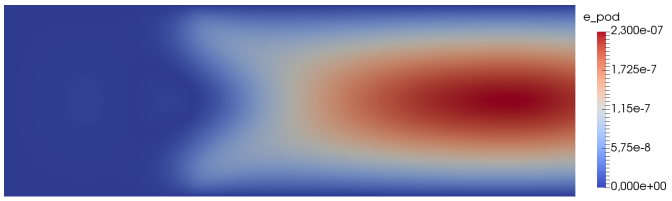
(a) *Reference full-order solution*



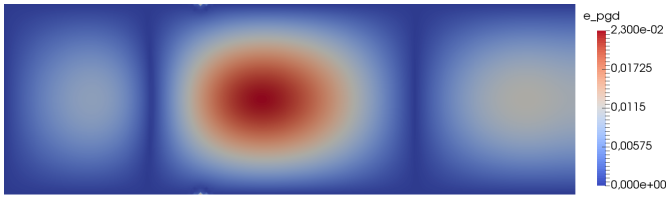
(b) *Reference POD reduced-order solution*



(c) *PGD reduced-order solution*

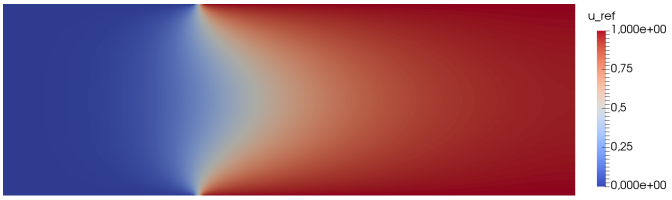


(d) *POD reduced-order solution pointwise error*

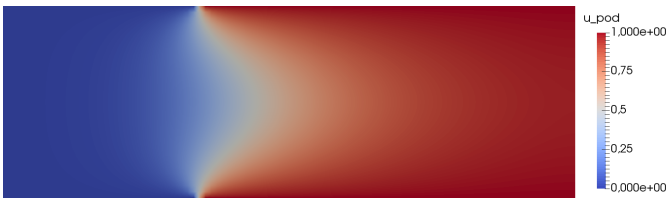


(e) *PGD reduced-order solution pointwise error*

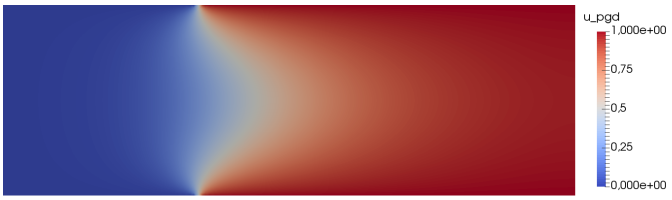
Figure 4.11: *Graetz problem numerical solutions for k=1 with $N_K = 400$.*



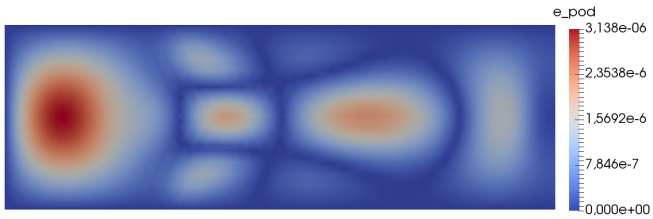
(a) *Reference full-order solution*



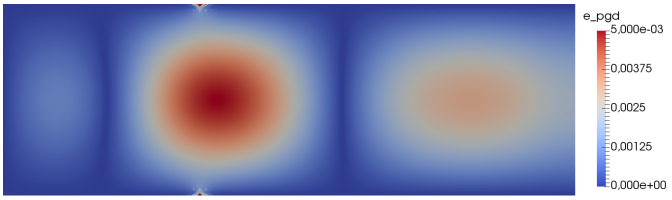
(b) *Reference POD reduced-order solution*



(c) *PGD reduced-order solution*



(d) *POD reduced-order solution pointwise error*



(e) *PGD reduced-order solution pointwise error*

Figure 4.12: *Graetz problem numerical solutions for k=5 with $N_K = 400$.*

Table 4.4: *Performances comparison among full-order solution, POD reduced-order solution, and PGD reduced-order solution for the considered Graetz problem with $N_K = 400$. Results on Linux 4.2.0-27-generic x86_64 on Intel® Core™i5-2500K CPU @ 3.30 GHz.*

Full-order simulation	*average time on* 401	0.4803	s
Full-order total time		192.5905	s
POD reduced-order	*offline time*	189.8819	s
	online average time	0.0026	s
	online total time	1.0518	s
POD reduced-order total time		190, 9337	s
POD total time over full-order total time ratio		99.14	%
POD average online time over full-order average time ratio		0.54	%
PGD reduced-order	*offline time*	16.3494	s
	online total time	1.0364	s
	online average time	0.0026	s
PGD reduced-order total time		17.3858	s
PGD number of enrichment steps		5	
PGD average number of nonlinear steps		2	
PGD total time over full-order total time ratio		9.03	%
PGD average online time over full-order average time ratio		0.54	%

problem making the PGD more convenient compared with the reference POD reduced-order method.

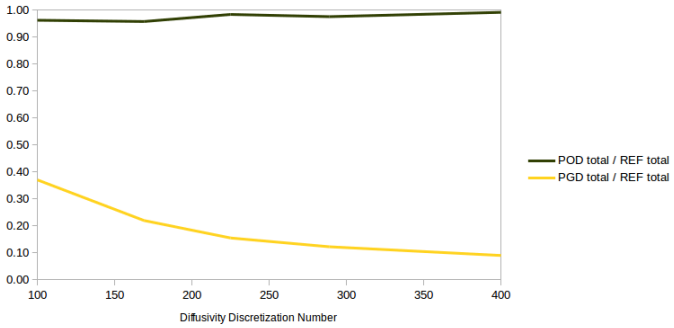


Figure 4.13: *Plot of the ratios between the reduced-order total time divided by the total reference time.*

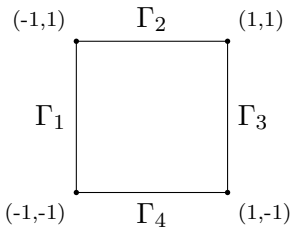


Figure 4.14: *Bidimensional domain for the considered vectorial problem*

4.4 A Vectorial Problem

4.4.1 Problem Definition

The last test problem is a vectorial diffusion equation in two dimensions. This equation can be can be assumed to model a linear elasticity problem in three dimensions, specified for a bidimensional plate of an orthotropic material. In particular, for the considered equation, the unknown represents the displacement field $\mathbf{u}(\mathbf{x})$, while the source term $\mathbf{f}$ represents the applied body forces.

The diffusion equation considered is characterized by the parametric diffusivity matrix $\mathbf{K}$, such that

$$-\mathbf{K}\nabla^2\mathbf{u}(\mathbf{x}) = \mathbf{f} \quad \textbf{in } \Omega_{\mathbf{x}}\,. \tag{4.38}$$

The bidimensional matrix $\mathbf{K}$ has 4 components – namely k_{11}, k_{12}, k_{21}, and k_{11} – but it is be symmetric, thus being characterized by only 3 independent components. Additionally, in order to ensure that the problem (4.38) is solvable, the matrix is chosen to be positive defined. In this way we ensure the coercivity of the problem.

In this context, we consider problem (4.38) where the components of the matrix are considered variable parameters. The spatial domain $\Omega_{\mathbf{x}}$ is a the square $[-1;1]^2$ shown in Figure 4.14. Exploiting the symmetry of $\mathbf{K}$, we can formulate the parametric problem as

$$-\begin{bmatrix} k_{11} & k_{12} \\ k_{12} & k_{22} \end{bmatrix} \begin{Bmatrix} \nabla^2 u_1 \\ \nabla^2 u_2 \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} \quad \text{in } \Omega \tag{4.39}$$

where the domain is defined as

$$\Omega = \Omega_{\mathbf{x}} \times \Omega_{k_{11}} \times \Omega_{k_{12}} \times \Omega_{k_{22}}\,. \tag{4.40}$$

The considered boundary conditions are defined by the displacement $\mathbf{u}$ imposed

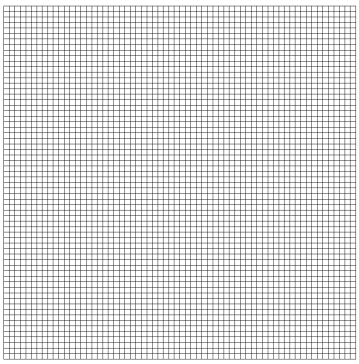


Figure 4.15: *2-D mesh generated with Gmsh for the considered vectorial problem*

to be null on the boundary Γ_2, while the body force term $\mathbf{f}$ has an imposed value on the whole domain. Thus, the problem models a 2D plate fixed at one of the edges while it is subject to an applied force. Note that the force is coplanar to the plate. The problem can be stated in a generic integral formulation, by means of integration by parts, as

$$\int_\Omega \begin{Bmatrix} \nabla^T u_1^* & \nabla^T u_2^* \end{Bmatrix} \begin{bmatrix} k_{11} & k_{12} \\ k_{12} & k_{22} \end{bmatrix} \begin{Bmatrix} \nabla u_1 \\ \nabla u_2 \end{Bmatrix} \mathrm{d}\Omega = \int_\Omega \begin{Bmatrix} u_1^* u_2^* \end{Bmatrix} \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} \mathrm{d}\Omega \tag{4.41}$$

for any suitable test function u^*, where

$$u, u^* \in H^1_{0,\Gamma_2}(\Omega_{\mathbf{x}}) \times \Omega_{k_{11}} \times \Omega_{k_{22}} \times \Omega_{k_{12}} \tag{4.42}$$

The 2D spatial domain is discretized via Gmsh [1] obtaining the mesh in Figure 4.15. Each of the three diffusivity parameters varies in a certain range so that the diffusivity matrix $\mathbf{K}$ is symmetric positive defined. In order to do so, the parameter k_{12} is chosen to vary from 1 to 4, while the parameters k_{11} and k_{22} are chosen to vary in the range $[5; 10]$.

4.4.2 Full-Order Solution

The full-order reference solution is computed for the vectorial parametric problem, discretizing the ranges of the involved parameters. In particular, we consider all the possible values assumed by the parameters k_{11}, k_{22}, k_{12} after uniformly discretizing the corresponding ranges in 8 steps. The discretization of

Table 4.5: *Greater POD eigenvalues in decreasing order for the considered vectorial problem.*

$\lambda_1 = 8.17e-02$
$\lambda_2 = 8.66e-04$
$\lambda_3 = 3.93e-17$
$\lambda_4 = 1.78e-17$
$\vdots$

the problemsis performed via a standard FEM based on the mesh in Figure 4.15 for the spatial discretization. Note that, with the chosen discretization, we obtain 729 different problems, which, even in case of coarse spatial discretization, require a considerable computational capability. The chosen finite elements are piecewise quadratic combined with a fifth order quadrature rule, thus dealing with a total number of 32282 degrees of freedom, for each subproblem for a fixed parameter. Thus, we deal with a total number of degrees of freedom of the order of magnitude of 10^7 for the whole parametric problems.

4.4.3 POD Reduced-Order Solution

The POD reduced-order solution is built by considering different combinations of the diffusivity matrix components. In more detail, the POD is performed using 729 *training samples* during the offline phase, the same adopted in the full order setting. The same discretization of the space Ω as in Section 4.4.2. Checking the eigenvalue magnitude in Table 4.5, we have that it decreases very fast, so that most of the energy of the system is retained by the first modes. Indeed, for a tolerance of $10e-12$, the number of the POD basis functions to be considered during the online phase is only 2 (see Section 2.2 for the tolerance definition).

4.4.4 PGD Reduced-Order Solution

The PGD method applied to (4.39) assumes, as usual, that the unknown field can be approximated by a separated form. The separated form for the field $\mathbf{u}(\mathbf{x}, \mathbf{K})$ reads as

$$\mathbf{u}(\mathbf{x}, \mathbf{K}) \approx \sum_{i=1}^{N} \mathbf{U}_x^i(\mathbf{x}) \circ \mathbf{U}_{k_{11}}^i(k_{11}) \circ \mathbf{U}_{k_{12}}^i(k_{12}) \circ \mathbf{U}_{k_{22}}^i(k_{22}) \,, \tag{4.43}$$

where the unknown functions associated with each subproblem are multiplied via the Hadamard product, accordingly to (3.1). In particular,the defined product reflects a separated form of its components and the coupled nature of the vectorial problem is represented by the matrix $\mathbf{K}$. In order to guarantee $\mathbf{K}$ to be positive defined, we chose the parametric discretized spaces such that $k_{11}k_{22} > k_{12}^2$, for each considered value of k_{11}, k_{12}, and k_{22}. In particular, we choose:

$$k_{11} \in [5;10]\,; \qquad k_{12} \in [1;4]\,; \qquad k_{22} \in [5;10]\,. \tag{4.44}$$

Following the process described in Section 3.2.2, at the n-th enrichment step, the approximated solution reads

$$\begin{aligned}\mathbf{u}^n(\mathbf{x},k_{11},k_{12},k_{22}) \;=\; &\sum_{i=1}^{n-1} \mathbf{U}_x^i(\mathbf{x}) \circ \mathbf{U}_{k_{11}}^i(k_{11}) \circ \mathbf{U}_{k_{12}}^i(k_{12}) \circ \mathbf{U}_{k_{22}}^i(k_{22}) \\ &+ \mathbf{U}_x^n(\mathbf{x}) \circ \mathbf{U}_{k_{11}}^n(k_{11}) \circ \mathbf{U}_{k_{12}}^n(k_{12}) \circ \mathbf{U}_{k_{22}}^n(k_{22})\end{aligned}\,. \tag{4.45}$$

The alternated direction strategy provides the computation of the unknown function starting from initial guesses for the parametric coordinates, say $\mathbf{U}_{k_{11}}^{n,0}(k_{11})$, $\mathbf{U}_{k_{12}}^{n,0}(k_{12})$, $\mathbf{U}_{k_{22}}^{n,0}(k_{22})$. Then, the scheme computes, alternately, the unknown from the already known functions.

Each p-th iteration of the fixed point loop consists in one step for each coordinate. We define q as the index associated to the "known" function, which is equal to $p-1$ for the not yet solved subproblem in the current iteration, and equal to p for the just solved ones.

The step associated with the physical space $\mathbf{x}$ consists in computing the unknown $\mathbf{U}_x^{n,p}$ from $\mathbf{U}_{k_{11}}^{n,q}$, $\mathbf{U}_{k_{12}}^{n,q}$, and $\mathbf{U}_{k_{22}}^{n,q}$. The simplest choice of the test function in (4.41) is

$$\mathbf{u}^*(\mathbf{x},k_{11},k_{12},k_{22}) \;=\; \mathbf{U}_x^{n,*} \circ \mathbf{U}_{k_{11}}^{n_q} \circ \mathbf{U}_{k_{12}}^{n,q} \circ \mathbf{U}_{k_{11}}^{n,q}\,. \tag{4.46}$$

Replacing (4.43) and (4.46) in (4.41), yields

$$\begin{aligned}&\int_\Omega \Big(\nabla U_{x,i}^{n,*} \cdot \nabla U_{x,j}^{n,p}\, U_{k_{11},i}^{n,q} U_{k_{12},i}^{n,q} U_{k_{22},i}^{n,q}\, k_{ij}\, U_{k_{11},j}^{n,q} U_{k_{12},j}^{n,q} U_{k_{22},j}^{n,q} \Big) \mathrm{d}\Omega \\ &\quad = \int_\Omega \Big(U_{x,i}^{n,*} U_{k_{11},i}^{n,q} U_{k_{12},i}^{n,q} U_{k_{22},i}^{n,q}\, f_i \\ &\quad - \sum_{e=1}^{n-1} \nabla U_{x,i}^{n,*} \cdot \nabla U_{x,j}^{e}\, U_{k_{11},i}^{n,q} U_{k_{12},i}^{n,q} U_{k_{22},i}^{n,q}\, k_{ij}\, U_{k_{11},j}^{e} U_{k_{12},j}^{e} U_{k_{11},j}^{e} \Big) \mathrm{d}\Omega\end{aligned} \tag{4.47}$$

where we use the Einstein notation to simplify the notation. Since all k_{ij}-dependent functions are known, it is possible to integrate (4.47) over the spaces $\Omega_{k_{11}}$, $\Omega_{k_{12}}$, and $\Omega_{k_{22}}$, respectively. This process can be achieved integrating, separately, the functions dependent on the single coordinates on the correspondent spaces due to the separated nature of the solution (4.43). For each (i,j)-th term in (4.47), we can define the integrated quantities over the domains $\Omega_{k_{ij}}$ as

$$\begin{aligned} K_{1_{ij}}^{k_{lm}} &= \int_{\Omega_{k_{lm}}} U_{k_{lm},i}^{n,q}\, \alpha_{ij}^{lm}\, U_{k_{lm},j}^{n,q}\, \mathrm{d}k_{lm} \\ K_{2_i}^{k_{lm}} &= \int_{\Omega_{k_{lm}}} U_{k_{lm},i}^{n,q}\, \mathrm{d}k_{lm} \\ K_{3_{ij},e}^{k_{lm}} &= \int_{\Omega_{k_{lm}}} U_{k_{lm},i}^{n,q}\, \alpha_{ij}^{lm}\, U_{k_{lm},j}^{e}\, \mathrm{d}k_{lm} \end{aligned} \quad , \tag{4.48}$$

where the terms corresponding to $(l,m) = (1,2)$ and to $(l,m) = (2,1)$ are dependent on the same variable k_{12}, due to the symmetry of the matrix K. In this way, the terms of (4.48) evaluated for $(l,m) = (2,1)$ are written with inverted (l,m) indexes, for example the first one becomes $K_{1_{21}}^{k_{21}} = K_{1_{21}}^{k_{12}}$. The coefficient α is defined as

$$\alpha_{ij}^{lm} = \begin{cases} k_{ij} & \text{if } (i,j) = (l,m) \\ 1 & \text{otherwise} \end{cases} \tag{4.49}$$

where the index (l,m) can assume only the values which define the problem parameter, i.e., $l = 1,2$ and $m = 2$. Then, we obtain 3 terms varying (l,m) – defined for each parametric coordinate – and for each one of this we have 4 terms associated with each of the 4 terms of the implicit sum in (4.47). Introducing expressions (4.48) in (4.47), we obtain:

$$\begin{aligned} &\int_\Omega \left(\nabla U_{x,i}^{n,*} \cdot \nabla U_{x,j}^{n,p}\, K_{1_{ij}}^{k_{11}}\, K_{1_{ij}}^{k_{12}}\, K_{1_{ij}}^{k_{22}} \right) \mathrm{d}\mathbf{x} \\ &\quad = \int_\Omega \left(U_{x,i}^{n,*}\, K_{2_i}^{k_{11}}\, K_{2_i}^{k_{12}}\, K_{2_i}^{k_{22}}\, f_i \right. \\ &\quad \left. - \sum_{e=1}^{n-1} \nabla U_{x,i}^{n,*} \cdot \nabla U_{x,j}^{e}\, K_{3_{ij},e}^{k_{11}}\, K_{3_{ij},e}^{k_{12}}\, K_{3_{ij},e}^{k_{22}} \right) \mathrm{d}\mathbf{x} \end{aligned} \quad . \tag{4.50}$$

Then, it is possible to numerically solve the sub-problem (4.50) with the preferred technique, such as a finite element method.

The subsequent three steps for the current p-th nonlinear loop iteration require the sequential solution of three different problems depending on each

parameter, i.e., k_{11} , k_{12}, and k_{22}. First, consider the $(1,1)$-th diffusivity component as unknown, i.e., $\mathbf{U}^{n,p}_{k_{11}}(k_{11})$, and compute it from the previously computed $\mathbf{U}^{n,p}_{x}(\mathbf{x})$ and $\mathbf{U}^{n,q}_{k_{ij}}(k_{ij})$ – with $(i,j) \neq (1,1)$ – considered known. The test function to be used in (4.41) becomes

$$\mathbf{u}^*(\mathbf{x},k) \;=\; \mathbf{U}^{n,*}_{k_{11}}(k_{11}) \circ \mathbf{U}^{n,p}_{x}(\mathbf{x}) \circ \mathbf{U}^{n,q}_{k_{12}}(k_{12}) \circ \mathbf{U}^{n,q}_{k_{22}}(k_{22}) \,. \tag{4.51}$$

Combining (4.43) and (4.51) in (4.41) yields

$$\begin{aligned}
&\int_\Omega \Big(\nabla U^{n,p}_{x,i} \cdot \nabla U^{n,p}_{x,j}\, U^{n,*}_{k_{11},i}\, U^{n,q}_{k_{12},i}\, U^{n,q}_{k_{22},i}\, k_{ij}\, U^{n,q}_{k_{11},j}\, U^{n,q}_{k_{12},j}\, U^{n,q}_{k_{22},j} \Big) \mathrm{d}\Omega \\
&= \int_\Omega \Big(U^{n,p}_{x,i} U^{n,*}_{k_{11},i}\, U^{n,q}_{k_{12},i}\, U^{n,q}_{k_{22},i}\, f_i \\
&- \sum_{e=1}^{n-1} \nabla U^{n,p}_{x,i} \cdot \nabla U^{e}_{x,j}\, U^{n,*}_{k_{11},i}\, U^{n,q}_{k_{12},i}\, U^{n,q}_{k_{22},i}\, k_{ij}\, U^{e}_{k_{11},j}\, U^{e}_{k_{12},j}\, U^{e}_{k_{22},j} \Big) \mathrm{d}\Omega
\end{aligned} \tag{4.52}$$

Similarly to (4.47), all the $\mathbf{x}$-dependent and k_{ij}-dependent functions, for $(i,j) \neq (1,1)$, are known. So it is possible to integrate (4.52) over the spaces Ω_x and $\Omega_{k_{ij}}$. Then, defining the known quantities integrated over the spatial domain Ω_x as

$$\begin{aligned}
K^x_{1_{ij}} &= \int_{\Omega_x} \nabla U^{n,p}_{x,i} \cdot \nabla U^{n,p}_{x,j} \mathrm{d}\mathbf{x} \\
K^x_{2_i} &= \int_{\Omega_x} U^{n,p}_{x,i} \mathrm{d}\mathbf{x} \\
K^x_{3_{ij},e} &= \int_{\Omega_x} \nabla U^{n,p}_{x,i} \cdot \nabla U^{e}_{x,j} \mathrm{d}\mathbf{x}
\end{aligned} \tag{4.53}$$

and using (4.48), equation (4.52) becomes

$$\begin{aligned}
&\int_{\Omega_{k_{11}}} \Big(U^{n,*}_{k_{11},1}\, k_{11}\, U^{n,p}_{k_{11},1}\, K^{x}_{1_{11}}\, K^{k_{12}}_{1_{11}}\, K^{k_{22}}_{1_{11}} \;+\; U^{n,*}_{k_{11},1}\, U^{n,p}_{k_{11},2}\, K^{x}_{1_{12}}\, K^{k_{12}}_{1_{12}}\, K^{k_{22}}_{1_{12}} \\
&+\; U^{n,*}_{k_{11},2}(U^{n,p}_{k_{11},1}\, K^{x}_{1_{21}}\, K^{k_{12}}_{1_{21}}\, K^{k_{22}}_{1_{21}} \;+\; U^{n,p}_{k_{11},2}\, K^{x}_{1_{22}}\, K^{k_{12}}_{1_{22}}\, K^{k_{22}}_{1_{22}}) \Big)\, \mathrm{d}k_{11} \\
&= \int_{\Omega_{k_{11}}} \Big(U^{n,*}_{k_{11},1}\, K^{x}_{2_1}\, K^{k_{12}}_{2_1}\, K^{k_{22}}_{2_1}\, f_1 \;+\; U^{n,*}_{k_{11},2}\, K^{x}_{2_2}\, K^{k_{12}}_{2_2}\, K^{k_{12}}_{2_2}\, f_2 \\
&-\; \sum_{e=1}^{n-1} \Big(U^{n,*}_{k_{11},1}\, k_{11}\, U^{e}_{k_{11},1}\, K^{x}_{3_{11},e}\, K^{k_{12}}_{3_{11},e}\, K^{k_{22}}_{3_{11},e} \;+\; U^{n,*}_{k_{11},1}\, U^{e}_{k_{11},2}\, K^{x}_{3_{12},e}\, K^{k_{12}}_{3_{12},e}\, K^{k_{22}}_{3_{12},e} \\
&+\; U^{n,*}_{k_{11},2}(U^{e}_{k_{11},1}\, K^{x}_{3_{21},e}\, K^{k_{12}}_{3_{21},e}\, K^{k_{22}}_{3_{21},e} \;+\; U^{n,*}_{k_{11},2}\, K^{x}_{3_{22},e}\, K^{k_{12}}_{3_{22},e}\, K^{k_{22}}_{3_{22},e}) \Big) \Big)\, \mathrm{d}k_{11}
\end{aligned} \tag{4.54}$$

We repeat the same computations, to highlight the dependence on k_{22}. The resulting problem will be

$$\begin{aligned}
&\int_{\Omega_{k_{22}}} \Big(U^{n,*}_{k_{22},2}\, k_{22}\, U^{n,p}_{k_{22},2}\, K^{x}_{1_{22}}\, K^{k_{12}}_{1_{22}}\, K^{k_{11}}_{1_{22}} \;+\; U^{n,*}_{k_{22},2}\, U^{n,p}_{k_{22},1}\, K^{x}_{1_{21}}\, K^{k_{12}}_{1_{21}}\, K^{k_{11}}_{1_{21}} \\
&+\; U^{n,*}_{k_{22},1}(U^{n,p}_{k_{22},2}\, K^{x}_{1_{12}}\, K^{k_{12}}_{1_{12}}\, K^{k_{22}}_{1_{12}} \;+\; U^{n,p}_{k_{22},1}\, K^{x}_{1_{11}}\, K^{k_{12}}_{1_{11}}\, K^{k_{22}}_{1_{11}}) \Big)\, \mathrm{d}k_{22} \\
&= \int_{\Omega_{k_{22}}} \Big(U^{n,*}_{k_{22},1}\, K^{x}_{2_1}\, K^{k_{12}}_{2_1}\, K^{k_{11}}_{2_1}\, f_1 \;+\; \hat{U}^{k_{22}}_{2,n}\, K^{x}_{2_2}\, K^{k_{12}}_{2_2}\, K^{k_{11}}_{2_2}\, f_2 \\
&-\; \sum_{e=1}^{n-1} \Big(U^{n,*}_{k_{22},2}\, k_{22}\, U^{e}_{k_{22},2}\, K^{x}_{3_{22},e}\, K^{k_{12}}_{3_{22},e}\, K^{k_{22}}_{e_{22},e} \;+\; U^{n,*}_{k_{22},2}\, U^{e}_{k_{22},1}\, K^{x}_{3_{21},e}\, K^{k_{12}}_{3_{21},e}\, K^{k_{22}}_{3_{21},e} \\
&+\; U^{n,*}_{k_{22},1}(U^{e}_{k_{22},2}\, K^{x}_{3_{12},e}\, K^{k_{12}}_{3_{12},e}\, K^{k_{22}}_{3_{12},e} \;+\; U^{e}_{k_{22},1}\, K^{x}_{3_{11},e}\, K^{k_{12}}_{3_{11},e}\, K^{k_{11}}_{3_{11},e}) \Big) \Big)\, \mathrm{d}k_{22}
\end{aligned} \tag{4.55}$$

Considering now the extra-diagonal term, we have the presence of two different terms depending on the unknown k_{12} due to the imposed symmetry. The

associated problem, similarly to the other terms, results

$$\begin{aligned}
&\int_{\Omega_{k_{12}}} \Big(k_{12} (U^{n,*}_{k_{12},1}\, U^{n,p}_{k_{12},2}\, K^{x}_{1_{12}}\, K^{k_{11}}_{1_{12}}\, K^{k_{22}}_{1_{12}} \,+\, U^{n,*}_{k_{12},2}\, U^{n,p}_{k_{12},1}\, K^{x}_{1_{21}}\, K^{k_{11}}_{1_{21}}\, K^{k_{22}}_{1_{21}}) \\
&+\, U^{n,*}_{k_{12},1}\, U^{n,p}_{k_{12},1}\, K^{x}_{1_{11}}\, K^{k_{11}}_{1_{11}}\, K^{k_{22}}_{1_{11}} \,+\, U^{n,*}_{k_{12},2}\, U^{n,p}_{k_{12},2}\, K^{x}_{1_{22}}\, K^{k_{11}}_{1_{22}}\, K^{k_{22}}_{1_{22}} \Big)\, \mathrm{d}k_{12} \\
&= \int_{\Omega_{k_{12}}} \Big(U^{n,*}_{k_{12},1}\, K^{x}_{2_1}\, K^{k_{11}}_{2_1}\, K^{k_{22}}_{2_1}\, f_1 \,+\, U^{n,*}_{k_{12},2}\, K^{x}_{2_2}\, K^{k_{11}}_{2_2}\, K^{k_{22}}_{2_2}\, f_2 \\
&-\, \sum_{e=1}^{n-1} \Big((U^{n,*}_{k_{12},1}\, U^{e}_{k_{12},2}\, K^{x}_{2_{12},e}\, K^{k_{11}}_{2_{12},e}\, K^{k_{22}}_{2_{12},e} \,+\, U^{n,*}_{k_{12},2}\, U^{e}_{k_{12},1}\, K^{x}_{2_{21},e}\, K^{k_{11}}_{2_{21},e}\, K^{k_{22}}_{2_{21},e}) \\
&+\, U^{n,*}_{k_{12},1}\, U^{e}_{k_{12},1}\, K^{x}_{2_{11},e}\, K^{k_{11}}_{2_{11},e}\, K^{k_{22}}_{2_{11},e} \,+\, U^{n,*}_{k_{12},2}\, U^{e}_{k_{12},2}\, K^{x}_{2_{22},e}\, K^{k_{11}}_{2_{22},e}\, K^{k_{22}}_{2_{22},e} \Big) \Big)\, \mathrm{d}k_{12}
\end{aligned} \tag{4.56}$$

For each one of the three subproblems associated with the diffusivity matrix components, we obtain an equation which does not involve any differential operator, since (4.41) does not involve any derivative with respect to k_{ij}. So, we can formulate the strong form associated with (4.54), (4.55), and (4.56), obtaining, for each of the unknown an algebraic equation which can be easily solved.

Iterating the above procedure until reaching a fixed point for the nonlinear loop by satisfying a specific stopping criterion (see Section 3.2.3) leads to identify the n-th enrichment function product

$$\mathbf{U}^{x}_{n}(\mathbf{x})\, \mathbf{U}^{k_{11}}_{n}(k_{11})\, \mathbf{U}^{k_{12}}_{n}(k_{12})\, \mathbf{U}^{k_{22}}_{n}(k_{22}) \tag{4.57}$$

by the assignment of the last non-linear iteration solution, i.e. $\mathbf{U}^{n}_{x}(\mathbf{x}) \leftarrow \mathbf{U}^{n,p}_{x}(\mathbf{x})$, $\mathbf{U}^{n}_{k_{11}}(k_{11}) \leftarrow \mathbf{U}^{n,p}_{k_{11}}(k_{11})$, $\mathbf{U}^{n}_{k_{12}}(k_{12}) \leftarrow \mathbf{U}^{n,p}_{k_{12}}(k_{12})$, and $\mathbf{U}^{n}_{k_{22}}(k_{22}) \leftarrow \mathbf{U}^{n,p}_{k_{22}}(k_{22})$.

4.4.5 Performance Comparisons

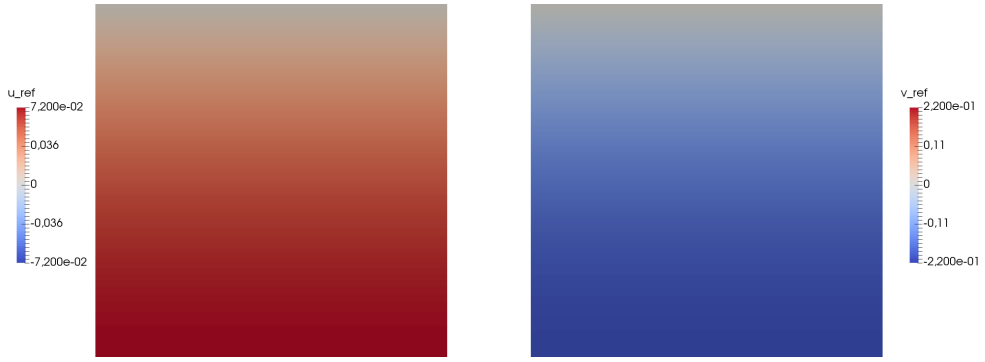
The full-order solutions require to solve different problems, one for each value of the discretized diffusivity coefficient. In order to compare the time needed to compute the full-order solutions with the time needed by the reduced-order methods, we reconstruct the online solutions for all the (diffusivity) parameter values used during the full-order computation. In this way, we compare the time needed by the three different approaches to compute the same number of solutions.

In Table 4.6 the results obtained with a sequence of serial simulations confirming that the reduced models provide a reduction of the computational costs with respect to the full-order solution. The performance is evaluated via the total time required by all the simulations, and via the average time required by each simulation associated with one of the 729 possible choices for the parameters. The POD performance indices are the time required by the POD offline stage, the time required by the POD online stage, and the ratio between the total time needed by the POD and the full-order FEM to compute the same number of simulations. The same performance indices are considered for the PGD with the addition of details on the number of enrichment steps and average nonlinear steps required during the PGD offline stage.

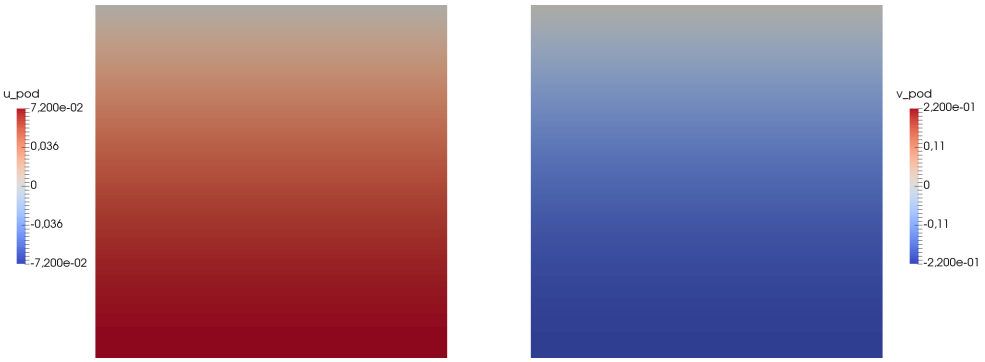
During the PGD offline phase the tolerances used for both the nonlinear loop and the enrichment loop are set to $1e-4$ (see Section 3.2.3 for more details on the stopping criteria).

The computational time needed to perform the PGD offline construction and the online evaluation is about the 1% of the total time needed to perform the classical FEM simulations. This huge computational reduction underlines the potentialities of the PGD method in the context of parametric problems, even if the error of the PGD reduced order solution is meanly high compared to the POD reduced order reference solution (see Figure 4.17). The PGD online average computational time is about the 0.1% of the average time needed by the full-order simulations, like in the previous tests. Comparing the PGD performances with the reference reduced-order POD simulations we notice that the computational saving of the PGD method is considerable for both the offline and the online phases. Even if the total time needed by the POD is about one third of the reference classical FEM, which represents a great computational saving, the PGD method compute the same amount of information in about one third of the time demanded by the reference POD method. In particular, concerning the online time required to reconstruct the reduced solutions, we notice that PGD performs in a lower time, confirming its capability for a drastic reduction of the "on-the-fly" computational cost.

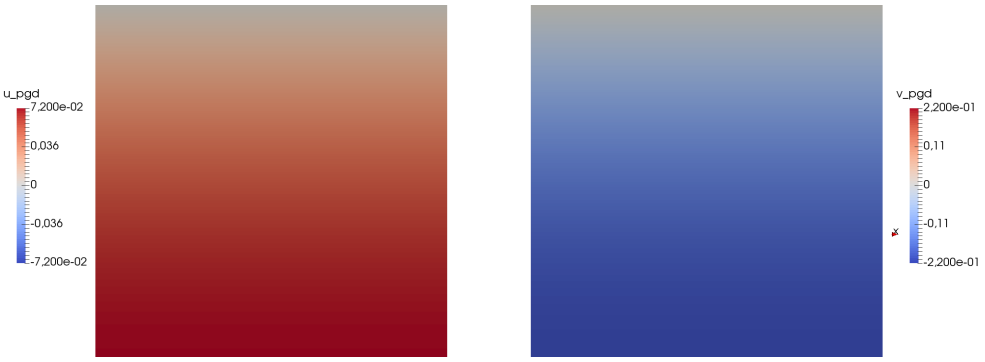
The qualitative results of the numerical simulations are compared for a possible combination of the diffusivity coefficients. In particular, from the 729 possible combinations of the 3 parameters, we choose the values $k_{11} = 10$, $k_{22} = 5$, and $k_{12} = 1$. The results are shown in Figure 4.16, and the associated errors in Figure 4.17, where the solution and the pointwise error associated with the different model reduction techniques are plotted for both the components, namely u and v. The pointwise error defined is simply the absolute value of the difference between the full order solution and the reduced-order solutions. It is evident that the excellent performance of PGD are paid with an accuracy of the solution which is two order of magnitude lower.



(a) *Reference full-order solution,* u *(left) and* v *(right) components*



(b) *Reference POD reduced-order solution,* u *(left) and* v *(right) components*



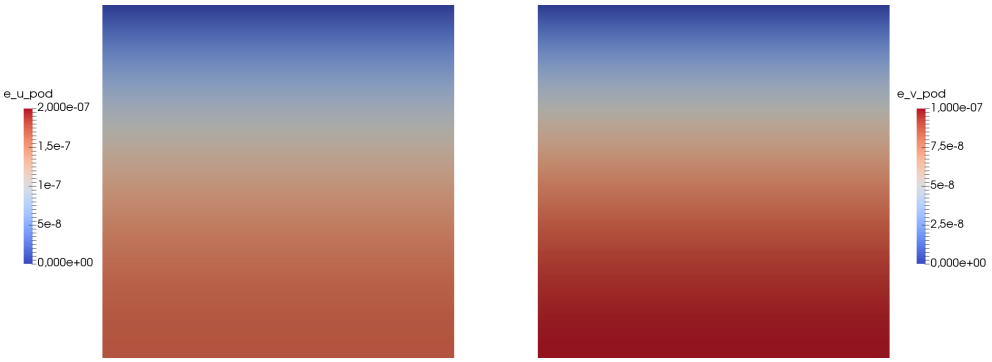
(c) *PGD reduced-order solution,* u *(left) and* v *(right) components*

Figure 4.16: *Linear elasticity problem for* $k_{11} = 10$*,* $k_{22} = 5$*, and* $k_{12} = 1$*.*

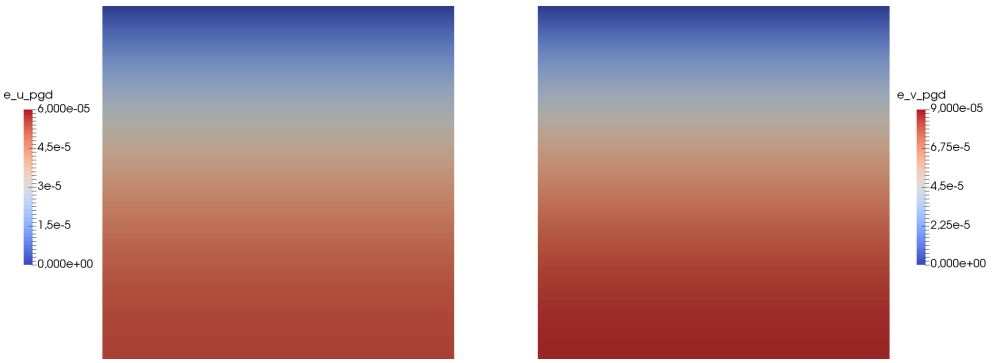
Table 4.6: *Performances comparison among full-order solution, POD reduced-order solution, and PGD reduced-order solution for the considered linear elasticity problem. Results on Linux 4.2.0-27-generic x86_64 on Intel® Core™i5-2500K CPU @ 3.30 GHz.*

Full-order simulation	*average time on* 729	2.64339	s
Full-order total time		1927	s
POD reduced-order	*offline time*	629.4290	s
	online average time	0.01247	s
	online total time	9.08844	s
POD reduced-order total time		638.51744	s
POD total time over full-order total time ratio		33.13	%
POD average online time over full-order average time ratio		0.47	%
PGD reduced-order	*offline time*	21.1754	s
	online total time	2.16451	s
	online average time	0.00297	s
PGD reduced-order total time		23.33991	s
PGD number of enrichment steps		3	
PGD average number of nonlinear steps		1	
PGD total time over full-order total time ratio		1.12	%
PGD average online time over full-order average time ratio		0.11	%

Thus, the PGD algorithm confirms its better computational performances compared to the POD reduced-order approach, demanding, in particular, a reduced computational time during both the offline and the online phases. Analogously to the Graetz problem, we expect that the computational cost saving given by PGD – with respect to POD – increases when dealing with parametric discretized spaces of higher dimension.



(a) *POD reduced-order solution pointwise error,* u *(left) and* v *(right) components*



(b) *PGD reduced-order solution pointwise error,* u *(left) and* v *(right) components*

Figure 4.17: *Linear elasticity problem pointwise errors associated to the reduced models for* $k_{11} = 10$*,* $k_{22} = 5$*, and* $k_{12} = 1$*.*

Conclusion 5

The PGD approach introduces a computational cost adding a nonlinearity to the problem (see Section 3.2). During the implementation of the method we noticed that the nonlinear loop is one of the key issues of the PGD method. Indeed, the solution of this loop requires to pay a particular attention in finding the stopping criterion which is sensitive to variations in the problem, even if the computational cost added by the presence of this loop is negligible compared with the benefits led by the construction of the reduced order basis during the offline phase. Moreover, the problem computed with the PGD method is subject to numerical stability issues in case of coarse discretizations, and, in this case, the computational cost added by the nonlinear loop is not negligible. However, thanks to the separated nature of the method, it is possible to solve a parametric problem for a huge number of parameter discretization steps, obtaining a global reduced computational cost – offline plus online phases – which is competitive compared with the one demanded by the well-established POD method. This feature has been noticed in the numerical assessment, in particular in (Section 4.3) where the results show the drastic computational saving when increasing the number of the discretization steps. Also the physical space discretization affects the results, even if we focus on the dependence of the method on the parametric discretization.

It is evident that the PGD method has strong potentialities despite the introduced nonlinearity. The pointwise error – defined simply as the absolute value of the difference between the full order solution and the reduced-order solutions – characterizing the PGD solutions are likely associated with the lack of an "a priori" error estimator which does not help in selecting an appropriate stopping criterion for the enrichment loop. As a consequence, it is difficult to limit the error generated during the computation of the PGD reduced-solution,

at least with a general approach based on simple implementations of the single subproblems (see Chapter 3). Indeed, we develop an implementation of the method oriented at the role of "outer shell", thinking at the possibility of deciding the preferred and independent solver for each subproblem as well as the solver of the nonlinear loop. The last one – implemented as a simple fixed point alternating direction scheme in Section 3.2.2 – is one of the crucial points we find to be responsible of the low robustness of the PGD. A further analysis on the nonlinear solver could probably lead to results with a lower error.

The results obtained underline the very low computational costs required by both the POD and the PGD reduced order methods during the online solution reconstruction, which is a fundamental requirement in the ROM oriented to an offline-online splitting of the algorithm and it allows the online phase to be performed also on low performance devices. However, if we compare the computational cost needed to perform the offline computation with the POD and the PGD methods, we notice a drastic reduction of the PGD offline time with respect to the POD reduced order method, in particular way for high parameter discretization dimensions. Then, assuming that this behaviour characterizes the method performances even for more complex problems, the PGD method offline computational cost can be much more than competitive with respect to a classical reduced order approach. Indeed, the PGD allows the computation to be achievable at a lower computational performance platforms, making the computational cost reachable for a greater number of industrial applications.

In general, from the mathematical point of view, a lot of work needs to be done in order to develop a reliable and effective tool to be used in parallel with reliable well-tested methods like the POD. In view of the results obtained during this work, we can assert that a "raw" PGD method can be a good starting point for investigating this family of methods. Indeed, the separated character of the method allows to define the appropriate arbitrary method for the single subproblem in order to best solve the issues – are either associated to the physics or to the numerical complexity – introduced by its variable dependence. The key point is that this target can be achieved using well-tested solvers, alternatively called as "black boxes", by the PGD outer shell, concentrating the attention on the behaviour of these solvers for the specified subproblems and on the coupling effects of the different selected solvers.

Bibliography

[1] Gmsh open source software web page. `http://gmsh.info/`.

[2] libMesh open source software library web page. `http://libmesh.github.io/`.

[3] Vedat S. Arpaci. Condustion heat transfer. 1966.

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

[5] Maxime Barrault, Yvon Maday, Ngoc Cuong Nguyen, and Anthony T. Patera. An "empirical interpolation" method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667–672, 2004.

[6] Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015.

[7] Robert Byron Bird, Robert Calvin Armstrong, Ole Hassager, and Charles F. Curtiss. *Dynamics of polymeric liquids*, volume 1. Wiley New York, 1977.

[8] Brice Bognet, Felipe Bordeu, Francisco Chinesta, Adrien Leygue, and Arnaud Poitou. Advanced simulation of models defined in plate geometries: 3D solutions with 2D computational complexity. *Computer Methods in Applied Mechanics and Engineering*, 201:1–12, 2012.

[9] John Burkardt, Max Gunzburger, and Hyung-Chun Lee. POD and CVT-based reduced-order modeling of navier–stokes flows. *Computer Methods in Applied Mechanics and Engineering*, 196(1):337–355, 2006.

[10] Eric Cances, Mireille Defranceschi, Werner Kutzelnigg, Claude Le Bris, and Yvon Maday. Computational quantum chemistry: a primer. *Handbook of Numerical Analysis*, 10:3–270, 2003.

[11] Saifon Chaturantabut and Danny C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.

[12] Francisco Chinesta, Amine Ammar, and Elías Cueto. Recent advances and new challenges in the use of the proper generalized decomposition for solving multidimensional models. *Archives of Computational Methods in Engineering*, 17(4):327–350, 2010.

[13] Francisco Chinesta, Antonio Huerta, Gianluigi Rozza, and Karen Willcox. Model order reduction. *Encyclopedia of Computational Mechanics*, 2016.

[14] Francisco Chinesta, Roland Keunings, and Adrien Leygue. *The proper generalized decomposition for advanced numerical simulations: a primer*. Springer Science & Business Media, 2013.

[15] Francisco Chinesta and Pierre Ladevèze. *Separated Representations and PGD-Based Model Reduction: Fundamentals and Applications*. CISM International Centre for Mechanical Sciences. Springer Vienna, 2014.

[16] Francisco Chinesta, Adrien Leygue, Felipe Bordeu, Jose V. Aguado, Elías Cueto, David González, Ivan Alfaro, Amine Ammar, and Antonio Huerta. PGD-based computational vademecum for efficient design, optimization and control. *Archives of Computational Methods in Engineering*, 20(1):31–59, 2013.

[17] Chandler Davis. The norm of the Schur product operation. *Numerische Mathematik*, 4(1):343–344, 1962.

[18] Simone Deparis, Davide Forti, and Alfio Quarteroni. A rescaled localized radial basis function interpolation on non-cartesian and non-conforming grids. *SIAM Journal on Scientific Computing*, 36(6), 2014.

[19] Joseph Fourier. Mémoire sur la propagation de la chaleur dans les corps solides. *Nouveau Bulletin des Sciences de la Société Philomathique de Paris*, 6:112–116, 1808.

[20] David González, Amine Ammar, Francisco Chinesta, and Elías Cueto. Recent advances on the use of separated representations. *International Journal for Numerical Methods in Engineering*, 81(5):637–659, 2010.

[21] von L Graetz. Ueber die wärmeleitungsfähigkeit von flüssigkeiten. *Annalen der Physik*, 254(1):79–94, 1882.

[22] Max D. Gunzburger. *Perspectives in flow control and optimization*, volume 5. Siam, 2003.

[23] Jan S. Hesthaven, Gianluigi Rozza, and Benjamin Stamm. Reduced basis methods. In *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*, pages 27–43. Springer, 2016.

[24] Anwar Koshakji. Free Form Deformation techniques for 3D shape optimization problems. Master's thesis, Politecnico di Milano, Italy, 2010.

[25] Karl Kunisch and Stefan Volkwein. Galerkin proper orthogonal decomposition methods for parabolic problems. *Numerische mathematik*, 90(1):117–148, 2001.

[26] Karl Kunisch and Stefan Volkwein. Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. *SIAM Journal on Numerical analysis*, 40(2):492–515, 2002.

[27] Pierre Ladevèze. The large time increment method for the analysis of structures with non-linear behavior described by internal variables. *Comptes Rendus de l'Academie des Sciences Serie II*, 309(11):1095–1099, 1989.

[28] Pierre Ladevèze. A computational technique for the integrals over the time-space domain in connection with the latin method. Technical report, Tech. Rep. 193, LMT-Cachan, 1997.

[29] Pierre Ladevèze, Jean-Charles Passieux, and David Néron. The latin multiscale computational method and the proper generalized decomposition. *Computer Methods in Applied Mechanics and Engineering*, 199(21):1287–1296, 2010.

[30] Toni Lassila, Andrea Manzoni, Alfio Quarteroni, and Gianluigi Rozza. Model order reduction in fluid dynamics: challenges and perspectives. In *Reduced Order Methods for Modeling and Computational Reduction*, volume 9 of *MS&A - Modeling, Simulations and Applications*, pages 235–273. Springer, 2014.

[31] Andrea Manzoni, Alfio Quarteroni, and Gianluigi Rozza. Computational reduction for parametrized PDEs: strategies and applications. *Milan Journal of Mathematics*, 80(2):283–309, 2012.

[32] Juan Pedro Milaszewicz. Improving jacobi and gauss-seidel iterations. *Linear Algebra and Its Applications*, 93:161–170, 1987.

[33] Siamak Niroomandi, David González, Ivan Alfaro, Felipe Bordeu, Adrien Leygue, Elías Cueto, and Francisco Chinesta. Real-time simulation of biological soft tissues: a PGD approach. *International Journal for Nnumerical Methods in Biomedical Engineering*, 29(5):586–600, 2013.

[34] Anthony Nouy. Proper generalized decompositions and separated representations for the numerical solution of high dimensional stochastic problems. *Archives of Computational Methods in Engineering*, 17(4):403–434, 2010.

[35] Karl Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[36] Etienne Pruliere, Francisco Chinesta, and Amine Ammar. On the deterministic solution of multidimensional parametric models using the proper generalized decomposition. *Mathematics and Computers in Simulation*, 81(4):791–810, 2010.

[37] Alfio Quarteroni. *Numerical models for differential problems*, volume 2. Springer Science & Business Media, 2010.

[38] Alfio Quarteroni, Gianluigi Rozza, and Andrea Manzoni. Certified reduced basis approximation for parametrized partial differential equations and applications. *Journal of Mathematics in Industry*, 1(1):1–49, 2011.

[39] Alfio Quarteroni and Fausto Saleri. *Scientific Computing with Matlab*, volume 2 of *Texts in Computational Science and Engineering*. Springer, 2003.

[40] Sivaguru S. Ravindran. A reduced-order approach for optimal control of fluids using proper orthogonal decomposition. *International journal for numerical methods in fluids*, 34(5):425–448, 2000.

[41] Gianluigi Rozza. Fundamentals of reduced basis method for problems governed by parametrized pdes and applications. In *Separated Representations and PGD-Based Model Reduction*, pages 153–227. Springer, 2014.

[42] Gianluigi Rozza, Dinh Bao Phuong Huynh, and Anthony 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(3):229–275, 2008.

[43] Jesús M. Sanz-Serna and Marc N. Spijker. Regions of stability, equivalence theorems and the Courant-Friedrichs-Lewy condition. *Numerische Mathematik*, 49(2-3):319–329, 1986.

[44] T. W. Sederberg and S. R. Parry. Free-Form Deformation of solid geometric models. In *Proceedings of SIGGRAPH - Special Interest Group on GRAPHics and Interactive Techniques*, pages 151–159. SIGGRAPH, 1986.

[45] Lawrence Sirovich. Turbulence and the dynamics of coherent structures. part i: Coherent structures. *Quarterly of applied mathematics*, 45(3):561–571, 1987.

[46] Jeroen A.S. Witteveen. Explicit and robust Inverse Distance Weighting mesh deformation for CFD. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*. AIAA, 2010.

[47] Jeroen A.S. Witteveen and Hester Bijl. Explicit mesh deformation using Inverse Distance Weighting interpolation. In *19th AIAA Computational Fluid Dynamics*. AIAA, 2009.