Advanced strategies for the separated formulation of problems in the Proper Generalized Decomposition framework

José Vicente Aguado

l'université nantes angers le mans
PÔLE DE RECHERCHE ET D'ENSEIGNEMENT SUPÉRIEUR

Centrale Nantes

Thèse de Doctorat

José Vicente AGUADO

Mémoire présenté en vue de l'obtention du
grade de Docteur de l'Ecole Centrale de Nantes
sous le label de L'Université Nantes Angers Le Mans

École doctorale : Sciences Pour l'Ingénieur, Géosciences, Architecture

Discipline : mécanique des solides, des matériaux, des structures et des surfaces
Unité de recherche : ***Institut de Recherche en Génie Civil et Mécanique, UMR 6183 CNRS***

Soutenue le 10 juin 2015

Advanced strategies for the separated formulation of problems in the Proper Generalized Decomposition framework

JURY

Président :	**Antonio HUERTA**, Professeur des Universités, Universitat Politècnica de Catalunya
Rapporteurs :	**Pedro DÍEZ**, Professeur des Universités, Universitat Politècnica de Catalunya **Aziz HAMDOUNI**, Professeur des Universités, Université de la Rochelle
Examinateurs :	**Elias CUETO,** Professeur des Universités, Universidad de Zaragoza **Antonio HOSPITALER**, Professeur des Universités, Universitat Politècnica de València
Directeur de Thèse :	**Francisco CHINESTA**, Professeur des Universités, Ecole Centrale Nantes
Co-encadrant de Thèse :	**Adrien LEYGUE**, Chargé de Recherche, Ecole Centrale Nantes

A mis padres, Vicente y Sara.
A mes parents, Vicente et Sara.

Contents

Table of figures

List of Algorithms

Introduction

After the impressive progresses in computational power made in the last two decades, it might be thought that there is nothing that escapes to the limits of computability. We live in a time in which incredibly realistic animations and astonishing visual effects can be produced using computers but we are still not able to simulate —in the engineering sense of the word— many apparently banal physical phenomena. Of course, there is huge gulf between mimicking reality and truly simulating it. Limitations come not only from physics but also from applied mathematics and even from computational sciences. Computational Mechanics is a discipline placed somewhere in between these three branches of science whose goal is to conceive computational methods to solve real engineering problems.

The engineers' wish of simulating the more and more complex problems scales exponentially so as the computational power does. But it has to be emphasized that it is not just because the computational power increases that we are able to simulate more *realistic* problems. And this is specially true when considering physical models which are naturally multidimensional such as those describing kinetics of complex materials, competition in biological systems, social dynamics and economics systems, among many others. In those cases, *realistic* is a synonym of *multidimensional*. Multidimensional models are also encountered in the context of both stochastic and parametric partial differential equations (PDE), the last of which is of great interest for the manufacturing industry, as parametric PDE are related to very general tasks such as design, optimization or inverse analysis. Solving multidimensional models of any nature implies many challenges, the most basic involving the *curse of dimensionality* concept. First coined by Richard E. Bellman in [20], the curse of dimensionality refers to the exponential growth of the computational complexity of problems defined in many dimensions.

It is worth to emphasize that the difficulty regarding multidimensional models does not necessarily concern the *solvability* of the equations. Indeed, in many cases there are well-established methods allowing for the resolution of such models in small dimension, but these methods fail in addressing multidimensional models because they were not designed to face this context. It is the aim of engineers not only to create comprehensive and realistic models but to be able to compute them, and regarding multidimensional models this implies creat-

ing appropriate computational methods paying special attention to data representation and algorithmic complexity with regard to the available computational ressources. During last years, several methods have been proposed to address multidimensional models efficiently. Among them, this thesis is particularly concerned with the Proper Generalized Decomposition (PGD) method, which is based on separated data representations. Although several important developments have been recently made in this area, the separated formulation of problems, which is a critical requirement for the efficiency of PGD, remains challenging in many cases. These work proposes new advanced techniques allowing for the separated problem formulation of transient problems involving wave-like excitations and nonlinear models.

The contents are structured as follows. In Chapter 1 we provide a comprehensive review on the use of separated representations as a means to understand the strengths, but also the current limitations, of the PGD method. A variety of topics ranging from Computational Mechanics to Applied mathematics are covered. Chapter 2 proposes an efficient formulation of structural dynamics problems in the frequency domain. In Chapter 3, the frequency-domain approach is further developed so as to overcome the non-separability in the space-time domain of wave-like excitations. The problem formulation is then naturally separated. Besides, the reciprocity principle can be proven thanks to the symmetrization introduced by the frequency-based formulation, yielding a direct application for the real-time monitoring of processes. Finally, an innovative, general purpose method to compute the separated representations of multivariate functions is presented in Chapter 4. Among its applications, we refer to both model coefficients and operators separability in nonlinear problems, which are required conditions in order to formulate a separated representation of the problem.

Chapter 1

State of the art in separated representations

This chapter provides a comprehensive review on the use of separated representations as a means to understand the strengths, but also the current limitations, of Proper Generalized Decomposition. To this end, a variety of topics concerning different scientific communities are covered here. They range from Model Order Reduction, which is a discipline of Computational Mechanics, to Low-rank Tensor Approximations, which is closer to Applied Mathematics. Although these communities wish to attain basically the same objectives, they provide different approaches to problems that are essentially equivalent. Hence, the goal is not only to present both approaches but to elucidate the links between them.

Separated representations are most naturally introduced in the context of tensor product spaces, but they can also be regarded as a tool for reducing the computational complexity through Model Order Reduction methods, or as a manner of compressing tensors via low-rank decompositions. Our exposition, however, does not seek to be exhaustive. The topics are chosen according to the matter of concern of this work, and thus Proper Generalized Decomposition is highly privileged.

A final idea that we would like to support in this chapter is that of Proper Generalized Decomposition being placed at the interface between Model Order Reduction and Low-rank Tensor Approximations.

Contents

1.1 The parametrized problem as an illustrative example

If we had to state in few words what is the aim of people working in Model Order Reduction (MOR) [13, 22, 38] methods and Low-rank approximation methods [62, 80], it could be expressed as the wish of computing arbitrarily precise approximations of the solution of a problem while the computational complexity is reduced. Both communities share many aspects but still they are different. In this section we shall compare both approaches and we will try to elucidate the links between them.

We shall begin the exposition by considering a parametrized problem to which MOR methods wish to be applied. Let us introduce a generic parametrized problem using the usual setting and notation of computational mechanics: find $u \in V_s$, some appropriate approximation space, such that

$$a(u, w; \mu) = l(w), \tag{1.1}$$

for every $w \in V_s$ and some $\mu \in I_\mu$. Bilinear and linear forms are denoted by $a(\cdot, \cdot) : V_s \times V_s \to \mathbb{K}$ and $l(\cdot) : V_s \to \mathbb{K}$, respectively. The parametric dependence is denoted inside the bilinear form after the semicolon. For the sake of concretion, throughout this section we shall only consider approximation spaces of finite dimension, such as finite element spaces. Therefore we can think of u being an approximation of the *true* solution, although no distinct notation is used.

The interest of solving parametrized problems does not need much motivation. Just consider that Eq. (1.1), also referred as *high-fidelity* or *full* model in this context, governs the behaviour (mechanical, thermal...) of some part, and suppose that it wants to be optimized with respect to a parameter (material, geometry...). A first possible approach is to implement appropriate numerical methods, and perform as many direct solves of the full model as demanded by the optimization algorithm. Each request from the optimization algorithm is also called an *instance*. Since the optimization algorithm will generally need several iterations to attain a satisfactory result, this *brute-force* approach may be computationally too expensive for complex models, or when real-time constraints apply.

1.1.1 The Model Reduction and Low-rank approaches

MOR methods are designed to face this *multi-query*, or *multi-instance*, context in a smarter way. They are formulated with the main objective of reducing the computational complexity of some model, usually expressed in terms of partial differential equations (PDE). They are particularly useful when the model has to be solved repeatedly to get some quantities of interest (QoI), such as in design, optimization and control [40]. MOR methods are classically grouped as follows:

A posteriori methods. The reduced model is built after (hence, a posteriori) some inspection of the solution has been performed. That inspection usually consists in performing direct solves of the full model. We shall briefly present the main ideas behind two representative techniques of this category: Proper Orthogonal Decomposition (POD) [75, 90] and Reduced Basis Method (RBM) [50, 94].

A priori methods. They are based on some hypothesis that allow constructing the reduced model with no inspection of the solution (i.e. with no direct solve of the full model). The Proper Generalized Decomposition (PGD) [7] can be assigned to this category.

In next lines, however, we move slightly away from these classic definitions. To our discussion, it is more useful to regard MOR methods according to whether they define an underlying approximation of the solution or not. This can be illustrated more clearly by means of the parametrized problem presented previously: while the goal of some methods is limited to reduce the computational complexity of solving an instance, others wish to reduce the computational complexity associated to solve for every possible instance. The first case only needs to compute a reduced basis of small dimension. The second case needs of introducing an underlying approximation in the parametric domain.

A posteriori methods can be often identified with those that do not define an approximation in the parametric domain. POD, for instance, gathers some data from which a reduced basis is built, but the approximation itself is not explicitly provided. However, it is possible to think of other a posteriori formulations that do define an approximation in the parametric domain, such as Sparse Grids [31, 104] or Kernel Methods [134, 135]. Contrary to POD, they predefine the reduced basis and then they try to fit the coefficients to get an approximation in the solution in the parametric domain. The fitting needs of performing some direct solves of the full model, and this is why they are a posteriori methods. Finally, a priori methods always define an approximation in the parametric domain, but they do not predefine neither the reduced basis nor the coefficients. In fact, a priori methods require a reformulation of the problem in order to consider the parameter as a new coordinate [112]. The computational domain increases its dimensionality since the parameter is now at the same level than space, or time in transient models. Suitable techniques, such as *separated representations*, are needed to manage efficiently this growth of dimensionality.

MOR methods achieve the complexity reduction — and thus, the speed-up of simulations — in two different manners:

- By splitting the computational effort in *offline* and *online* phases. This is normally the case of those methods that do not define an approximation in the parametric domain: the approximation basis is computed in the offline phase, while the weights are computed at each instance by solving a small system of equations.

- By putting the whole computational effort in the offline phase. This category refers to those methods that define an approximation in the parametric domain, disregarding

they are a priori or a posteriori: at each instance, the solution is directly available because the whole work has been carried out offline.

Note that the classification based on the approximation is not banal at all because it is precisely the existence of such approximation what determines how the complexity reduction is expected to be attained. Besides, it shall be shown that it allows establishing a direct link between both MOR and Low-rank approximations communities through a priori methods.

Low-rank tensor approximations are originated from the wish of computing an approximation of the solution in the parametric domain. In such sense, they are absolutely comparable to a priori methods. The approximation is formulated in terms of tensor product spaces [64], which are introduced in §1.1.3. For the time being, let us simply assume that the solution of the parametrized problem defined in Eq. (1.1) is sought in a tensor product space. It will be shown that an algebraic problem whose solution is a two-dimensional tensor, is obtained by applying standard techniques such as the Galerkin method[1] [71]. It is well-known that any matrix —equivalently, two-dimensional tensor— can be decomposed into a rank representation by the Singular Value Decomposition. The rank representation can be seen as a compressed version of the matrix and hence the question is how to compute it implicitly, that is, directly in the compressed format. Low-rank tensor Approximations are an attempt to generalize rank representations to tensors in dimensions higher than two. Separated representations appear in this context as a manner to compress tensors and the Proper Generalized Decomposition as the algorithm to compute the tensor directly in compressed format.

From the above discussion, an evident parallelism between a priori MOR methods —PGD in particular— and Low-rank Tensor Approximations can be drawn. The following two interpretations of PGD yield:

- A method to compute Low-rank tensor approximations from the solution of an algebraic tensor problem. In such sense, PGD is an algebraic solver.

- An a priori MOR method to compute a reduced basis that solves some PDE. In such sense, PGD is a differential solver.

Although the interpretation as differential solver is probably the classical one, both are valid and from a practical point of view, equivalent. In §1.1.3.3 we will introduce the PGD as a differential solver that seeks the solution of the parametrized problem in a subset of the approximation space, previously introduced in §1.1.3.1. The author considers, however, that the interpretation as algebraic solver is particularly fruitful and thus it is privileged in subsequent sections. We refer to §1.4 for a complete exposition.

[1] Assume for simplicity that the bilinear form has good properties: bounded, continuous, symmetric and elliptic.

1.1.2 Methods to reduce the computational complexity

In this section, we shall review briefly two important methods among those that only seek reducing the computational complexity of solving an instance of the parametric problem.

1.1.2.1 Proper Orthogonal Decomposition

POD is possibly one of the pioneering and most popular MOR methods. The literature and the number of applications is extensive: heat transfer simulation [24], liquid crystalline polymers [10], Navier-Stokes flows [32], real-time simulation of non-linear tissues [102, 103] or chemistry [109], among others. Without going into much details, the main idea is to take advantage of the data already computed —previous instances of the full model— to reduce the computational complexity of subsequent solves of the parametrized equation. Therefore we introduce here the idea of sampling of the parametric domain as the collection of points —also called *snapshots*— at which the full model has been previously solved. POD extracts a reduced basis (also called singular vectors, principal components) from that data. In other words: POD seeks a subspace $S_{pod} := \text{span}\{\psi^{1\leq i\leq M}\} \subset V_s$ on which data can be still well represented, that is:

$$u(\mu) = \sum_{i=1}^{M} \alpha_i(\mu)\, \psi^i. \tag{1.2}$$

Eq. (1.2) constitutes indeed a first definition of what we understand by separated representations, since the coefficients α_i depend on the parameter and the basis functions on the space coordinates. We expect that

$$\dim(S_{pod}) = M \ll \dim(V_s).$$

The *reduced* problem is usually formed by Galerkin projection of the full one onto the computed subspace. In practice, the dimension of the subspace can be chosen according to predefined truncation error, that can be easily computed, see §1.3.1. However, it is worth to remark that the truncation error expresses the amount of data that cannot be represented onto the computed subspace, and it should not be confused with the error of a reduced solution with respect to a full one for a new parameter value. This latter error cannot be assessed so easily without coming back to the full model [105].

It is worth to mention that there exist some variants that suggest interpolating between reduced solutions [12, 68]. That is, given S_{pod} and the coefficients $\alpha_i(\mu_1)$ and $\alpha_i(\mu_2)$ associated to parameter value μ_1 and μ_2, respectively, the coefficients associated to some $\mu \in [\mu_1, \mu_2]$ could be found by interpolating between $\alpha_i(\mu_1)$ and $\alpha_i(\mu_2)$. This constitutes a manner of constructing an a posteriori approximation in the parametric domain without truly solving the parametrized problem. This approach can only be used if the solution varies smoothly in the parametric domain.

1.1.2.2 Reduced Basis Method

The Reduced Basis Method (RBM) operates quite similarly to POD but defines a strategy to sample the parametric domain wisely [93]. Although different RBM approaches are possible, the *greedy* one is probably the most attractive. It consists in expanding the reduced basis by adding one term at a time. Hence, suppose that the reduced basis already contains M functions, i.e. $\dim(S_{rbm}) = M$. If the error indicator computed at the so-called *train points* is not satisfied, the basis is expanded by adding one term. A new snapshot is computed where the error indicator is maximized. This snapshot is usually orthogonalized with respect to the elements of the reduced basis using a Gram-Schmidt procedure and then incorporated to it. This is the offline part of the work. Once it is done, if we wish to compute the solution for a new parameter value, a reduced problem of size $\dim(S_{rbm})$ must be solved. The reduced problem is generally formed by Galerkin projection as well. The latter constitutes the online part of the work. The computational cost is thus reduced, as it was in POD, but in this case the outputs of the reduced system are guaranteed thanks to the error estimate [111].

Although RBM can be considered an improvement of POD, it does not provide an approximation of the solution in the parametric space; it provides a reduced basis able to represent the solution for any parameter value in the parametric domain with a controlled error level. This results in a very practical, non-intrusive approach, at least for single-parameter problems. However, neither brute-force, POD nor RBM approaches permit computing an approximation of the solution in the parametric domain.

1.1.3 Methods based on defining an approximation

In this section, tensor product spaces are introduced as the foundations of both a priori MOR methods —PGD in particular— and Low-rank tensor approximations. Tensor Product Spaces are nothing but an appropriate framework to formalize the approximation in the parametric domain. Furthermore, the separated representations and tensor product spaces are closely related. PGD interpreted as a differential solver will be presented afterwards using the Tensor Product Spaces framework.

Since introducing Low-rank tensor approximations here would take much place, this task is left to a subsequent Section, see §1.3.

1.1.3.1 Introducing tensor product spaces

Computing an approximation of the solution in the parametric domain requires reformulating the variational problem defined in Eq. (1.1). We first introduce an approximation space on the parametric domain, denoted by V_μ, which may be a finite element space or the space of polynomials of degree $\dim(V_\mu) - 1$, for instance. The new problem is written as follows:

find $u \in V := V_s \otimes V_\mu$, a tensor product space, such that

$$a(u, w) = l(w), \tag{1.3}$$

for every $w \in V$. The parametric dependence is omitted in Eq. (1.3) because in this case we are defining an approximation that covers also the parametric domain. Or in other words, the parameter is regarded as a new *coordinate* just as space and time are. It is not the aim of this work to provide an exhaustive mathematical introduction to tensor product spaces because it is outside the scope of this work. If $V_s := \text{span}\{v_s^{1 \leq i \leq N}\}$ and $V_\mu := \text{span}\{v_\mu^{1 \leq i \leq M}\}$, the tensor product space of these spaces is defined as

$$V := \text{span}\{v_s^i \, v_\mu^j, \text{ for } 1 \leq i \leq N \text{ and } 1 \leq j \leq M\}.$$

An element of V writes as follows:

$$u = \sum_{i=1}^{N} \sum_{j=1}^{M} u_{ij} \, v_s^i v_\mu^j, \tag{1.4}$$

where u_{ij} defines the entries of a two-dimensional tensor, denoted by $\mathcal{U} \in \mathbb{K}^{N \times M}$.

Example 1.1 (*Finite element spaces built from tensor product spaces*). To keep in mind a practical idea of these concepts, let us highlight that two and three-dimensional finite element spaces [19, 71] can be built from the tensor product space of one-dimensional finite element spaces. For instance, the bilinear quadrangular element of 4 nodes is introduced from

$$P_1(\xi) = \text{span}\{1, \xi\} \quad \text{and} \quad P_1(\eta) = \text{span}\{1, \eta\},$$

whose tensor product space is:

$$P_{1,1}(\xi, \eta) = \text{span}\{1, \xi, \eta, \xi\eta\}.$$

In the reference configuration with $\xi, \eta \in [-1, 1] \times [-1, 1]$, the shape functions are obtained with the following four possible coefficient combinations $1/4\,(1, \alpha_1, \alpha_2, \alpha_3)$, being $\alpha_1, \alpha_2 = \pm 1$ and $\alpha_3 = \text{sign}(\alpha_1)\,\text{sign}(\alpha_2)$. Of course, there exist many finite element formulations in which some elements of the basis are subtracted on purpose, for many different reasons that are not being discussed here. The solution is sought then in a subspace of the tensor product space. The only difference between Eq. (1.4) and finite element formulations is that a linear indexation is usually preferred,

$$u = \sum_{i=1}^{NM} u_i \, N^i,$$

and thus u_i defines a vector that is in fact the vectorization of the tensor already defined, i.e. $\boldsymbol{u} = \text{vect}(\mathcal{U}) \in \mathbb{K}^{NM \times 1}$. Additionally, basis functions are usually identified with N^i, the shape functions.

Eq. (1.4) defines a separated representation in a general sense: the solution is written as a sum of products of functions of each individual coordinate. In this framework, it is possible to compute *truly* an approximation of the solution in the parametric space.

1.1.3.2 Galerkin formulation and structure of the problem

In next lines an algebraic problem with tensor structure is obtained by applying the Galerkin method. First note that Eq. (1.3) is equivalent to solve [116]:

$$\langle A(u), w\rangle = \langle b, w\rangle, \tag{1.5}$$

where $\langle\,\cdot\,,\,\cdot\,\rangle$ denotes an inner product on V, A is a linear operator defined by $A : V \to V$ and b is an element of V. Using a Galerkin approach, that is, w is approximated in the same tensor product space as the solution, the left and and right-hand sides of Eq. (1.5) are rewritten as follows:

$$\begin{aligned}\langle A(u), w\rangle &= \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{N}\sum_{l=1}^{M} u_{ij}\langle A(v_s^i v_\mu^j), v_s^k v_\mu^l\rangle w_{kl},\\ \langle b, w\rangle &= \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{N}\sum_{l=1}^{M} b_{ij}\langle v_s^i v_\mu^j, v_s^k v_\mu^l\rangle w_{kl}.\end{aligned} \tag{1.6}$$

Eq. (1.6) is conceptually very simple, although quite tedious to manipulate. Therefore, it is worth to spend some lines to illustrate it with an example.

Example 1.2 (*Structure of the problem with a laplacian operator*). Consider that A is the laplacian operator and the parameter μ is the diffusivity coefficient. Therefore $A(\,\cdot\,) = \mu\,\Delta(\,\cdot\,)$. Then, for $1 \le i, k \le N$ and $1 \le j, l \le M$ we have

$$\begin{aligned}\langle A(v_s^i v_\mu^j), v_s^k v_\mu^l\rangle &= \langle \Delta v_s^i, v_s^k\rangle_s \langle \mu v_\mu^j, v_\mu^l\rangle_\mu \quad &\rightsquigarrow \quad \boldsymbol{K}_s \otimes \boldsymbol{D}_\mu,\\ \langle v_s^i v_\mu^j, v_s^k v_\mu^l\rangle &= \langle v_s^i, v_s^k\rangle_s \langle v_\mu^j, v_\mu^l\rangle_\mu \quad &\rightsquigarrow \quad \boldsymbol{M}_s \otimes \boldsymbol{M}_\mu,\end{aligned} \tag{1.7}$$

where $\boldsymbol{K}_s$, $\boldsymbol{M}_s$ are the diffusion (laplacian-like) matrix and the mass matrix in the physical domain, respectively. $\boldsymbol{D}_\mu$, $\boldsymbol{M}_\mu$ are a sort of weighted mass matrix and the mass matrix itself in the parametric domain, respectively. We denote by "$\otimes$" the Kronecker product.

Notice that it is possible to write Eq. (1.7) thanks to the following facts:

- The operator A is linear and admits a separated representation, also called an *affine decomposition* is some contexts. In the case of the laplacian operator it is trivial. In general, for linear problems this requirement is not difficult to satisfy.
- The inner product in tensor product spaces has the following property: for $u = u_s u_\mu$ and $v = v_s v_\mu$, we have $\langle u, v\rangle = \langle u_s, v_s\rangle_s \langle u_\mu, v_\mu\rangle_\mu$, where $\langle\,\cdot\,,\,\cdot\,\rangle_s$ and $\langle\,\cdot\,,\,\cdot\,\rangle_\mu$ are inner products on V_s and V_μ, respectively. The norm inherits the same property.

Introducing the discrete matrix operators that we have just presented, Eq. (1.5) can be written now in algebraic form as follows:

$$\boldsymbol{w}^H \boldsymbol{A}\boldsymbol{u} = \boldsymbol{w}^H \boldsymbol{f}, \tag{1.8}$$

where $\boldsymbol{w}$ and $\boldsymbol{u}$ are vectorizations of their corresponding tensors and $\boldsymbol{w}^H$ denotes the conjugate-transpose of $\boldsymbol{w}$. In example with the laplacian operator, we have $\boldsymbol{A} = \boldsymbol{K}_s \otimes \boldsymbol{D}_\mu$ and $\boldsymbol{f} = (\boldsymbol{M}_s \otimes \boldsymbol{M}_\mu)\boldsymbol{b}$, where $\boldsymbol{b}$ is a vectorization of its corresponding tensor.

Let us emphasize that Eq. (1.8) has been obtained in the framework of tensor product spaces using completely standard methods, exactly the same one would use to formulate a finite element problem. However, computing an approximation such as the one defined in this section has a price, which is increasing the dimensionality. Where we only had the physical dimension, i.e. the physical space, the parameter has been added as an additional coordinate. The size of the algebraic system to be solved is considerably increased. It will be shown in §1.2 that when formulating multiparametrized or multidimensional problems in tensor product spaces the computational complexity scales exponentially with the number of dimensions. In consequence, obtaining an explicit solution becomes quickly unfeasible. Appropriate techniques like separated representations are therefore needed.

1.1.3.3 Proper Generalized Decomposition as a differential solver

In this Section, we take advantage of tensor product spaces already presented in §1.1.3.1 to introduce the Proper Generalized Decomposition [7] as a differential solver in the context of a priori MOR methods. In this way, the idea of PGD being at the interface between MOR and Low-rank Tensor Approximations is clearly supported. Furthermore, we refer to §1.4.2 for PGD regarded as an algebraic solver in the context of Low-rank tensor approximations.

PGD is here regarded as a solver that builds a reduced basis progressively by splitting the parametrized problem into two lower-dimensional problems: one defined in the physical space and another one defined on the parametric domain. If those lower-dimensional problems had analytical solution, one would not need to introduce approximation spaces of any kind. That is why we say that PGD can be interpreted as a model-reduction differential solver. Of course, analytical solutions are rare and in practice the low-dimensional problems will be solved by introducing classic approximation spaces such as finite elements.

Let us consider functional spaces $\mathcal{V}_s$ and $\mathcal{V}_\mu$ that fulfill the requirements of Eq. (1.3), including the boundary conditions[2], and let $\mathcal{V} := \mathcal{V}_s \otimes \mathcal{V}_\mu$ be the tensor product space built from them. Consider R pairs of functions $(u_s^r, u_\mu^r) \in \mathcal{V}_s \times \mathcal{V}_\mu$ such that the solution of Eq. (1.3) can be well approximated as follows:

$$u \approx \sum_{r=1}^{R} u_s^r u_\mu^r. \tag{1.9}$$

Eq. (1.9) is called a separated representation of order R, because it is a sum of function products of space and parameter. The objective of PGD is to compute the function pairs

[2]For the Laplacian problem considered in §1.1.3.2, $\mathcal{V}_\mu$ can be identified with the space of square-integrable functions $L^2(I_\mu)$ and $\mathcal{V}_s$ can be identified with the space of square integrable functions whose first (weak) derivative is also square integrable, $H^1(I_s)$. Here I_s denotes the physical domain.

(u_s^r, u_μ^r), also called space and parameter *modes*, respectively. Observe that this can be recast into the notation already introduced in §1.1.3.1 by considering the following finite dimensional approximation spaces defined from the modes:

$$V_s := \text{span}\left\{v_s^r = \frac{u_s^r}{\|u_s^r\|},\ 1 \le r \le R\right\} \quad \text{and} \quad V_\mu := \text{span}\left\{v_\mu^r = \frac{u_\mu^r}{\|u_\mu^r\|},\ 1 \le r \le R\right\},$$

both of dimension R. Think of v_s^r and v_μ^r being globally supported functions. A tensor product space can be built from these spaces as $V := V_s \otimes V_\mu \subset \mathcal{V}$. Let us introduce a subset of that tensor product space:

$$S_R := \left\{ v \in V :\ v = \sum_{r=1}^{R} u_r\, v_s^r v_\mu^r,\ \text{with}\ v_s^r \in V_s, v_\mu^r \in V_\mu\ \text{and}\ u_r \in \mathbb{K} \right\}.$$

We denote by $u^{(R)}$ an element of S_R, also called a rank-R separated representation by reasons that will be clearer in §1.3. From Eq. (1.9), the approximation $u \approx u^{(R)}$ belongs to that subset.

PGD allows building S_R progressively: $S_1, S_2, \ldots$, each one of them defined from $S_R = S_{R-1} + S_1$, for $R \ge 2$. Therefore, the successive approximation spaces are nested, i.e. $S_{R-1} \subset S_R$. This is made by seeking a pair $(u_s, u_\mu) \in \mathcal{V}_s \times \mathcal{V}_\mu$. When it is available, V_s and V_μ are updated by normalizing u_s and u_μ, respectively. The new approximation will be defined from

$$u^{(R+1)} = u^{(R)} + u_s u_\mu \quad \Leftrightarrow \quad u^{(R+1)} = \sum_{r=1}^{R+1} u_r\, v_s^r v_\mu^r.$$

The algorithm to compute those pairs will be detailed later. At this point, two questions arise:

- First is about the convergence of $u^{(R)}$ towards u as $R \to +\infty$. For an important class of problems, convergence can be proven [52]. In many other cases, even though convergence is not proven, the method has been successfully applied. See §1.5 for a short review on the applications.

- Second is about the optimality of such approximation. That is, assuming that a PGD decomposition $u^{(R)}$ has been computed, is it the most compact one? Although it can be proven that PGD generalizes POD in some cases [52, 106], and therefore, it is optimal in the same sense as POD, in general the optimality problem is ill-posed and there is not an answer to it [47]. See §1.3.3.1 for more details.

The question on convergence yields a parallelism with classic numerical methods. In the same spirit as h-adaptivity is defined in the context of mesh-based approximations, or p-adaptivity for high-order methods, a sort of rank adaptivity —r-adaptivity— may require increasing the approximation rank to improve the solution.

The PGD is adaptive in such sense. In order to present the PGD algorithm, let us assume that we have already built S_R, i.e. $u^{(R)}$ is known, and we want to compute a couple $(u_s, u_\mu) \in \mathcal{V}_s \times \mathcal{V}_\mu$ such that:

$$a(u_s u_\mu, w) = r(u^{(R)}, w), \tag{1.10}$$

for every $w \in \mathcal{V}$. We denote by $r(\,\cdot\,,\,\cdot\,)$ the residual, defined as

$$r(u^{(R)}, w) := l(w) - a(u^{(R)}, w).$$

Observe that computing both u_s and u_μ is a nonlinear problem. Let us assume by now that the problem to be solved is elliptic. Then, Eq. (1.10) can be turned into an equivalent nonlinear optimization problem:

$$\min_{u_s, u_\mu} \mathcal{J}(u_s, u_\mu) := \frac{1}{2} a(u_s u_\mu, u_s u_\mu) - r(u^{(R)}, u_s u_\mu).$$

The stationarity conditions of the functional can be found by means of calculus of variations. Consider the following arbitrary variations: $u_s + \xi w_s$ and $u_\mu + \eta w_\mu$. Substituting and taking derivatives with respect to ξ and η:

$$a(u_s u_\mu, w_s u_\mu) = r(u^{(R)}, w_s u_\mu), \quad \forall w_s \in \mathcal{V}_s, \tag{1.11a}$$

$$a(u_s u_\mu, u_s w_\mu) = r(u^{(R)}, u_s w_\mu), \quad \forall w_\mu \in \mathcal{V}_\mu. \tag{1.11b}$$

The stationarity conditions can be expressed in a single equation as follows: find $(u_s, u_\mu) \in \mathcal{V}_s \times \mathcal{V}_\mu$ such that

$$a(u_s u_\mu, w_s u_\mu + u_s w_\mu) = r(u^{(R)}, w_s u_\mu + u_s w_\mu), \quad \forall (w_s, w_\mu) \in \mathcal{V}_s \times \mathcal{V}_\mu. \tag{1.12}$$

Eq. (1.12) can be interpreted as a Galerkin formulation which imposes the cancellation (i.e. orthogonality) of the residual simultaneously with respect to $\mathcal{V}_s \otimes \{u_\mu\}$ and $\{u_s\} \otimes \mathcal{V}_\mu$ [106]. Depending on the type of problem, alternative formulations may be more convenient, see §1.4.3.

Eq. (1.11) suggests applying an fixed-point algorithm to solve the nonlinear optimization problem. The fixed-point is as follows:

1. Assume u_μ is known, then update u_s from Eq. (1.11a).
2. From u_s just computed, update u_μ by solving Eq. (1.11b).
3. Go back to the first step.

Let us introduce the linear operator $A : \mathcal{V} \to \mathcal{V}$ such that $a(u, w) = \langle A(u), w \rangle$, being $\langle\,\cdot\,,\,\cdot\,\rangle$ the inner product on $\mathcal{V}$. We also introduce $b \in \mathcal{V}$ such that $l(w) = \langle b, w \rangle$. The following two crucial hypothesis are needed:

Hypothesis 1.1 (*Affine decomposition of the operator*). Given $u_s u_\mu \in \mathcal{V}$, the operator admits an affine decomposition of order T:

$$A(u_s u_\mu) = \sum_{t=1}^{T} A_s^t(u_s) A_\mu^t(u_\mu),$$

with $A_s^t : \mathcal{V}_s \to \mathcal{V}_s$ and $A_\mu^t : \mathcal{V}_\mu \to \mathcal{V}_\mu$, for $1 \leq t \leq T$.

Hypothesis 1.2 (*Separated representation of the right-hand side*). The right-hand side admits a separated representation of order J:

$$b = \sum_{j=1}^{J} b_s^j \, b_\mu^j,$$

with $b_s^j \in \mathcal{V}_s$ and $b_\mu^j \in \mathcal{V}_\mu$, for $1 \leq j \leq J$.

These two hypothesis motivate most of works concerned with PGD, and in particular, this thesis. Considering the last two hypothesis, the left and right-hand sides of Eq. (1.10) can be rewritten as

$$\begin{aligned} a(u_s u_\mu, w) &= \sum_{t=1}^{T} \langle A_s^t(u_s) A_\mu^t(u_\mu), w \rangle \quad \text{and} \\ r(u^{(R)}, w) &= \sum_{j=1}^{J} \langle b_s^j b_\mu^j, w \rangle - \sum_{t=1}^{T} \sum_{r=1}^{R} \langle A_s^t(u_s^r) A_\mu^t(u_\mu^r), w \rangle, \end{aligned} \tag{1.13}$$

respectively, where w is either $w = w_s u_\mu$, which corresponds to Eq. (1.11a), or $w = u_s w_\mu$, which corresponds to Eq. (1.11b). Recalling the separation property of the inner product in tensor product spaces[3] we get the following problem: find $(u_s, u_\mu) \in \mathcal{V}_s \times \mathcal{V}_\mu$ such that

$$\langle \tilde{A}_s(u_s), w_s \rangle_s = \langle \tilde{r}_s, w_s \rangle_s \tag{1.14a}$$

$$\langle \tilde{A}_\mu(u_\mu), w_\mu \rangle_\mu = \langle \tilde{r}_\mu, w_\mu \rangle_\mu, \tag{1.14b}$$

for every $(w_s, w_\mu) \in \mathcal{V}_s \times \mathcal{V}_\mu$. The following notation has been introduced:

$$\begin{aligned} \tilde{A}_s(u_s) &= \sum_{t=1}^{T} \alpha_\mu^t \, A_s^t(u_s), \qquad \tilde{r}_s = \sum_{j=1}^{J} \beta_\mu^j \, b_s^j - \sum_{t=1}^{T} \sum_{r=1}^{R} \gamma_\mu^{tr} \, A_s^t(u_s^r), \\ \tilde{A}_\mu(u_\mu) &= \sum_{t=1}^{T} \alpha_s^t \, A_\mu^t(u_\mu), \qquad \tilde{r}_\mu = \sum_{j=1}^{J} \beta_s^j \, b_\mu^j - \sum_{t=1}^{T} \sum_{r=1}^{R} \gamma_s^{tr} \, A_\mu^t(u_\mu^r), \end{aligned}$$

where the following scalars have been used:

$$\alpha_\tau^t = \langle A_\tau^t(u_\tau), u_\tau \rangle_\tau, \quad \beta_\tau^j = \langle b_\tau^j, u_\tau \rangle_\tau, \quad \text{and} \quad \gamma_\tau^{tr} = \langle A_\tau^t(u_\tau^r), u_\tau \rangle_\tau, \tag{1.15}$$

[3] $\langle u_s u_\mu, v_s v_\mu \rangle = \langle u_s, v_s \rangle_s \langle u_\mu, v_\mu \rangle_\mu$, with $\langle \cdot, \cdot \rangle_s$ and $\langle \cdot, \cdot \rangle_\mu$ inner products in $\mathcal{V}_s$ and $\mathcal{V}_\mu$, respectively.

with $\tau \equiv s, \mu$. Observe that Eq. (1.14) involves the solution of two lower-dimensional problems: the first of them, Eq. (1.14a), is defined on the space domain while the second problem, Eq. (1.14b), is defined on the parametric domain. Both problems are coupled by some scalars defined in Eq. (1.15).

Example 1.3 (*Structure of lower-dimensional problems with a laplacian operator*). In this example we illustrate Eq. (1.14) when a laplacian operator $A(\,\cdot\,) = \mu\Delta(\,\cdot\,)$ is of interest. Additionally, we assume that $b_s = f$ and $b_\mu = 1$. First we note that given v, $A_s(v) = \Delta v$ and $A_\mu(v) = \mu v$. Therefore:

$$\alpha_\mu \langle \Delta u_s, w_s \rangle_s = \beta_\mu \langle f, w_s \rangle_s - \sum_{r=1}^{R} \gamma_\mu^r \langle \Delta u_s^r, w_s \rangle_s, \quad \forall w_s \in \mathcal{V}_s$$

$$\alpha_s \langle \mu u_\mu, w_\mu \rangle_\mu = \beta_s \langle 1, w_\mu \rangle_\mu - \sum_{r=1}^{R} \gamma_s^r \langle \mu u_\mu^r, w_\mu \rangle_\mu \quad \forall w_\mu \in \mathcal{V}_\mu,$$

which may be rewritten in strong form as follows:

$$\alpha_\mu \, \Delta u_s = \beta_\mu \, f - \sum_{r=1}^{R} \gamma_\mu^r \, \Delta u_s^r$$

$$\alpha_s \, \mu u_\mu = \beta_s - \sum_{r=1}^{R} \gamma_s^r \, \mu u_\mu^r.$$

A laplacian equation has been obtained to compute the space function u_s, while the parameter function u_μ can be computed by solving an algebraic equation whose cost is negligible. The laplacian equation can be solved applying standard numerical techniques: finite differences, finite elements, or others.

A fruitful observation can be made from the previous example. PGD defines two approximation levels:

- First approximation level is that of building approximation spaces V_s and V_μ such that the solution is sought in a subset of $\mathcal{V}$.
- Second approximation level may not be needed if the exact solution of the lower-dimensional problems is known. Otherwise, u_s and u_μ can only be computed by introducing additional approximation spaces such as finite elements, for instance. Discrete operators (i.e. matrices) associated to this approximation level are exactly the same obtained in §1.1.3.2: $\boldsymbol{K}_s$, $\boldsymbol{M}_s$, $\boldsymbol{D}_\mu$ and $\boldsymbol{M}_\mu$.

With the last observation, we reach the parallelism between Low-rank tensor approximations and a priori MOR through PGD: both approaches yield the same problem although the arguments are different.

The fixed-point algorithm for solving Eq. (1.14) is said to be converged either when a maximum number of iterations is attained or a stationary point is reached. We evaluate the

following criterion:

$$\|(v_s v_\mu)_k - (v_s v_\mu)_{k-1}\| < \varepsilon_1,$$

where $\|\cdot\| := \langle \cdot, \cdot \rangle^{1/2}$ denotes the norm on $\mathcal{V}$. It states that the fixed-point iterations are stopped if the difference between two successive iterations k and $k-1$ is smaller than ε_1. Recall that v_s and v_μ are the normalized version of u_s and u_μ, respectively. The separated representation is said to be converged if, after computing a reduced basis of size R, the residual compared to the right-hand side has been reduced by some factor ε_2:

$$r(u^{(R)}, u^{(R)}) < \varepsilon_2 \, l(u^{(R)}).$$

1.2 Multiparametrized and multidimensional problems

Multiparametrized problems are the direct extension of parametrized problems already introduced in §1.1: rather than models with a single parameter, real problems found in design, optimization and control in many areas of engineering depend on many parameters [40, 63].

Multidimensional models, most commonly found in physics, are of very different nature. They describe the kinetics of complex materials [26], competition in biological systems [5], social dynamics and economics systems, among many others; see [21, 125]. These models are conceived from their origin as multidimensional. Lagrangian mechanics, for instance, is a classical example of multidimensional model which reformulates Newtonian mechanics into a new space of *generalized coordinates*, one of them *per* degree of freedom.

Although both multiparametrized and multidimensional models are defined in higher-dimensional domains, MOR methods cannot always be applied in the same manner. MOR methods whose only aim is to reduce the computational complexity, presented in §1.1.2, can be applied to multiparametrized problems but not to multidimensional problems in the manner they are defined here. Instead, tensor product spaces and a priori MOR methods, presented in §1.1.3, provide an unified framework to treat both multiparametrized and multidimensional problems [112].

1.2.1 Complexity reduction of multiparametrized problems

In this section, we shall only consider the complexity reduction of multiparametrized problems by Reduced Basis Method, although POD could also be applied [63]. The RBM procedure can be extended for higher dimensions with almost no modifications. The reduced basis is constructed by sampling the parametric space $I_\mu \subset \mathbb{R}^D$, now higher-dimensional, for some parameter combinations chosen with an appropriate error estimator. Once the reduced basis is large enough to represent the solution according to a prescribed error level,

the solution is expressed in terms of such basis as:

$$u(\boldsymbol{\mu}) = \sum_{i=1}^{M} \alpha_i(\boldsymbol{\mu})\, \psi^i, \tag{1.16}$$

where ψ^i are the basis functions, α_i are the coefficients to be computed online and $\boldsymbol{\mu} \in I_\mu$. Note that:

- The cardinal of the set of train points, defined in §1.1.2, grows exponentially with the number of dimensions. This means that the number of direct —full— solves grows accordingly. The offline cost to be paid to build the reduced basis may become rapidly unaffordable.
- The dimension of the reduced basis might grow also rapidly as it must capture the variations of the solution over the whole multiparametric domain. Recall that the dimension of the reduced basis is directly related to the computational cost of the online part in the RBM context.

Despite these potential weaknesses, RBM has been successfully applied to reduce the computational complexity of more complex problems than any other MOR technique, including Navier-Stokes equations [120, 130]. However, multidimensional models (in the sense they are described in this section) cannot be treated easily by RBM nor, in most of cases, by any a posteriori MOR method. This is because they are designed to reduce the computational complexity, not to solve the multidimensional model. Recall that solving a multidimensional model means computing an approximation of the solution on the higher-dimensional domain in which it is defined.

1.2.2 A unified framework for multiparametrized and multidimensional problems

Parameters are considered as new coordinates of the model, and thus, multiparametrized models become indistinguishable from multidimensional models. From now on we shall refer to multidimensional models disregarding they are multiparametrized or not.

We proceed merely as in §1.1.3. Let us consider an open bounded domain $I \subset \mathbb{R}^D$ formed from the cartesian product of lower dimensional domains (one-dimensional without loss of generality), that is $I := I_1 \times \cdots \times I_D$. In order to compute an approximation of the solution, let us introduce approximation spaces of finite dimension defined on each one-dimensional domain, $V_d(I_d) := \text{span}\{v_d^{1 \leq i_d \leq N_d}\}$, for $1 \leq d \leq D$. We denote by N_d their dimension. The tensor product space is built from the approximation spaces as $V := \bigotimes_{d=1}^{D} V_d$. Any $u \in V$ writes as:

$$u = \sum_{i_1=1}^{N_1} \cdots \sum_{i_D=1}^{N_D} u_{i_1 \cdots i_D}\, v_1^{i_1} \otimes \cdots \otimes v_D^{i_D}, \tag{1.17}$$

where the entries $u_{i_1 \cdots i_D}$ define a tensor $\mathcal{U} \in \mathbb{K}^{\mathcal{N}}$, being $\mathcal{N} := N_1 \times \cdots \times N_D$. Recall that the inner product on such space enjoys the following separation property:

$$\text{Given} \quad u = \bigotimes_{d=1}^{D} u_d \quad \text{and} \quad v = \bigotimes_{d=1}^{D} v_d, \quad \langle u, v \rangle = \prod_{d=1}^{D} \langle u_d, v_d \rangle_d, \tag{1.18}$$

which is of course inherited by the norm. Eq. (1.5) defines the problem to be solved and, since it remains unchanged for the multidimensional case, we do not rewrite it. Using a Galerkin approach and the separability of the operator as we did in §1.1.3.2, the problem can be turned into an algebraic, formally equivalent to Eq. (1.8), but defined in dimension D:

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}. \tag{1.19}$$

If the operator can be written in separated representation using T terms, the algebraic operator can be written as follows:

$$\boldsymbol{A} = \sum_{t=1}^{T} \otimes_{d=1}^{D} \boldsymbol{A}_d^t, \tag{1.20}$$

where matrices $\boldsymbol{A}_d^t$ are operators on the d-th one-dimensional domain.

Example 1.4 (*D-dimensional laplacian operator*). The laplacian operator in D-dimension is trivially written in separated form with D terms:

$$\Delta := \sum_{j=1}^{D} \frac{\partial^2}{\partial x_j^2}.$$

By considering for instance finite element spaces in each dimension, the matrix operator defined in Eq. (1.20) takes the following form:

$$\boldsymbol{A}_d^j = \begin{cases} \boldsymbol{M}_d & \text{if} \quad d \neq j \\ \boldsymbol{K}_d & \text{if} \quad d = j \end{cases},$$

where $\boldsymbol{M}_d$ and $\boldsymbol{K}_d$ denote the mass and diffusion (laplacian-like) matrices in dimension d, respectively.

In §1.1.3.2, it was observed that the computational complexity of a direct solve of Eq. (1.8) —which only involved the space and one parameter— was considerably increased. In general, for a D-dimensional problem such as Eq. (1.19), the computational complexity becomes rapidly unaffordable.

PGD could be extended to multidimensional problems with little conceptual effort. However, we prefer to present PGD for multidimensional problems in the context of implicit Low-rank tensor approximations in §1.4.2, simply because it has been shown in §1.1.3.3 that it is equivalent to see PGD as a differential or as an algebraic solver.

1.2.3 The curse of dimensionality

Up to this point we have not paid much attention the computational aspects associated to computing an approximation of the solution in higher-dimensional domains. Eq. (1.17) defines a D-dimensional tensor. Therefore, a system of equations with $|\mathcal{N}| = \prod_{d=1}^{D} N_d \sim N^D$[4] unknowns must be solved to compute an approximation of the solution. This complexity is comparable to the brute-force approach and thus, unaffordable. This is known as the *curse of dimensionality* [20]. Since the number of unknowns scales exponentially with the number of dimensions, reducing the computational complexity becomes compulsory.

Still, let us assume that we are able to solve multidimensional models at an affordable computational cost. Several observations would follow: first is that the higher dimensional domain could be explored almost *for free*; we only need to *particularize* the solution at the desired coordinates or parameter values. Second is that the multidimensional model has to be solved only once and then stored and used; this defines the concept of *Computational Vademecums* [40]. Third is that, as a consequence, many multi-query or real-time applications would benefit from a great speed-up.

In summary, one would like to keep the aforementioned benefits but reducing the computational complexity at the same time. Two crucial questions must be answered to make it possible:

1. Assume that we are able to compute the tensor defined in Eq. (1.17). An explicit storage of such tensor is obviously expensive. As an example, let $N = 10$ be the size of the approximation basis in every dimension and $D = 10$, the number of dimensions. An explicit storage of such tensor in double-precision data type requires roughly 75 GBytes. Therefore, is it possible to store it more efficiently, using a sort of compression? This is directly linked to the study of tensor formats and low-rank tensor representations in §1.3.

2. Is there an algorithm capable of computing such tensor directly in the compressed format? And more specifically, does it operate at a computational cost which does not scale exponentially with the number of dimensions? This question —partially answered in §1.1.3.3 for the space-parameter case— introduces the discussion on the PGD method as an algebraic solver in §1.4.2.

Separated representations provide an answer to these questions, although they are not the only alternative. Separated representations have been partially introduced in this section: Eq. (1.2) in the context of POD for single parameter problem; Eq. (1.16) in the context of RBM for multiparametrized problems; or Eq. (1.4) in the context of tensor product spaces, as the most general definition of separated representation. In next sections, separated representations are regarded in the context of Low-rank tensor approximations.

[4] $|\cdot|$ denotes the cardinal of a set.

1.3 Tensor representations and low-rank approximations

Low-rank tensor approximations constitute a family of increasingly popular techniques due to its generality and success in many fields in science and engineering; see [62] for an extensive list of applications. Although in this work tensors are mainly regarded as elements in a tensor product space, in general there is no need of this functional framework to study their representation. One may simply introduce tensors in the sense of multidimensional arrays coming from the discretization of some multivariate function. Therefore, let us consider some tensor in higher dimension and assume that we have enough computational ressources to manipulate it. It is to be highlighted that, in practice, this is not the case: tensors are the solution of some equation and therefore we only have an implicit representation.

There are two questions that we would like to address in this section:

- The first one has already been formulated in §1.2.3 and concerns the existence a "compressed" format for tensors. The answer is of course positive, and three different formats are introduced in §1.3.3.
- The second one concerns the concept of optimality of such compressed format. It must be pointed out that in computational mechanics we are not truly interested in computing the best (roughly, the most compressed) representation of a tensor, which might require many computational ressources. Instead, we prefer computing a less compressed tensor but at a lower computational cost. If the compressed version is still too heavy for the targeted application, we may want to compress it further. For this reason, we shall provide a brief compilation of results on optimality in §1.3.4.

Before considering tensor of any dimension, let us introduce briefly the paradigmatic two-dimensional case.

1.3.1 Rank representation of matrices: the Singular Value Decomposition

Just as *Singular Value Decomposition* (SVD) allows finding a compressed representation of a matrix, we would like to know if the same is generalizable for higher dimensions. Consider a matrix $\boldsymbol{U} \in \mathbb{K}^{N_1 \times N_2}$. There exist a factorization such that

$$\boldsymbol{U} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{S}^H,$$
$$\boldsymbol{L} \in \mathbb{K}^{N_1 \times N_1}, \quad \boldsymbol{D} \in \mathbb{R}_+^{N_1 \times N_2} \quad \text{and} \quad \boldsymbol{S} \in \mathbb{K}^{N_2 \times N_2}.$$

where $\boldsymbol{L}$ and $\boldsymbol{S}$ contain the left and right *singular vectors* by columns, respectively. They form orthogonal basis. Matrix $\boldsymbol{D}$ is pseudo-diagonal (see Fig. 1.1) with non-negative entries $(\sigma_1, \ldots, \sigma_M)$ ordered decreasingly. Those entries are called *singular values*. Since the row and column ranks of a matrix coincide, which is not the case in higher dimensions, the rank

$$\begin{bmatrix} \star & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \star \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} \star & 0 & 0 & 0 & \cdots & 0 \\ 0 & \ddots & 0 & 0 & \cdots & 0 \\ 0 & 0 & \star & 0 & \cdots & 0 \end{bmatrix}$$

Figure 1.1: Pseudo-diagonal matrices: $N_1 > N_2$ (left) and $N_1 < N_2$ (right)

R is such that $R \leq M = \min(N_1, N_2)$. Equivalently, there are are $R \leq M$ non-zero singular values. Introducing

$$\tilde{\boldsymbol{L}} = \boldsymbol{L}\boldsymbol{D}^{1/2} \in \mathbb{K}^{N_1 \times R} \quad \text{and} \quad \tilde{\boldsymbol{S}} = \boldsymbol{S}\boldsymbol{D}^{1/2} \in \mathbb{K}^{N_2 \times R},$$

the left and right singular vectors associated to zero singular vectors are eliminated. This yields a *rank* representation of the matrix:

$$\boldsymbol{U} = \tilde{\boldsymbol{L}}\tilde{\boldsymbol{S}}^H = \sum_{r=1}^{R} \tilde{\boldsymbol{l}}^r \otimes \tilde{\boldsymbol{s}}^r, \tag{1.21}$$

where $\tilde{\boldsymbol{l}}^r$ and $\tilde{\boldsymbol{s}}^r$ are the r-th columns of $\tilde{\boldsymbol{L}}$ and $\tilde{\boldsymbol{S}}$, respectively. The amount of information to be stored is reduced from $N_1 N_2$ to $(N_1 + N_2)R$. By choosing $R^* < R$ the matrix can be further compressed at the cost of introducing some approximation error that can be easily evaluated thanks to the optimality of SVD[5] [64]. Such approximation, computed by truncation of the expansion, is called a *low-rank* approximation.

1.3.2 Preliminaries and notation

The extension of the concepts presented in §1.3.1 to higher dimensions is not immediate; it requires further conceptual developments and the notation is also quite technical. To keep the exposition of the different tensor representations as clear as possible, let us introduce in this section some notation, definitions and very basic operations on tensors. Of course we do not mean to provide a complete collection of them but only those strictly needed for the subsequent discussion. For a more exhaustive compilation, see [77] or [80]. Let us denote by

$$\boldsymbol{\mathcal{U}} \in \mathbb{K}^{\mathcal{N}}$$

a D-dimensional tensor with $N_1, \ldots, N_D$ entries per dimension, that we assume known. Recall that $\mathcal{N} = N_1 \times \ldots \times N_D$. A tensor entry is denoted by $u_{i_1,\ldots,i_D}$, with $1 \leq i_d \leq N_d$ for $1 \leq d \leq D$. Then we define:

[5] Optimality in the sense that $\forall R^* \leq R$, the first R^* left and right singular vectors provide the best possible rank-R^* approximation of the matrix.

Diagonal tensors. A tensor is diagonal if $u_{i_1,\ldots,i_D} \neq 0$ only if $i_1 = i_2 = \ldots = i_D$. Moreover, it is superdiagonal if $N_1 = \ldots = N_D$.

Rank-one tensors. A tensor is rank one if it can be written as $\boldsymbol{u}_1 \otimes \boldsymbol{u}_2 \otimes \cdots \otimes \boldsymbol{u}_D$, with $\boldsymbol{u}_d \in \mathbb{K}^{N_d}$ vectors for $1 \leq d \leq D$.

Fiber. A d-fiber is the vector obtained by fixing every dimension except d. By convention, fibers are oriented as columns.

Matricization. A d-matricization, denoted by $\boldsymbol{U}_{(d)}$, arranges every d-fiber as columns of a matrix. Therefore,

$$\boldsymbol{U}_{(d)} \in \mathbb{K}^{N_d \times \mu} \quad \text{with} \quad \mu = \prod_{k=1, k\neq d}^{D} N_k.$$

This definition can be generalized by considering a tuple of dimensions $t \subset \{1, \ldots, D\}$ instead of a single dimension d. If for instance $t = \{1, 2\}$, the matricization is then obtained by arranging the fibers of every dimension contained in t as columns in a matrix $\boldsymbol{U}_{(t)} \in \mathbb{K}^{N_1 N_2 \times \mu_t}$, with $\mu_t = \prod_{k=1, k\notin t}^{d} N_k$.

Tensor d-product. The tensor d-product defines a change of basis in the case when the tensor defines a multilinear operator. A new tensor $\mathcal{Y}$ is obtained from the old one $\mathcal{U}$ by the linear application $\boldsymbol{A} \in \mathbb{K}^{N_d^* \times N_d}$ to each d-fiber. If we consider the d-matricization of the tensor, the operation can be expressed as $\boldsymbol{Y}_{(d)} = \boldsymbol{A}\boldsymbol{U}_{(d)}$. Avoiding matricizations, the operation is denoted as $\mathcal{Y} = \mathcal{U} \times_d \boldsymbol{A}$, whose size in dimension d is now N_d^* instead of N_d.

Multilinear multiplication. Consider matrices $\boldsymbol{A}_d \in \mathbb{K}^{N_d^* \times N_d}$ for $1 \leq d \leq D$. The multilinear multiplication is obtained from the tensor d-product as

$$\mathcal{Y} = \mathcal{U} \times_1 \boldsymbol{A}_1 \cdots \times_D \boldsymbol{A}_D,$$

whose size is now $N_1^* \times \cdots \times N_D^*$.

1.3.3 Some tensor representations of interest

In this section we introduce three different kinds of tensor representations: Canonical[6] [33, 66], Tucker [127] and Hierarchical Tucker representations [65], although other tensor representations exist. We refer to [80] and [62] for recent and complete reviews on tensor representations. We pay particular attention to the computational complexity associated to each one of these tensor formats.

[6] Called CP in some communities. CP stands for CANDECOMP/PARAFAC (CANonical DECOMPosition/PARAllel FACtors)

1.3.3.1 Canonical representation

The canonical representation [33, 66] is an intuitive extension to the higher-dimensional case of the matrix representation introduced in Eq. (1.21). Therefore the canonical rank is defined analogously as the smallest R such that

$$\mathcal{U} = \mathcal{U}^{(R)} := \sum_{r=1}^{R} \boldsymbol{u}_1^r \otimes \cdots \otimes \boldsymbol{u}_D^r, \tag{1.22}$$

the sum of R rank-one tensors. Eq. (1.22) also defines the canonical representation of a tensor. We will show in §1.3.3.2 that the canonical rank is not a proper definition of tensor rank. To facilitate the exposition of the other tensor sets in next sections, let us define a matrix $\boldsymbol{U}_d \in \mathbb{K}^{N_d \times R}$ that contains all vectors $\boldsymbol{u}_d^r$ corresponding to dimension d. If they are are normalized, the weight of each rank-one tensor in the tensor representation can be represented by a vector $\boldsymbol{\lambda} \in \mathbb{R}^R$ so that

$$\mathcal{U} \quad \text{and} \quad \mathcal{C}_R(\mathcal{U}) := [\![\boldsymbol{\lambda}; \boldsymbol{U}_1, \ldots, \boldsymbol{U}_D]\!] \tag{1.23}$$

are equivalent representations of the same data but with different computational complexity. The operator $[\![\cdot]\!]$ denotes the compressed representation to be stored in practice. It is to highlight that in canonical format the tensor is approximated using only $(N_1 + \ldots + N_D)\, R$ data, which for small R is much less than $\prod_{d=1}^{D} N_D$, corresponding to a full representation. The complexity is therefore on the order of $\mathcal{O}(DNR + R)$, provided that a uniform grid is chosen, i.e. $N_1 = \ldots = N_D = N$. Therefore the complexity of a tensor in Canonical format scales linearly with the number of dimensions instead of exponentially.

However, the main concern with canonical representation is that given a tensor, it is extremely hard to find out its rank, as we will show in §1.3.4. Therefore, although it is obvious that every tensor has a canonical representation, it may be as complex as the full representation[7], although this is not what is experienced most of practical applications with PGD method, that implements the canonical representation. In next sections we present other formats that generalize the canonical one and allow an extension of SVD for higher dimensions.

1.3.3.2 Tucker representation

Consider Eq. (1.22) and Eq. (1.23). Let us highlight that:

- The r-th column of $\boldsymbol{U}_d$ is only combined with the r-th column of the other matrices $\boldsymbol{U}_k$, with $k \neq d$.

[7] Worst case consists in considering the Euclidean canonical basis in each dimension and taking every possible combination.

- All matrices $\boldsymbol{U}_d$ have the same number of columns, that is, a generalization of the equivalence of row and column ranks for matrices.

It can be therefore shown intuitively that the canonical format can be generalized by permuting columns: the r_1-th column of $\boldsymbol{U}_1$ could multiply the r_2-th column of $\boldsymbol{U}_2$ and the r_3-th column of $\boldsymbol{U}_3$... and so on, with $1 \leq r_1, r_2 \ldots \leq R$. More compactly, rank-one tensors may not share the same superscript: $\boldsymbol{u}_1^{r_1} \otimes \boldsymbol{u}_2^{r_2} \cdots \otimes \boldsymbol{u}_D^{r_D}$. Furthermore, one could also wonder why each dimension should have the same rank R; this hypothesis is clearly unnecessary and may be removed by letting each dimension to have its own rank R_d. Hence, $1 \leq r_d \leq R_d \leq N_d$ for $1 \leq d \leq D$, which states that of course the rank in each dimension cannot be bigger than number of entries.

With all these ingredients, the Tucker representation [127] can be readily introduced as follows:

$$\mathcal{U} = \mathcal{U}^{(\boldsymbol{R})} := \sum_{r_1=1}^{R_1} \cdots \sum_{r_D=1}^{R_D} \boldsymbol{u}_1^{r_1} \otimes \cdots \otimes \boldsymbol{u}_D^{r_D}, \tag{1.24}$$

where $\boldsymbol{u}_d^{r_d} \in \mathbb{K}^{N_d}$ is the r_d-th principal component or singular vector[8] in dimension d. Scalars R_d are called the d-rank of the tensor. They define the tensor rank, sometimes called *multi-rank*, denoted by

$$\boldsymbol{R}_T := (R_1, \ldots, R_D).$$

If vectors $\boldsymbol{u}_d^{r_d}$ are normalized and stored as columns in matrices $\boldsymbol{U}_d \in \mathbb{K}^{N_d \times R_d}$, the weight of each rank-one tensor formed from the $(r_1, \ldots, r_D)$ indices can be represented as the *core tensor*, denoted by $\mathcal{K} \in \mathbb{R}^{R_1 \times \cdots \times R_d}$. The entries of the core tensor are also called singular values. Therefore:

$$\mathcal{U} \quad \text{and} \quad \mathcal{T}_{\boldsymbol{R}}(\mathcal{U}) := [\![\mathcal{K}; \boldsymbol{U}_1, \ldots, \boldsymbol{U}_D]\!] \tag{1.25}$$

are equivalent representations of the same data but with different computational complexity. The Tucker representation is a compressed version of the original tensor if the following condition is satisfied: $\prod_{d=1}^{D} R_d < \prod_{d=1}^{D} N_d$. Compared to the Canonical representation, the Tucker's one is quite expensive as the complexity of the approximation is in the order of $\mathcal{O}(R^D + DNR)$, under the following simplification assumptions: same number of entries N per dimension; and same d-rank in every dimension. Therefore, in some sense Tucker representations suffer from the *curse of rank* due to the need of storing the core tensor, whose size scales exponentially with the number of dimensions.

It is worth to highlight that Tucker representation generalizes the Canonical representation; indeed, if the core tensor is taken superdiagonal (§1.3.2 for its definition), the Tucker representation reduces to the Canonical one. Tucker representation allows introducing a generalization of the SVD —called the High-Order SVD (HOSVD) [46]— that yields quasi-optimal approximations of the tensor. This will be discussed in §1.3.4. Finally, it is worth

[8] It is called like that because Tucker format is closely related to a generalization of the SVD, see §1.3.4.1.

to remark that Tucker representation is analogous to tensor product spaces: it suffices to introduce Euclidean vector spaces of dimension R_d that span the approximation space V_d of dimension N_d.

1.3.3.3 Hierarchical Tucker representation

Hierarchical Tucker (HT) representations [65] are an attempt to keep the generality of the Tucker representation but reducing at the same time its complexity, which was greatly affected by the size of the core tensor. In Chapter 4, we shall refer to this kind of representation. HT is based on the idea of recursively splitting the dimensions. This makes the exposition quite technical. Consider a binary tree $\mathcal{T}$ constructed by recursive splitting of $\{1, 2, \ldots, D\}$. Fig. 1.2 shows two examples of this binary tree in the case of four and five dimensions. The points where a branch splits into two other branches are called nodes. We

$$\{1,2,3,4\} \to \left| \begin{array}{ll} \{1,2\} \to & \left| \begin{array}{l} \{1\} \\ \{2\} \end{array} \right. \\ \{3,4\} \to & \left| \begin{array}{l} \{3\} \\ \{4\} \end{array} \right. \end{array} \right. \qquad \{1,2,3,4,5\} \to \left| \begin{array}{ll} \{1,2,3\} \to & \left| \begin{array}{ll} \{1,2\} \to & \left| \begin{array}{l} \{1\} \\ \{2\} \end{array} \right. \\ & \{3\} \end{array} \right. \\ \{4,5\} \to & \left| \begin{array}{l} \{4\} \\ \{5\} \end{array} \right. \end{array} \right.$$

Figure 1.2: Examples of the binary tree $\mathcal{T}$ for $D = 4$ (left) and $D = 5$ (right)

associate each node with t, a subset of $\{1, 2, \ldots, d\}$. The collection of all these subsets is $\mathcal{T}$. The subset t is formed from the union of two subsets from an inferior level, i.e. $t = t_r \cup t_s$, such that $t_r \cap t_s = \emptyset$. In consequence, $\mathcal{T}$ is built in such a way that enjoys of the property of nestedness.

Let us consider some $t \in \mathcal{T}$ (i.e. some node of the tree) and its corresponding matricization, that is the matrix $\boldsymbol{U}_{(t)}$ (see §1.3.2 for details on notation). That matrix might not be of full rank, i.e. $R_t \leq \mu_t$, being μ_t the number of columns of the matricization. Let us denote by $\boldsymbol{U}_t$ the matrix whose columns span the image of the matricization. It has, of course, R_t columns. Recall that t is formed from the union of t_r and t_s. Again, we may consider matricizations $\boldsymbol{U}_{(t_r)}$ and $\boldsymbol{U}_{(t_s)}$, and their corresponding rank matrices $\boldsymbol{U}_{t_r}$ and $\boldsymbol{U}_{t_s}$. Their respective column ranks are denoted by

$$R_{t_r} \leq \mu_{t_r} \quad \text{and} \quad R_{t_s} \leq \mu_{t_s}.$$

It can be proven [61] that there exist a matrix $\boldsymbol{B}_t \in \mathbb{K}^{R_{t_r} R_{t_s} \times R_t}$ that allows passing from a lower level (t_r,t_s) in the network to the upper level (t) through the following expression:

$$\boldsymbol{U}_t = (\boldsymbol{U}_{t_l} \otimes \boldsymbol{U}_{t_s}) \boldsymbol{B}_t. \tag{1.26}$$

Eq. (1.26) has important practical implications. Using recursivity, it is clear that for any configuration of the network $\mathcal{T}$, matrices $\boldsymbol{U}_t$ only need to be stored on the final nodes —

also called leaf nodes— because any other level can be reached by applying successively the transformation defined by Eq. (1.26). To do so, matrices $\boldsymbol{B}_t$ must be stored at each node except the leaf ones. Therefore:

$$\begin{aligned} \mathcal{U} \quad \text{and} \quad \mathcal{H}_{\boldsymbol{R}}(\mathcal{U}) &:= [\![\boldsymbol{B}_t, \boldsymbol{U}_d]\!], \\ \forall t \in \mathcal{T} \setminus \{1, \dots, D\} \quad \text{and} \quad 1 \leq d \leq D, & \end{aligned} \tag{1.27}$$

are equivalent representations of the same data but with different computational complexity. HT representation also defines its own rank, called the HT-rank, which is formed by the tuple $\boldsymbol{R}_H := (R_t)_{t \in \mathcal{T}}$, with R_t the column rank of the matricization at node t, as it has already been defined.

Since matrices $\boldsymbol{U}_d$ only need to be stored in the leaf nodes and matrices $\boldsymbol{B}_t$ are stored in all nodes but the leaf ones, the computational complexity is in the order of $\mathcal{O}(DR^3 + DNR)$, which clearly improves Tucker's complexity. That estimation is done assuming that the rank at each node is R and for all dimensions the number of nodes is N. However, operating with HT tensors is not always straightforward. For instance, the complexity of the inner product between tensors in HT format is in the order of $\mathcal{O}(DNR^2 + DR^4)$.

1.3.4 Explicit algorithms to compute Low-rank Tensor Approximations

The best tensor representation of a given tensor is the solution of the following problem:

$$\mathcal{S}^{\star}_{\boldsymbol{R}}(\mathcal{U}) := \arg \min_{\mathcal{V} \in \mathcal{S}_{\boldsymbol{R}}} \|\mathcal{U} - \mathcal{V}\|,$$

where $\|\cdot\|$ denotes the analogous for tensors of the Frobenius norm form matrices. By $\mathcal{S}_{\boldsymbol{R}}$, we denote either

- The set of tensors with canonical rank bounded $R_C \leq R$, denoted by $\mathcal{C}_R$. Observe that it could also be denoted as $\mathcal{C}_{\boldsymbol{R}}$ with $\boldsymbol{R} = (R, R, \dots)$, since the canonical rank is a particular case of the tensor rank in which every d-rank is equal.
- The set of tensors with tensor rank bounded $\boldsymbol{R}_T \leq \boldsymbol{R}$, denoted by $\mathcal{T}_{\boldsymbol{R}}$.
- The set of tensors with HT rank bounded $\boldsymbol{R}_H \leq \boldsymbol{R}$, denoted by $\mathcal{H}_{\boldsymbol{R}}$.

It can be proved that the canonical set is in general not closed, with the following exceptions: the set of elementary tensors of rank one, $\mathcal{C}_1$, and the two-dimensional case for any rank —matrices—, which in general renders the problem of optimality ill posed [47]. There exist no algorithm to determine the canonical rank of a tensor; in fact, it is a NP-hard problem [67]. Moreover, the rank of a real-valued tensor may be actually different over $\mathbb{R}$ and $\mathbb{C}$; see [80] for an example of a tensor of rank two over the field of complex numbers and three over the field of real numbers.

All these reasons make that, although it is of course possible to compute canonical representations of a tensor, the problem of finding the best possible canonical representation is not considered in general. In addition, the truncation error associated to consider $R^* \leq R$ modes cannot be evaluated straightforwardly.

1.3.4.1 High-Order Singular Value Decomposition

Given a tensor it is easy to find its Tucker representation by means of the High-Order Singular Value Decomposition (HOSVD). The HOSVD algorithm [46] computes for each dimension the SVD of $\boldsymbol{U}_{(d)}$, the matricization along that dimension. If the column rank of this matricization is R_d, we only need to keep the leading R_d left singular vectors. The singular vectors are stored in matrices $\boldsymbol{U}_d$, already defined in §1.3.3.2. Once this orthonormal matrices have already been computed by successive SVD in each dimension, the core tensor can be obtained as:

$$\mathcal{K} = \mathcal{U} \times_1 \boldsymbol{U}_1^H \cdots \times_D \boldsymbol{U}_D^H.$$

See §1.3.2 for the definition of the multilinear product. Contrary to the canonical tensor set, the Tucker tensor set is closed. Taking advantage of the closedness, it turns out that HOSVD provides quasi-optimal representations. Truncation can be used to further compress the data by keeping a number of leading left singular vectors $R_d^* \leq R_d$, for each dimension. The error associated to that truncation can be assessed as follows:

$$\|\mathcal{U} - \mathcal{T}_{\boldsymbol{R}^*}(\mathcal{U})\| \leq \sqrt{\sum_{d=1}^{D} \sum_{i_d=R_d^*+1}^{R_d} \sigma_{i_d}^2} \leq \sqrt{D}\, \mathcal{T}_{\boldsymbol{R}}^{\star}(\mathcal{U}),$$

where σ_{i_d} are the entries of the core tensor discarded by effect of truncation.

1.3.4.2 Hierarchical Singular Value Decomposition

Given a tensor, a Tucker representation can be computed by means of the hierarchical singular value decomposition of tensors (HSVD) [61]. Since presenting the algorithm for arbitrary dimensions is quite technical, we will illustrate it by means of an example.

Example 1.5 (*Illustrating the HSVD algorithm*). We develop here the algorithm for a very simple case with a double objective:

- First, to illustrate the definition of HT tensors previously given.
- Second, to show practically the way matrices $\boldsymbol{B}_t$ are computed.

Hence, let us consider leaf nodes $t_r = \{1\}$ and $t_s = \{2\}$ such that the upper level of the tree is given by $t = t_r \cup t_s = \{1, 2\}$. Let us apply SVD to the matricizations $\boldsymbol{U}_{(t_r)}$ and $\boldsymbol{U}_{(t_s)}$ to compute their left singular values, respectively. However, it can be noticed that instead of performing two matricizations and two SVD, it is possible perform only one SVD to obtain both sets of left singular vectors:

$$\boldsymbol{U}_{(r)} = \boldsymbol{U}_r \boldsymbol{U}_s^H, \quad \text{equivalently} \quad \boldsymbol{U}_{(s)} = \boldsymbol{U}_s \boldsymbol{U}_r^H,$$
$$\text{with} \quad \boldsymbol{U}_r \in \mathbb{K}^{N_r \times R_r}, \quad \boldsymbol{U}_s \in \mathbb{K}^{N_s \times R_s}.$$

In our example, this is obvious because it is a two-dimensional case and indeed both R_r and R_s will be the same. This operation can be repeated until the leaf nodes are reached. Observe that if we define the transfer matrix

$$\boldsymbol{B}_t = (\boldsymbol{U}_s^H \otimes \boldsymbol{U}_r^H)\, \boldsymbol{U}_t,$$

then $\boldsymbol{U}_t$ can be recovered from

$$\boldsymbol{U}_t = (\boldsymbol{U}_r \otimes \boldsymbol{U}_s)\, \boldsymbol{B}_t, \tag{1.28}$$

because of the orthogonality of every matrix $\boldsymbol{U}_{t \in \mathcal{T}}$, except for the root node, that is $t = \{1, 2, \ldots, D\}$, which corresponds to the original tensor.

The HT set of tensors is closed and HT representations computed using the HTSVD algorithm enjoy from quasi-optimality, as shown in Eq. (1.29). Truncation can be performed by keeping $R_t^* < R_t$ left singular vectors at each node $t \in \mathcal{T}$. In that case, the error can be evaluated from the singular values associated to the discarded left singular vectors when performing the SVD of each matricization, as shown in the previous example.

$$\|\mathcal{U} - \mathcal{H}_{\boldsymbol{R}^*}(\mathcal{U})\| \leq \sqrt{\sum_{t \in \mathcal{T}} \sum_{i_t = R_t^* + 1}^{R_t} \sigma_{i_t}^2} \leq \sqrt{2D - 3}\, \mathcal{H}_{\boldsymbol{R}}^{\star}(\mathcal{U}), \tag{1.29}$$

1.3.4.3 Summary of tensor decomposition properties

In this section we summarize the most important properties of each one of the tensor representations considered in this work. As it can be seen in Table 1.1, the canonical format has a big advantage regarding its complexity; it is the format that a priori offer the potential compression of the original tensor. However, since the canonical rank of a tensor is very hard to compute —unless it is already given in canonial format, of course— there exist no methods that guarantee a quasi-best approximation. On the contrary, Tucker and Hierarchical Tucker representations allow computing quasi-best approximations using SVD-based algorithms.

It is worth to emphasize that the algorithms presented in §1.3.4.1 and §1.3.4.2 can be applied to a tensor that is known explicitly, which is not the usual case. More commonly,

	Canonical	**Tucker**	**Hierarchical**
Complexity	$\mathcal{O}(ndr)$	$\mathcal{O}(ndr + r^d)$	$\mathcal{O}(ndr + dr^3)$
Closedness	No	Yes	Yes
Quasi-best approx.	No	Yes	Yes
Best approx.	No	Exist (NP hard)	Exist (NP hard)

Table 1.1: Summary of properties of tensor representations

the tensor is given implicitly as the solution of some linear system of equations, as shown in Eq. (1.8). In §1.4.2 we address this kind of problems.

1.4 Implicit algorithms to compute low-rank tensor approximations

In §1.1, MOR methods where categorized depending on whether they define an approximation of the solution or not in the multidimensional domain. These approximations introduce essential differences that lead to a formulation in tensor product spaces. In §1.2.3, we showed that direct schemes are precluded to compute tensor approximations due to the curse of dimensionality. It was therefore concluded that tensors must be computed and manipulated in compressed formats. Such formats have been studied in §1.3 as well as some compression algorithms that operate on explicit tensors. In computational mechanics, however, tensors are given implicitly as the solution of some algebraic problem, e.g. a linear system, that may be written in the following equivalent forms:

$$\begin{cases} \boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{U}}) = \boldsymbol{\mathcal{F}} \\ \boldsymbol{\mathcal{A}} : \mathbb{K}^{\mathcal{N}} \to \mathbb{K}^{\mathcal{N}} \quad \text{and} \quad \boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{F}} \in \mathbb{K}^{\mathcal{N}} \end{cases} \Leftrightarrow \begin{cases} \boldsymbol{A}\boldsymbol{u} = \boldsymbol{f} \\ \boldsymbol{A} \in \mathbb{K}^{|\mathcal{N}|\times|\mathcal{N}|} \quad \text{and} \quad \boldsymbol{u}, \boldsymbol{f} \in \mathbb{K}^{|\mathcal{N}|}. \end{cases} \tag{1.30}$$

Therefore we are interested in algorithms that allow computing a low-rank approximation of the solution by constraining the tensor to belong to some tensor set defined in §1.3.3.

1.4.1 A general overview on optimization-based algorithms

Many algorithms are available to treat this kind of problems. The list includes classic iterative methods for solving linear systems (e.g. conjugate gradient like methods) combined with truncation in any of the low-rank tensor representations, optimization-based algorithms, and many others. We refer to [62] and the references therein for a complete list of them.

Optimization-based algorithms turn Eq. (1.30) into an optimization problem. As objective function one might choose:

- The norm of the error: $\|\boldsymbol{u} - \boldsymbol{A}^{-1}\boldsymbol{f}\|$, which is not useful at all from the point of view of optimization as it involves the direct solve of the system of equations.
- The norm of the residual: $\|\boldsymbol{f} - \boldsymbol{A}\boldsymbol{u}\|$.
- The energy norm of the error, defined as follows:

$$|||\boldsymbol{u} - \boldsymbol{A}^{-1}\boldsymbol{b}|||^2 := \langle \boldsymbol{A}(\boldsymbol{u} - \boldsymbol{A}^{-1}\boldsymbol{b}), \boldsymbol{u} - \boldsymbol{A}^{-1}\boldsymbol{b}\rangle.$$

  The operator $\boldsymbol{A}$ must define a norm, i.e. it must be hermitian positive definite.

Throughout this section, we use the Euclidean inner product and norm, that is:

$$\langle \boldsymbol{u}, \boldsymbol{v}\rangle = \boldsymbol{v}^H\boldsymbol{u} \quad \text{and} \quad \|\boldsymbol{u}\| = \langle \boldsymbol{u}, \boldsymbol{u}\rangle^{1/2}.$$

Observe that

$$|||\boldsymbol{u} - \boldsymbol{A}^{-1}\boldsymbol{b}|||^2 = \langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{u}\rangle - 2\langle \boldsymbol{b}, \boldsymbol{u}\rangle + \langle \boldsymbol{A}^{-1}\boldsymbol{b}, \boldsymbol{b}\rangle,$$

and therefore the optimization problem may be set in terms of a functional which is cheap to compute —recall the separation property of the inner product in spaces with tensor structure—:

$$\min_{\boldsymbol{u}\in \mathrm{vect}(\mathcal{S})} \mathcal{J}(\boldsymbol{u}) := \frac{1}{2}\langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{u}\rangle - \langle \boldsymbol{f}, \boldsymbol{u}\rangle. \tag{1.31}$$

In Eq. (1.31) we have constrained the solution to belong to a certain tensor subset such as the ones defined in §1.3.4. If the operator was not self-adjoint, the optimization problem would be obtained by minimizing the norm of the residual [25, 52, 106]. The resulting constrained optimization is nonlinear and non-convex in general [53, 62], even if the quadratic form associated to the original problem was strictly convex.

1.4.2 Proper Generalized Decomposition as an algebraic solver

In §1.1.3.3, PGD was introduced as a differential solver for a parametrized problem in the context of a priori model reduction. In this section, PGD is regarded as a method that allows constructing a Low-rank tensor approximation —recall that both regards are equivalent—. It uses the following principal ingredients:

- The tensor is sought in canonical format, already introduced in §1.3.3.1.
- Instead of predefining the rank approximation and then apply optimization techniques, the solution is built up from successive rank-one corrections. The convergence is driven by some measure of the residual.

- Each rank-one correction requires solving a nonlinear optimization problem. The alternating direction method is classically applied to solve it.

For the sake of simplicity, we only consider here hermitian positive-definite problems. Non-hermitian (i.e. non-symmetric) problems are considered in §1.4.3. Therefore, assume that after R successive corrections of the solution, a rank-R approximation has already been computed. As it is written in canonical format, we have:

$$\boldsymbol{u}^{(R)} = \sum_{r=1}^{R} \boldsymbol{u}^r \quad \text{with} \quad \boldsymbol{u}^r = \bigotimes_{d=1}^{D} \boldsymbol{u}_d^r \quad \text{for} \quad 1 \leq r \leq R. \tag{1.32}$$

Suppose that we want to compute a rank-one correction denoted by $\boldsymbol{u}$, that is

$$\boldsymbol{u}^{(R+1)} = \boldsymbol{u}^{(R)} + \boldsymbol{u},$$
$$\text{with} \quad \boldsymbol{u} \in \text{vect}(\mathcal{C}_1) \quad \Leftrightarrow \quad \boldsymbol{u} = \boldsymbol{u}_1 \otimes \cdots \otimes \boldsymbol{u}_D,$$

such that

$$\boldsymbol{A}\boldsymbol{u} \approx \boldsymbol{f} - \boldsymbol{A}\boldsymbol{u}^{(R)}. \tag{1.33}$$

Just as shown in Eq. (1.31), the correction can be determined by turning Eq. (1.33) into the following nonlinear optimization problem [7, 53]:

$$\min_{\boldsymbol{u} \in \text{vect}(\mathcal{C}_1)} \frac{1}{2}\langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{u}\rangle + \frac{1}{2}\langle \boldsymbol{A}\boldsymbol{u}^{(R)}, \boldsymbol{u}\rangle - \langle \boldsymbol{f}, \boldsymbol{u}\rangle, \tag{1.34}$$

which is solved in practice applying the alternating directions algorithm, although of course any other nonlinear optimization method would be valid. Let us define the residual after R corrections and suppose that it can be written as a rank-S tensor, with $S \geq R$ in principle. We denote the residual by

$$\boldsymbol{r}^{(S)} := \boldsymbol{f} - \boldsymbol{A}\boldsymbol{u}^{(R)}.$$

Let us suppose that the operator possesses a rank-T separated form as shown in Eq. (1.20). Using the definition of the residual, the functional to be minimized is defined as follows:

$$\mathcal{J}(\boldsymbol{u}) := \frac{1}{2}\sum_{t=1}^{T} \langle \boldsymbol{A}^t\boldsymbol{u}, \boldsymbol{u}\rangle - \sum_{s=1}^{S} \langle \boldsymbol{r}^s, \boldsymbol{u}\rangle$$

where matrices $\boldsymbol{A}^t \in \mathcal{C}_1(\mathbb{K}^{|\mathcal{N}|\times|\mathcal{N}|})$ are rank-one. By the separation property of the scalar product, see Eq. (1.18), we have:

$$\mathcal{J}(\boldsymbol{u}) = \frac{1}{2}\sum_{t=1}^{T}\prod_{d=1}^{D} \langle \boldsymbol{A}_d^t\boldsymbol{u}_d, \boldsymbol{u}_d\rangle - \sum_{s=1}^{S}\prod_{d=1}^{D} \langle \boldsymbol{r}_d^s, \boldsymbol{u}_d\rangle.$$

The alternating directions algorithm optimizes one direction at a time. Assume $\boldsymbol{u}_d$ known for $1 \leq d \leq D$, $d \neq k$. Consequently, every scalar product can be explicitly computed except

for $d = k$, which results in:

$$\mathcal{J}(\boldsymbol{u}_k) = \frac{1}{2}\sum_{t=1}^{T}\alpha_t\langle \boldsymbol{A}_k^t\boldsymbol{u}_k, \boldsymbol{u}_k\rangle - \sum_{s=1}^{S}\beta_s\langle \boldsymbol{r}_k^s, \boldsymbol{u}_k\rangle,$$

with the following scalars

$$\alpha_t = \prod_{d=1,d\neq k}^{D}\langle \boldsymbol{A}_d^t\boldsymbol{u}_d, \boldsymbol{u}_d\rangle, \quad 1\leq k\leq T \quad \text{and} \quad \beta_s = \prod_{d=1,d\neq k}^{D}\langle \boldsymbol{r}_d^s, \boldsymbol{u}_d\rangle, \quad 1\leq s\leq S. \tag{1.35}$$

Therefore, for each direction $1 \leq k \leq D$, the mode $\boldsymbol{u}_k$ is optimized by solving a minimization problem. Calculus of variations allows us writing the equivalent problem:

$$\min_{\boldsymbol{u}_k\in\mathbb{K}^{N_k}} \mathcal{J}(\boldsymbol{u}_k) \quad \Leftrightarrow \quad \text{Find } \boldsymbol{u}_k\in\mathbb{K}^{N_k}: \quad \langle \tilde{\boldsymbol{A}}_k\boldsymbol{u}_k, \boldsymbol{w}\rangle = \langle \tilde{\boldsymbol{r}}_k, \boldsymbol{w}\rangle, \quad \forall \boldsymbol{w}\in\mathbb{K}^{N_k}, \tag{1.36}$$

which imposes the cancellation of the projection of the residual for every $\boldsymbol{w}$. In such sense, this is a Galerkin formulation of the PGD. We have used the following notation:

$$\tilde{\boldsymbol{A}}_k := \sum_{t=1}^{T}\alpha_t\boldsymbol{A}_k^t \quad \text{and} \quad \tilde{\boldsymbol{r}}_k := \sum_{s=1}^{S}\beta_s\boldsymbol{r}_k^s. \tag{1.37}$$

We refer to an iteration of the alternating directions method as the updating and normalization of every dimension $1 \leq k \leq D$, and we denote it as $\boldsymbol{u}_{(i)}$, the i-th iteration. No distinct notation is used for normalized modes for the sake of simplicity. The alternating directions algorithm is said to be converged either when a maximum number of iterations is attained or a stationary point is reached, that is,

$$\|\boldsymbol{u}_{(i)} - \boldsymbol{u}_{(i-1)}\| < \varepsilon_1. \tag{1.38}$$

The low-rank approximation is said to be converged when after computing R corrections, the norm of the residual compared to the right-hand side (that is, the residual with $R = 0$) has been reduced by some factor ε_2:

$$\|\boldsymbol{f} - \boldsymbol{A}\boldsymbol{u}^{(R)}\| < \varepsilon_2\,\|\boldsymbol{f}\|. \tag{1.39}$$

Residual-based error estimates are also available in [4, 87].

It is to be highlighted that the PGD algorithm is able to build a canonical low-rank approximation by solving a series of low-dimensional problems. The potential bottleneck is therefore in the alternating directions method: if it requires many iterations to converge, the efficiency is deteriorated because several linear systems have to be solved. In multi-parametrized problems, this is of special importance since the number of degrees of freedom associated to the discretization of the physical space is generally much bigger than the number of DOFs used in the other (parametric) dimensions. Therefore the computational cost is driven by the linear systems that must be solved to update the space dimension. In practice, some authors [106] suggest to cut off the alternating iterations (three or four maximum) arguing that it is not worth spending much iterations to optimize a single correction that possibly will be improved by the subsequent correction anyway.

Algorithm 1 Rank-one corrections PGD

Require: $\boldsymbol{u}^{(R)}$ ▷ Rank-R approximation
1: Initialize $\boldsymbol{u}$
2: **for** $i = 1$ to I **do** ▷ Alternating directions loop
3: Set $\boldsymbol{u}_{\text{old}} = \boldsymbol{u}$
4: **for** $k = 1$ to D **do** ▷ Loop every dimension
5: Update $\boldsymbol{u}_k$
6: **end for**
7: Check $\boldsymbol{u} - \boldsymbol{u}_{\text{old}}$ ▷ Convergence Eq. (1.38)
8: **end for**
9: **return** $\boldsymbol{u}$ ▷ Rank-one correction

1.4.3 Alternative formulations of Proper Generalized Decomposition

Algorithm 1 is a simple and computationally very attractive formulation of PGD. Under some circumstances it can be proven that it constitutes a generalization of POD [106]. In particular, for two-dimensional problems and hermitian positive-definite operators, the rank-one corrections converge towards the POD decomposition. For the general case, however, PGD is not guaranteed to be optimal is such sense. Several alternative formulations have been proposed with the objective of obtaining more compact representations.

1.4.3.1 Optimal Galerkin formulation

At iteration R, rather than computing a rank-one correction we prefer computing a full rank-R canonical approximation [106]. Under the same assumptions made in §1.4.2, the optimization problem may be written now as

$$\min_{\boldsymbol{u}\in\text{vect}(\mathcal{C}_R)} \mathcal{J}(\boldsymbol{u}) := \frac{1}{2}\langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{u}\rangle - \langle \boldsymbol{f}, \boldsymbol{u}\rangle,$$

equivalently,

$$\min_{\boldsymbol{u}^r\in\text{vect}(\mathcal{C}_1),\, 1\le r\le R} \mathcal{J}(\boldsymbol{u}^1, \dots, \boldsymbol{u}^R) = \frac{1}{2}\sum_{r=1}^{R}\sum_{s=1}^{R}\langle \boldsymbol{A}\boldsymbol{u}^s, \boldsymbol{u}^r\rangle - \sum_{r=1}^{R}\langle \boldsymbol{f}, \boldsymbol{u}^r\rangle, \tag{1.40}$$

that may be solved applying the alternating directions algorithm. Assume $\boldsymbol{u}_d^r$ known for $1 \le d \le D$, $d \ne k$, and $1 \le r \le R$. The only unknowns of the functional are the R modes in dimension k, since every scalar product can be explicitly computed except for dimension k. Therefore, we do not aim at optimizing a single mode per dimension but every mode per dimension at the same time:

$$\min_{\boldsymbol{u}_k^r\in\mathbb{K}^{N_k},\, 1\le r\le R} \mathcal{J}(\boldsymbol{u}_k^1, \dots, \boldsymbol{u}_k^R). \tag{1.41}$$

Eq. (1.41) can be solved in a standard manner by applying calculus of variations. The resulting linear system is coupled of size $N_k R$. Then, the algebraic problem can be written as follows:

$$\text{Find } \boldsymbol{U}_k := \otimes_{r=1}^{R} \boldsymbol{u}_k^r \in \mathbb{K}^{N_k R} : \quad \langle \tilde{\boldsymbol{B}}_k \boldsymbol{U}_k, \boldsymbol{W} \rangle = \langle \tilde{\boldsymbol{F}}_k, \boldsymbol{W} \rangle, \quad \forall \boldsymbol{W} \in \mathbb{K}^{N_k R}, \tag{1.42}$$

where operator $\tilde{\boldsymbol{B}}_k$ and vector $\tilde{\boldsymbol{F}}_k$ are obtained from Eq. (1.41). We omit here their derivation which is not difficult but quite tedious. It is to remark that Eq. (1.42) ensures the cancellation of the projection of the residual simultaneously for every $\boldsymbol{W}$. In such sense, this is a optimal Galerkin formulation of the PGD.

This constitutes the first computational disadvantage of this formulation: instead of solving problems of size N_k in each dimension, we must solve much bigger problems. The second disadvantage is that many of them must be solved. To see this, let us suppose S such that the PGD expansion $\boldsymbol{u}^{(R)}$ is converged if and only if $R \geq S$. See Eq. (1.39) for the definition of convergence. Since S is not known a priori, we must compute (in principle) the rank-one, rank-two, rank-three... approximations until the rank-S is reached.

Algorithm 2 Optimal Galerkin PGD

for $r = 1$ to R **do**
  Initialize $\boldsymbol{u}^{(r)}$ ▷ Initialize a rank-r approximation
  for $i = 1$ to I **do**
    Set $\boldsymbol{u}_{\text{old}}^{(r)} = \boldsymbol{u}^{(r)}$
    for $k = 1$ to D **do**
      Update $\boldsymbol{u}_k^1, \ldots, \boldsymbol{u}_k^r$ ▷ Update every mode in current direction
    end for
    Check $\boldsymbol{u}^{(r)} - \boldsymbol{u}_{\text{old}}^{(r)}$
  end for
  Check $\|\boldsymbol{u}^{(r)}\| < \varepsilon_2 \|\boldsymbol{u}^{(0)}\|$ ▷ Convergence Eq. (1.39)
end for
return $\boldsymbol{u}^{(S)}$ ▷ Rank-$(S \leq R)$ decomposition

1.4.3.2 Rank-one corrections with update

We have just shown that the optimal Galerkin formulation of PGD is computationally very expensive, specially for multiparametrized problems in which the space dimension involves large scale (2D or 3D) problems whose number of unknowns is much bigger than the number of unknowns related to the parametric dimensions. In such cases, a mixed formulation of Algorithm 1 and Algorithm 2 may be convenient [106]. Assume that we have just computed the R-th rank-one correction. Equivalently, we dispose of a rank-R approximation. Before computing a new rank-one correction, we may optimize every mode per dimension as in Eq.

(1.41), except the space dimension because it is too expensive. At convergence, we replace the previous R modes by the optimized ones.

Algorithm 3 Rank-one with update PGD

Require: $\boldsymbol{u}^{(R)}$

1: **for** $i = 1$ to I **do**
2: Set $\boldsymbol{u}^r_{k,\text{old}} = \boldsymbol{u}^r_k$ for $2 \leq k \leq D$ and $1 \leq r \leq R$
3: **for** $k = 2$ to D **do** ▷ Assuming $k = 1$ is space dimension
4: Update $\boldsymbol{u}^1_k, \dots, \boldsymbol{u}^r_k$ ▷ Eq. (1.41)
5: **end for**
6: Check alternating directions convergence
7: **end for**
8: Call RANK-ONE PGD($\boldsymbol{u}^{(R)}$) ▷ New rank-one correction with Algorithm 1

1.4.3.3 Galerkin least-square formulation

Every PGD formulation considered until now is based on the hypothesis of the operator being hermitian positive-definite, that is, the operator defines an energy norm. Minimizing the error with respect to this energy norm allowed turning the linear system into a convex optimization problem whose solution was sought in the low-rank canonical tensor subset. The last constraint made the minimization problem to be not convex anymore. When the operator is not self-adjoint, i.e. it does not define a norm, the only possibility is to minimize the norm of the residual.

$$\|\boldsymbol{f} - \boldsymbol{A}\boldsymbol{u}\|^2 = \langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{A}\boldsymbol{u}\rangle - 2\langle \boldsymbol{f}, \boldsymbol{A}\boldsymbol{u}\rangle + \langle \boldsymbol{f}, \boldsymbol{f}\rangle.$$

This yields the following optimization problem:

$$\min_{\boldsymbol{u}\in\text{vect}(\mathcal{S})} \frac{1}{2}\langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{A}\boldsymbol{u}\rangle - \langle \boldsymbol{f}, \boldsymbol{A}\boldsymbol{u}\rangle,$$

which is equivalent to minimize the energy norm of the error for the normal equation: $\boldsymbol{A}^H\boldsymbol{A}\boldsymbol{u} = \boldsymbol{A}^H\boldsymbol{f}$. The optimization problem becomes analogous to Eq. (1.34) or to Eq. (1.40), depending on whether the rank-one corrections or the optimal Galerkin formulations want to be applied. If for instance, we choose the rank-one corrections formulation, the minimization problem after a rank-R decomposition has been computed is as follows:

$$\min_{\boldsymbol{u}\in\text{vect}(\mathcal{C}_1)} \mathcal{J}(\boldsymbol{u}) := \frac{1}{2}\langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{A}\boldsymbol{u}\rangle - \langle \boldsymbol{r}^{(S)}, \boldsymbol{A}\boldsymbol{u}\rangle.$$

Making use of the definition of the residual given in §1.4.2 and assuming that the operator is rank-T —see Eq. (1.20)—, observe that the functional to be optimized can be rewritten

as:

$$\mathcal{J}(\boldsymbol{u}) = \frac{1}{2}\sum_{p=1}^{T}\sum_{t=1}^{T}\langle \boldsymbol{A}^t\boldsymbol{u}, \boldsymbol{A}^p\boldsymbol{u}\rangle - \sum_{p=1}^{T}\sum_{s=1}^{S}\langle \boldsymbol{r}^s, \boldsymbol{A}^p\boldsymbol{u}\rangle.$$

This optimization problem may be solved using the alternating directions algorithm, as usual in the PGD context. In fact, Algorithm 1 can be used with no particular modifications, and therefore the development made in §1.4.2 is not repeated here. An equation analogous to Eq. (1.36) is obtained:

$$\text{Find } \boldsymbol{u}_k \in \mathbb{K}^{N_k}: \quad \langle \tilde{\boldsymbol{A}}_k\boldsymbol{w}, \tilde{\boldsymbol{A}}_k\boldsymbol{u}_k\rangle = \langle \tilde{\boldsymbol{r}}_k, \tilde{\boldsymbol{A}}_k\boldsymbol{w}\rangle, \quad \forall \boldsymbol{w} \in \mathbb{K}^{N_k}. \tag{1.43}$$

It must be noticed that in this case the orthogonality of the projection of the residual is imposed with respect to $\tilde{\boldsymbol{A}}_k\boldsymbol{w}$, for every $\boldsymbol{w} \in \mathbb{K}^{N_k}$. Several remarks about the performance of this algorithm are in order:

- It can be proved that the convergence of this formulation measured in residual norm is monotonic.
- In spite of this, several authors including the one who writes this thesis [15, 29, 106] have experienced that the convergence rate measured by the norm of the true error is in fact very poor:

  $$\|\boldsymbol{f} - \boldsymbol{A}\boldsymbol{u}^{(R)}\| \quad \nsim \quad \|\boldsymbol{u}^{(R)} - \boldsymbol{A}^{-1}\boldsymbol{f}\|.$$

  If the low-rank approximation is built from rank-one corrections, poor convergence rates yield the necessity of computing artificially high-rank representations to attain a certain precision.
- The computational cost is highly increased. If the operator was rank-T, the least-square formulation leads automatically to a rank-T^2 operator.
- The normal operators are full (as opposed to sparse) and their condition number is seriously increased.

The aforementioned disadvantages motivate the introduction of alternative techniques to solve non-adjoint problems.

1.4.3.4 Minimax Proper Generalized Decomposition

In previous sections, we have emphasized the fact of the orthogonality of the residual being imposed in a Galerkin sense. As an alternative to the least-squares Galerkin formulation to treat non self-adjoint problems, the Minimax PGD imposes the orthogonality of the residual with respect to the solution of some auxiliary problem that will be introduced later. This can be interpreted as a Petrov-Galerkin formulation of the PGD.

Suppose that a rank-one correction wants to be computed. Therefore, using barely the same notation as in the previous sections, we introduce the following functional:

$$\mathcal{J}(\boldsymbol{u},\boldsymbol{w}) := \frac{1}{2}\langle \boldsymbol{u},\boldsymbol{u}\rangle - \langle \boldsymbol{A}\boldsymbol{u},\boldsymbol{w}\rangle + \langle \boldsymbol{r}^{(S)},\boldsymbol{w}\rangle,$$

and we define the following optimization *minimax* problem:

$$\max_{\boldsymbol{w}\in \mathrm{vect}(\mathcal{C}_1)} \min_{\boldsymbol{u}\in \mathrm{vect}(\mathcal{C}_1)} \mathcal{J}(\boldsymbol{u},\boldsymbol{w}).$$

The stationarity conditions of the functional are found by calculus of variations. Let us denote such variations as $\boldsymbol{w}+\xi\tilde{\boldsymbol{w}}$ and $\boldsymbol{u}+\eta\tilde{\boldsymbol{u}}$. The stationarity conditions of the functional are:

$$\left.\frac{\partial \mathcal{J}}{\partial \xi}\right|_{\xi,\eta=0} = \langle \boldsymbol{r}^{(S)},\tilde{\boldsymbol{w}}\rangle - \langle \boldsymbol{A}\boldsymbol{u},\tilde{\boldsymbol{w}}\rangle = 0, \qquad \forall \tilde{\boldsymbol{w}} \in \mathrm{vect}(\mathcal{C}_1), \tag{1.44a}$$

$$\left.\frac{\partial \mathcal{J}}{\partial \eta}\right|_{\xi,\eta=0} = \langle \tilde{\boldsymbol{u}},\boldsymbol{u}\rangle - \langle \boldsymbol{A}\tilde{\boldsymbol{u}},\boldsymbol{w}\rangle \equiv \langle \boldsymbol{u},\tilde{\boldsymbol{u}}\rangle - \langle \boldsymbol{A}^H\boldsymbol{w},\tilde{\boldsymbol{u}}\rangle = 0, \qquad \forall \tilde{\boldsymbol{u}} \in \mathrm{vect}(\mathcal{C}_1). \tag{1.44b}$$

Eq. (1.44a) states that the orthogonality of the residual is imposed with respect to, in principle, every $\tilde{\boldsymbol{w}}$. However we shall constrain it to be the solution of Eq. (1.44b), denoted by $\boldsymbol{w}$, and thus it can be interpreted as a Lagrange multiplier. Therefore, we can reformulate the problem as follows:

$$\text{Find } \boldsymbol{u},\boldsymbol{w} \in \mathrm{vect}(\mathcal{C}_1) \text{ that solve : } \begin{cases} \langle \boldsymbol{A}\boldsymbol{u},\boldsymbol{w}\rangle = \langle \boldsymbol{r}^{(S)},\boldsymbol{w}\rangle \\ \qquad\qquad \text{and} \\ \langle \boldsymbol{A}^H\boldsymbol{w},\tilde{\boldsymbol{u}}\rangle = \langle \boldsymbol{u},\tilde{\boldsymbol{u}}\rangle, \quad \forall \tilde{\boldsymbol{u}} \in \mathrm{vect}(\mathcal{C}_1). \end{cases} \tag{1.45}$$

Recall that our interest in writing the problem in optimization form is due to we do not want to solve Eq. (1.45) directly because it is too expensive: $|\mathrm{vect}(\mathcal{C}_1)| = \mathcal{N}$. Optimization algorithms, such as alternating directions allow solving the problem with much less computational effort. We shall spend next lines to recast the Minimax problem already defined into the alternating directions methodology. Since the objective is to optimize one direction at a time, let us assume that

$$\boldsymbol{u} = \otimes_{d=1}^{D} \boldsymbol{u}_d \quad \text{and} \quad \boldsymbol{w} = \otimes_{d=1}^{D} \boldsymbol{w}_d$$

are known except $\boldsymbol{u}_k$ and $\boldsymbol{w}_k$, for some $1 \leq k \leq D$. Thanks to the separation property of the scalar product, every inner product in the functional $\mathcal{J}(\boldsymbol{u},\boldsymbol{w})$ can be computed except those related to dimension k. Therefore, the optimization problem is reduced to:

$$\max_{\boldsymbol{w}_k\in\mathbb{K}^{N_k}} \min_{\boldsymbol{u}_k\in\mathbb{K}^{N_k}} \mathcal{J}(\boldsymbol{u}_k,\boldsymbol{w}_k).$$

The pair $(\boldsymbol{u}_k, \boldsymbol{w}_k)$ can be updated from solving the *projected* version of Eq. (1.45), which can be written as follows:

$$\text{Find } \boldsymbol{u}_k, \boldsymbol{w}_k \in \mathbb{K}^{N_k} \text{ that solve : } \begin{cases} \langle \tilde{\boldsymbol{A}}_k \boldsymbol{u}_k, \boldsymbol{w}_k \rangle = \langle \tilde{\boldsymbol{r}}_k, \boldsymbol{w}_k \rangle \\ \text{and} \\ \langle \tilde{\boldsymbol{A}}_k^H \boldsymbol{w}_k, \tilde{\boldsymbol{u}} \rangle = \langle \boldsymbol{u}_k, \tilde{\boldsymbol{u}} \rangle, \quad \forall \tilde{\boldsymbol{u}} \in \mathbb{K}^{N_k}, \end{cases} \tag{1.46}$$

where $\tilde{\boldsymbol{A}}_k$ and $\tilde{\boldsymbol{r}}_k$ are computed as shown in Eq. (1.37), but with a different definition of scalars α_t and β_s, see Eq. (1.35). These coefficients are redefined as follows:

$$\alpha_t = \prod_{d=1, d\neq k}^{D} \langle \boldsymbol{A}_d^t \boldsymbol{u}_d, \boldsymbol{w}_d \rangle, \quad 1 \leq k \leq T \quad \text{and} \quad \beta_s = \prod_{d=1, d\neq k}^{D} \langle \boldsymbol{r}_d^s, \boldsymbol{w}_d \rangle, \quad 1 \leq s \leq S,$$

where we wish to recall that T and S are the rank of the operator and rank of the residual, respectively. See §1.4.2 for more details.

Although the Minimax algorithm has been developed here for the rank-one corrections case, it would take not much conceptual effort to introduce a Optimal Petrov-Galerkin PGD in which, for each direction, every mode is updated at the same time, as we did in §1.4.3.1.

Algorithm 4 Rank-one corrections Minimax PGD

Require: $\boldsymbol{u}^{(R)}$ ▷ Rank-R approximation
1: Initialize $\boldsymbol{u}$ and $\boldsymbol{w}$
2: **for** $i = 1$ to I **do**
3: Set $\boldsymbol{u}_{\text{old}} = \boldsymbol{u}$ and $\boldsymbol{w}_{\text{old}} = \boldsymbol{w}$
4: **for** $k = 1$ to D **do**
5: Update $\boldsymbol{u}_k$ and $\boldsymbol{w}_k$ ▷ Use Eq. (1.46)
6: **end for**
7: Check $\boldsymbol{u} - \boldsymbol{u}_{\text{old}}$ and $\boldsymbol{w} - \boldsymbol{w}_{\text{old}}$ ▷ Eq. (1.38) applies for $\boldsymbol{w}$ also
8: **end for**
9: **return** $\boldsymbol{u}$ ▷ Rank-one correction

1.5 Current limitations of Proper Generalized Decomposition

Although PGD has been successfully applied to a considerably wide variety of problems in several fields of physics and engineering, there are still several issues that have not been completely solved yet. Among the applications, we can cite the Fokker-Planck equation [7], rheology and complex flows [1, 8, 9], multiscale models and homogeneization [36, 37], optimization of thermal problems [39, 113], real-time simulation of soft tissues with surgical

applications [59, 114], Dynamic Data-Driven Application Systems (DDDAS) [57, 96], solid mechanics [11, 27, 28, 60], parametrized geometries [6] or the Chemical Master Equation [34].

This section is an attempt to provide a critical review on PGD as a means to establish what are the current major limitations of this method. Of course, limitations should be also understood as challenges that partially motivate this work and future research. Among them, nonlinear problems are briefly considered here for the first time in this work. Nonlinear problems are difficult and in general they cannot be treated in an uniform manner: each nonlinear problem requires its own appropriate numerical techniques. In spite of that, nonlinear problems treated in the PGD framework share some difficulties concerning separability that motivate the Chapter 4.

1.5.1 Solution separability and compactness of separated representations

The solution of a certain problem formulated in tensor product spaces is said to be *separable* if it can be approximated with arbitrary precision using low-rank separated representations. On the contrary case, the solution is said to be *bad-separable*, or simply *non-separable* by abuse of language. Roughly, the rank of the separated representation must fulfill the following requirement to achieve some compression of data:

$$R \leq \left\lfloor \frac{\prod_{d=1}^{D} N_d}{\sum_{d=1}^{D} N_d} \right\rfloor,$$

where $\lfloor \cdot \rfloor$ denotes the floor function. If not, the separated representation would be nearly as expensive as the explicit one.

In two dimensions, SVD can be used to determine whether a separated representation can be further compressed or not, and thus reveal some kind of suboptimality. However, in some cases not even SVD is much effective and relatively high-rank representations are obtained. This has been observed in scattering Helmholtz problems which yield a complex behaviour due to its multiscale nature and need of very fine discretizations to capture all wave propagation details [97]. Such cases are characterized by a sort of *intrinsic* non-separability against which, to the author's knowledge, there are not much remedies.

In higher dimensions, determining the canonical rank of a tensor is a NP-hard problem and therefore not much can be said in general about separability[9]. Is it the solution really non-separable or is it the algorithm (PGD or another one) which is the source of inefficiency? An attempt to measure somehow if a rank-R separated representation $\boldsymbol{u}^{(R)}$ could be further compressed is to apply a PGD projection step. Projection can be formulated the same as

[9] Observe that a tensor solution of a problem admits a Tucker or Hierarchical Tucker representation (hopefully of low rank), but nothing can be said about the rank of the equivalent canonical representation.

classical PGD: rank-one corrections computed by means of an alternating directions algorithm. To keep coherence with §1.4, the PGD projection can be formulated in terms of an optimization problem as follows:

$$\min_{\boldsymbol{v}\in\mathrm{vect}(\mathcal{C}_1)} \mathcal{J}(\boldsymbol{v}) := \frac{1}{2}\langle \boldsymbol{v}, \boldsymbol{v}\rangle - \langle \boldsymbol{u}^{(R)} - \boldsymbol{v}^{(S)}, \boldsymbol{v}\rangle,$$

where $\boldsymbol{v}$ is a rank-one correction and $\boldsymbol{v}^{(S)}$ is a rank-S compacted separated representation. If at convergence (evaluated from the norm of the residual), the rank is $R^* < R$, it probably means that there is some kind of inefficiency on the formulation of the problem. This is classically found in two contexts:

- When alternating directions iterations are cut-off prematurely, the resulting separated representation may eventually be further compressed.

- When non self-adjoint problems are of interest, any PGD formulation (Galerkin, Minimal Residual, Petrov-Galerkin...) leads in general to separated representations that are clearly sub-optimal and can be compacted using the PGD projection.

- When coupling PGD and some nonlinear iterative scheme, the rank of the separated representation may be driven by the convergence of the nonlinear scheme. Up to this point, nonlinear problems have been completely ignored but when limitations of PGD are to be considered, nonlinear problems cannot be omitted. Here we analyse briefly some difficulties associated to nonlinear problems, which are further developed in Chapter 4.

If $R^* = R$, it cannot be claimed that the original separated representation was optimal but, at least, it can be said that it was the *best* separated representation attainable by a rank-one PGD constructor. It is worth to remark that the notion of *best* attainable separated representation refers to the algorithm used to compute the PGD projection. That is, projection could be formulated analogously to an Optimal Galerkin PGD. Therefore, a separated representation may be the best one with respect to the rank-one PGD but not with respect to Optimal Galerkin PGD. Nevertheless, it remains as an useful measure to evaluate the performance of the algorithm.

1.5.2 Algorithmic inefficiencies and computational aspects

The compactness of the solution and the computational cost are in general mutually exclusive concepts. A compromise must be found depending on the available ressources and the application requirements. Regarding the classic rank-one PGD, the following principal sources of computational inefficiency are found:

- Slow convergence, or even divergence, of the Alternating Directions algorithm due to a bad definition of the optimization problem. This is classically observed when rank-one Galerkin corrections are applied to non self-adjoint problems. The optimization problem is not well defined and therefore there is no guarantee about the convergence of the alternating iterations.

- Slow convergence of the Alternating Directions algorithm due to bad conditioning of the operators. Iterative methods are sensitive to the condition number and alternating directions is not an exception. This is classically observed in Minimal Residual Galerkin formulations [15, 16].

- Slow convergence of the Alternating Directions algorithm in dimensions higher than two. When the algorithm alternates between the solution of several low-dimensional problems, how many extra iterations are needed to converge? To the author's knowledge, there are not much results on this issue.

- Heavy iterations due to separated representations of rank unnecessarily high. Observe that if the operator is rank-T, each rank-one correction makes the right-hand side to increase its rank by T. This implies that at each alternating iteration and for each dimension, the number of integrals to be computed increases by T. To give an order of magnitude, a three-dimensional damped structural dynamics problem such as the one addressed in Chapter 2 leads to $T \approx 40$. This serves to show that achieving compact representations is not only important in terms of storage requirements but also from a computational point of view.

1.5.3 Affine decomposition of the operator and problem data separability

In §1.1.3.3, two hypothesis were made about the existence of both an affine decomposition —equivalently, separated representation— of the operator and the separated representation of the right-hand side. They are of crucial importance because only under these hypothesis the problem has canonical tensor structure. Recall that PGD takes advantage of the tensor structure to split the high-dimensional problem into a series of low-dimensional problems. Low-dimensional problems are only coupled through coefficients computed from inner products —i.e. integrals— in low-dimensions. If the hypothesis are not fulfilled, the inner product separation property is useless and computing the coupling coefficients requires performing integrals in $D-1$ dimensional domains, which ruins the computational performance of PGD.

Fortunately, it is not difficult in general to find an affine decomposition of a linear differential operator. There are however at least two exceptions:

- The coefficients of the PDE depend on the coordinates of the problem and they cannot be trivially expressed in separated representation. This can be clearly illustrated by means of [97], which considers a wave propagation problem. The objective is therefore to compute a separated representation including the physical coordinates —space— and both the wave number and the angle of incidence. However, the coefficients of the Helmholtz equation depend on the space and the wave number and the right-hand side depends on space, the wave number and the angle of incidence. This dependence is such that it is not obvious to derive a separated representation in terms of those coordinates. A possible approach consists in performing a previous PGD projection step to separate the aforementioned quantities, see §1.5.1. Projection is made only once.

- There is a nonlinear term in the PDE. It may be a nonlinear coefficient, such as a diffusion coefficient depending on temperature, or a nonlinear differential operator by itself. Very much like in the previous item, PGD needs the nonlinear term to be written in the same separated format as the solution. Note that, disregarding the particular nonlinear scheme chosen, at some point of the implementation the nonlinear term will need to be evaluated at the previous iterate of the solution (thus known). The question is how to evaluate the nonlinear term in separated form. Observe that even if the previous iterate is known in separated form, the nonlinear term cannot be written in separated format trivially. PGD projection at each nonlinear iteration, although feasible, destroys the computational performance. Other alternatives must therefore be sought. We refer to Chapter 4 for more details.

By problem data separability we mean that the right-hand side of the equation, the boundary conditions or any other quantity has to be rewritten as a separated representation. Depending on their nature, it may be easy or not. Let us illustrate it by means of a 1D domain in which the transient heat equation wants to be solved, and a right-hand side represented by a wave $f(x-ct)$ that travels at speed c. Suppose that a space-time separated representation wants to be computed. Observe that in the space-time domain, the wave does not admit a separated representation. Therefore, a PGD projection is not an efficient solution for this kind of problems and new strategies are needed.

1.6 Scope of the thesis

Among the three general topics of research identified in §1.5 (solution compactness, algorithmic performances and data separability), this work is mainly concerned by those considered in §1.5.3. The aim of this thesis is therefore to introduce some non-conventional strategies to address the problem data separability.

In particular, we present the following three main contributions:

- We introduce space-frequency separated representations for transient problems in Chapter 2 (structural dynamics) and Chapter 3 (heat transfer). To this end, the equations are solved in the frequency domain by considering the frequency as an extra coordinate. This allows defining a sort of *Generalized Transfer Functions* (GTF) that encode the response of the system in a frequency band. Since any input signal (excitation) can be Fourier transformed under rather weak conditions, the response of the system can be recovered from the GTF by simple superposition.

- In Chapter 3 we present a technique that allows overcoming the separability issues due to external moving excitations. We show that the transient heat equation becomes symmetric although still non-hermitian in the frequency domain. The symmetry is exploited to prove that the Reciprocity Principle applies, yielding a method that allows monitoring system's behaviour in real time. In this way, the intrinsic non-separability of a traveling excitation wave along the boundary is solved.

- In Chapter 4 we introduce an a priori Empirical Interpolation technique for the separability of arbitrary multivariate functions.

Chapter 2

Space-frequency separated representations in structural dynamics

This chapter is concerned with the solution of structural dynamics equations. The technique here presented is closely related to Harmonic Analysis, and therefore it is only concerned with the long-term forced response. PGD is used to compute space-frequency separated representations by considering the frequency as an extra coordinate. This formulation constitutes an alternative to classical methods such as Modal Analysis and it is specially advantageous when parametrized structural dynamics equations are of interest. In such case, there is no need to solve the parametrized eigenvalue problem, the space-time solution can be recovered with a Fourier inverse transform, the PGD solution is valid for any forcing term that can be written as a combination of the considered frequencies and, finally, the solution is available for any value of the parameter.

Contents

2.1 An overview on classic structural dynamics

The frequency formulation of structural dynamics is a powerful approach to study the response of structural systems when initial conditions can be neglected; that is, far enough from the initial transient response. This framework, which was integrated into the finite element framework from the very beginning, has been, and still is, extensively used. It is well described in most textbooks, some of them linked to many generations of scientists and engineers [19, 41, 71, 137]. This approach is even now an active research area because several challenges are still open.

The application of space-frequency separated representations was considered in [17, 48, 81, 82, 84, 86, 117, 118, 119] for the so-called: *variational theory of complex rays*. Obviously, there have been many attempts considering such descriptions within the model reduction framework; we refer to [68] and the references therein.

Since structural dynamics have been studied in many scientific articles and textbooks, some of them mentioned above, it is not the aim of the author to provide an exhaustive literature review. We simply consider the matrix differential equations of motion,

$$\boldsymbol{M}\ddot{\boldsymbol{u}}(t) + \boldsymbol{C}\dot{\boldsymbol{u}}(t) + \boldsymbol{K}\boldsymbol{u}(t) = \boldsymbol{f}(t), \tag{2.1}$$

disregarding the physical system (discretized continuous or intrinsically discrete) they represent. To set the notation, let $\boldsymbol{M}, \boldsymbol{C}, \boldsymbol{K} \in \mathbb{R}^{N\times N}$ be the mass, damping and stiffness matrices, respectively, while $\boldsymbol{f}(t) \in \mathbb{R}^N$ is the generalized force vector and $\boldsymbol{u}(t) \in \mathbb{R}^N$ is generalized displacement vector. The time interval in which the motion of the structure is of interest is denoted by $I_t = [0, T]$. Appropriate initial and boundary conditions must be provided.

Eq. (2.1) may represent, for instance, the Finite Element (FE) discretization of a linear viscoelastic continuum solid that occupies the open bounded domain $I \subset \mathbb{R}^{d\leq 3}$. In that case, the generalized forces are the static equivalent nodal forces, while the generalized displacements are simply the nodal displacements. For plate and beam shaped solids, it may be appropriate to introduce some kinematic assumptions that allow deriving classic plate and beam theories from the continuum model. In that case, the generalized force vector includes resultant forces and moments, while the generalized displacement vector contains displacements and rotations. However, Eq. (2.1) remains formally unchanged, even if the system is purely discrete, i.e. there is no notion of continuum, like in a mass-spring-damper network.

2.1.1 Proportional and non proportional damping

Although the equations of motion remain invariable for the aforementioned cases, their physical meaning can be of a very different nature. In solid mechanics, the damping term usually comes from the constitutive relation and represents the internal friction due to the

viscous behaviour of the material. However, in structural mechanics it may be of interest to introduce a *structural damping* [137], independent to that coming from the material constitutive relation. This kind of damping can be introduced to model either a mechanical device (e.g. automotive applications, seismic vibration control) or some diffuse damping effects due to contacts between structural parts. In the first case, it should be possible to characterize the behaviour of the dispositive. In the second case, empirical approaches are traditionally used. Among them, *proportional damping* [126] supposes that damping is a linear combination of the mass and stiffness matrices. This is equivalent to assume that damping is distributed over the structure in the same way as the mass and stiffness matrices. The combination coefficients are to be determined experimentally.

Proportional damping is indeed a mathematical artifice that provides great advantages in the context of Modal Analysis, since Eq. (2.1) becomes diagonal when it is projected onto the subspace formed by the eigenvectors. In other words, the N-size system is transformed into N uncoupled ODEs. Eigenvectors are computed by solving the undamped Generalized Eigenvalue Problem (GEP), i.e. $\boldsymbol{C} \equiv 0$ and $\boldsymbol{f}(t) \equiv 0$ in Eq. (2.1). However, assuming a proportional damping is quite restrictive and may yield to unsatisfactory results. An accurate consideration of damping, disregarding its material, mechanical or diffuse nature, needs of considering more general models, not necessarily proportional. In that case, two options are possible:

1. The problem can still be projected onto the eigenvectors subspace but, since the damping matrix does not necessarily satisfy any orthogonality condition with respect to them, the problem does not transform into diagonal. Hence, the computational cost of solving the projected or the original ODE system is roughly the same, unless the eigenbasis is *reduced* by truncation.

2. Alternatively, the eigenvectors can be computed from the unforced damped equations, i.e. $\boldsymbol{f}(t) \equiv 0$ in Eq. (2.1), to decouple the ODE system and use Modal Analysis efficiently. Unfortunately this is not a GEP anymore but a Quadratic Eigenvalue Problem (QEP) instead [126]. QEP is a particular type of nonlinear eigenvalue problem considerably harder to solve. The second order ODE system of size $N \times N$ needs to be reformulated into a $2N \times 2N$ first order system, and therefore the computational cost scales accordingly. The projected system is diagonal of size $2N$, unless it is truncated.

In consequence, Modal Analysis, which is very attractive when proportionally damped structures are of interest, becomes less efficient when non-proportionally damped structures are considered.

2.1.2 Parallelism between Modal Analysis and Model Order Reduction

Structural dynamics is a pioneering field in which these techniques were early applied. In fact, there is a strong parallelism between Modal Analysis and Model Order Reduction (MOR). Let us recall briefly that the objective of most of *a posteriori* MOR techniques is to build a *reduced* model from the *full* one, which in our case corresponds to Eq. (2.1). The reduced model is constructed by performing a (Galerkin) projection of the full model onto a subspace previously constructed from several solves of the full model. The reduction in complexity depends on the dimension of the subspace. In general, truncation is needed to achieve such reduction in complexity. Many works applying MOR to structural dynamics have have already been published, see [55, 69, 79, 133] for instance. As it can be appreciated, Modal Analysis performs very much in the same way, with two significative differences. First is that the subspace is built from the eigenvectors instead of from snapshots. And second is that the computational complexity is reduced without introducing any error, provided that the conditions that make the projected system to be diagonal are respected and all eigenvectors are kept. Apart from these two precisions, *a posteriori* MOR techniques and Modal Analysis share many features.

2.1.3 Parametrized structural dynamics equations

Let us consider now a parametrized structural dynamics model. For some value of the parameter in the interval of interest, $\mu \in I_\mu$, we write:

$$\boldsymbol{M}\ddot{\boldsymbol{u}}(t) + \boldsymbol{C}(\mu)\dot{\boldsymbol{u}}(t) + \boldsymbol{K}\boldsymbol{u}(t) = \boldsymbol{f}(t), \tag{2.2}$$

where the parametric dependence has been assigned only to the damping operator without loss of generality. If Modal Analysis wants to be applied properly to solve Eq. (2.2), each time the parameter changes, a nonlinear (quadratic) eigenvalue problem must be solved to find the eigenvectors. The reader will understand at this point why parametric problems, and specially parametric non-proportionally damped problems, are computationally expensive to solve using Modal Analysis. It is worth to recall that, of course, it is always possible to solve Eq. (2.1) using a time-integration scheme. This procedure, although conceptually simple, is computationally expensive for systems involving a large number of degrees of freedom. Besides, it does not provide any insight into the resonant behaviour of the system.

The interest of solving parametric problems does not need much motivation. Consider a part or structure that wants to be optimized with respect to its dynamical behaviour. An optimization algorithm could be coupled to a structural dynamics model to predict the behaviour of the system each time one parameter changes. Thus, many solves of the equations of motion are required. This *multi-query* context constitutes a prototypical application of MOR techniques. The procedure proposed in [115] is as follows: first, the QEP is repeatedly

solved for a parameter sampling $\mu_1, \ldots, \mu_s \in I_\mu$. The eigenvectors of all these problems are collected in a big matrix. By means of a Singular Value Decomposition (SVD), the left singular vectors associated to the largest singular values are kept and used as a subspace in which the reduced model can be projected. Although the left singular vectors are orthogonal each other, they are no longer eigenvectors, so that the reduced system is not diagonal.

Most of the references on MOR given so far concern *a posteriori* techniques, i.e. the reduced model can only be built after several solves of the full model. However it has been shown in the Chapter 1 that *a priori* methods, such as the one this work is concerned with, allow computing a reduced approximation with no information in advance. In this Chapter, we propose a technique based on the Proper Generalized Decomposition (PGD) to compute space-frequency separated representations, see §2.4.

2.2 Approaches based on Modal Analysis

In this section we shall review in detail different approaches based on Modal Analysis for both damped and non-proportionally damped problems.

2.2.1 Free and forced responses of undamped and proportionally damped problems

Consider the unforced, undamped equations of motion, i.e. $\boldsymbol{C} \equiv 0$ and $\boldsymbol{f}(t) \equiv 0$ in Eq. (2.1). The general solution of such homogeneous second order ODE can be written as a linear combination of eigenvectors of a Generalized Eigenvalue Problem (GEP) defined in Eq. (2.4):

$$\boldsymbol{u}(t) = \boldsymbol{E} \exp(\mathrm{i}\boldsymbol{\Lambda} t)\boldsymbol{\alpha}, \tag{2.3}$$
$$\text{with} \quad \boldsymbol{E} = [\boldsymbol{e}^1 \boldsymbol{e}^2 \cdots \boldsymbol{e}^N] \in \mathbb{R}^{N\times N}, \quad \boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n) \in \mathbb{R}^{N\times N} \quad \text{and} \quad \boldsymbol{\alpha} \in \mathbb{R}^N,$$

where $\boldsymbol{e}^{1\leq i\leq N}$ are the eigenvectors, $\lambda_{1\leq i\leq N}$ are the eigenvalues and $\boldsymbol{\alpha}$ is a vector of arbitrary constants. The eigenvectors and the eigenvalues are real-valued under rather weak conditions (symmetry, positive definiteness) for mass and stiffness matrices, coming for instance from a FE discretization. The imaginary unit is denoted by "i" (no italic type), and should not be confused with index "i" (italic type). Replacing the general solution gives place to the following Generalized Eigenvalue Problem (GEP):

$$\boldsymbol{K}\boldsymbol{E} = \boldsymbol{M}\boldsymbol{E}\boldsymbol{\Lambda}^2. \tag{2.4}$$

If the eigenvectors are normalized with respect to the mass matrix, then the following properties of orthogonality apply:

$$\boldsymbol{E}^T\boldsymbol{M}\boldsymbol{E} = \boldsymbol{I} \quad \text{and} \quad \boldsymbol{E}^T\boldsymbol{K}\boldsymbol{E} = \boldsymbol{\Lambda}^2. \tag{2.5}$$

Now, if forced equations of motion want to be solved using modal superposition, we require the solution to live in a subset of the eigenspace, i.e.

$$\forall t \in I_t : \quad \boldsymbol{u}(t) \in \boldsymbol{\mathcal{E}}_m := \text{span}\{\boldsymbol{e}^{1 \leq i \leq m \leq N}\} \subset \boldsymbol{\mathcal{E}}_N \quad \Leftrightarrow \quad \boldsymbol{u}(t) = \boldsymbol{E}\boldsymbol{\alpha}(t) \in \boldsymbol{\mathcal{E}}_N.$$

Performing a Galerkin projection and recalling the orthogonality conditions stated in Eq. (2.5), the reduced model is obtained:

$$\ddot{\boldsymbol{\alpha}}(t) + \boldsymbol{\Lambda}^2 \boldsymbol{\alpha}(t) = \boldsymbol{\beta}(t), \tag{2.6}$$

where $\forall t \in I_t : \boldsymbol{\beta}(t) = \boldsymbol{E}^T \boldsymbol{f}(t)$ is the projection of the generalized force vector on $\boldsymbol{\mathcal{E}}_m$. If $m < N$, then $\boldsymbol{u}(t)$ can only be approximated since some components of the forcing term may not be well captured, introducing a truncation error. In return, the computationally is further reduced as less equations have to be solved. It is worth to recall at this point that MOR techniques such as POD or Reduced Basis achieve a reduction of the computational complexity precisely thanks to truncation by properly choosing a subspace of smaller dimension. Eq. (2.6) can be solved inexpensively by direct time integration since it is diagonal. If a proportional damping is used, it is possible to keep the same procedure and, at the end, the projected equations are still diagonal. Consider one particular type of proportional damping —see [126] for more general options—, the *Rayleigh damping* [41, 136]:

$$\boldsymbol{C} = a\boldsymbol{M} + b\boldsymbol{K},$$

where coefficients a and b usually are to be determined experimentally. Then, the reduced model system writes:

$$\ddot{\boldsymbol{\alpha}}(t) + (a\boldsymbol{I} + b\boldsymbol{\Lambda}^2)\dot{\boldsymbol{\alpha}}(t) + \boldsymbol{\Lambda}^2 \boldsymbol{\alpha}(t) = \boldsymbol{\beta}(t).$$

For the more general case in which damping is not proportional, orthogonality of the eigenvectors, computed from Eq. (2.4), with respect to the damping matrix does not hold. This means that when projecting, the problem does not become diagonal and therefore the computational cost is roughly the same as if the equations were directly time-integrated.

2.2.2 Free and forced response of non proportionally damped problems

In order to decouple the equations of motion in presence of a non proportional damping, it is necessary to solve a Quadratic Eigenvalue Problem (QEP) [126]. This is an important class of nonlinear eigenvalue problems. In the most general case, the QEP consists in finding eigenvalues λ, right eigenvectors $\boldsymbol{e}_r$ and left eigenvectors $\boldsymbol{e}_l$ satisfying

$$\begin{aligned} &\boldsymbol{A}(\lambda)\boldsymbol{e}_r = \boldsymbol{0} \quad \text{and} \quad \boldsymbol{e}_l^H \boldsymbol{A}(\lambda) = \boldsymbol{0}, \\ &\text{with} \quad \boldsymbol{A}(\lambda) = \lambda^2 \boldsymbol{M} + \lambda \boldsymbol{C} + \boldsymbol{K}. \end{aligned} \tag{2.7}$$

The symbol "$\bullet^H$" stands for the complex conjugate transpose, or Hermitian transpose. The QEP has $2N$ eigenvalues and up to $2N$ right eigenvectors and $2N$ left eigenvectors. It is worth to recall that if there are more than N eigenvectors, they do not form a linear independent set. For real-valued matrices $\boldsymbol{M}$, $\boldsymbol{C}$ and $\boldsymbol{K}$, the following simplifications hold:

1. The eigenvalues are real or come in pairs $(\lambda, \bar{\lambda})$. The symbol "$\bar{\bullet}$" stands for the complex conjugate.

2. If $\boldsymbol{e}_r$ is a right eigenvector of λ, then $\bar{\boldsymbol{e}}_r$ is also a right eigenvector of $\bar{\lambda}$.

Additionally, if matrices are symmetric, the left and right sets of eigenvectors coincide. Although for most structural applications the matrices are symmetric positive (semi) definite, we are not taking this into account in order to keep the exposition as general as possible. For further details on QEP, see [126].

Consider the unforced, damped equations of motion, i.e. $\boldsymbol{f}(t) \equiv 0$ in Eq. (2.1). This clearly defines a QEP. Several methods are available to solve this class of problem. One option, possibly the easiest, is to rewrite the homogenous second order ODE system as a first order system of $2N$ equations:

$$\boldsymbol{P}\dot{\boldsymbol{v}}(t) + \boldsymbol{Q}\boldsymbol{v}(t) = \boldsymbol{0},$$
$$\text{with} \quad \boldsymbol{P} = \begin{bmatrix} -\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{M} \end{bmatrix}, \quad \boldsymbol{Q} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{K} & \boldsymbol{C} \end{bmatrix}, \quad \text{and} \quad \boldsymbol{v}(t) = \begin{bmatrix} \boldsymbol{u}(t) \\ \dot{\boldsymbol{u}}(t) \end{bmatrix}. \tag{2.8}$$

The general solution of a first order ODE system such as Eq. (2.8) can be written as a linear combination eigenvectors coming from a GEP defined in Eq. (2.9):

$$\boldsymbol{v}(t) = \boldsymbol{V} \exp(\boldsymbol{\Sigma} t)\boldsymbol{\alpha},$$
$$\text{with} \quad \boldsymbol{V} = [\boldsymbol{v}^1 \boldsymbol{v}^2 \dots \boldsymbol{v}^{2N}] \in \mathbb{C}^{2N \times 2N}, \quad \boldsymbol{\Sigma} = \operatorname{diag}(\sigma_1, \sigma_2, \dots, \sigma_{2N}) \in \mathbb{C}^{2N \times 2N} \quad \text{and} \quad \boldsymbol{\alpha} \in \mathbb{R}^{2N}.$$

On substitution, a GEP such as Eq. (2.4), of twice the size of the original problem is obtained:

$$\boldsymbol{Q}\boldsymbol{V} = \boldsymbol{P}\boldsymbol{V}\boldsymbol{\Sigma}. \tag{2.9}$$

Both eigenvectors and eigenvalues are complex-valued since matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ are not symmetric. The eigenvectors are orthogonal with respect to those matrices, and if they are normalized with respect to $\boldsymbol{P}$, then we can write:

$$\boldsymbol{V}^T\boldsymbol{P}\boldsymbol{V} = \boldsymbol{I} \quad \text{and} \quad \boldsymbol{V}^T\boldsymbol{Q}\boldsymbol{V} = \boldsymbol{\Sigma}. \tag{2.10}$$

Now, if forced equations of motion want to be solved using modal superposition, we proceed very much as discussed in Section 2.2.1, i.e.

1. Write the solution, at each time, as a linear combination of eigenvectors:

$$\forall t \in I_t : \quad \boldsymbol{v}(t) = \boldsymbol{V}\boldsymbol{\alpha}(t). \tag{2.11}$$

2. Perform Galerkin projection to obtain a decoupled system of equations (recall orthogonality conditions stated in Eq. (2.10)):

$$\dot{\boldsymbol{\alpha}}(t) + \boldsymbol{\Sigma}\boldsymbol{\alpha}(t) = \boldsymbol{\gamma}(t), \quad \text{with} \quad \boldsymbol{\gamma}(t) = \boldsymbol{V}^T \begin{bmatrix} \mathbf{0} \\ \boldsymbol{f}(t) \end{bmatrix} \tag{2.12}$$

3. Solve Eq. (2.12) inexpensively by direct time integration, since it is diagonal. However, it must be noticed that $2N$ equations instead of N are to be solved in this case. Fortunately, in practice the dynamic response is dominated by few eigenvalues near to the real axis and therefore it is not needed to keep all the eigenvectors.

Finally, the solution $\boldsymbol{u}(t)$ can be recovered from $\boldsymbol{v}(t)$ as follows:

$$\boldsymbol{u}(t) = [\boldsymbol{I} \quad \mathbf{0}]\, \boldsymbol{V}\boldsymbol{\alpha}(t). \tag{2.13}$$

Remark 2.1 (*Stability considerations*). From the analysis of the eigenvalues of the QEP problem we can deduce the stability of the system. Hence, it is said to be stable if the real part of every eigenvalue is negative, i.e. $\mathrm{Re}(\sigma_i) < 0$. By stable, we mean that every solution decreases exponentially to zero as $t \to \infty$. If every eigenvalue $\mathrm{Re}(\sigma_i) \leq 0$ and those which $\mathrm{Re}(\sigma_i) = 0$ are semi-simple (complex-conjugated pairs), the system is said to be weakly stable in the sense that every solution is bounded as $t \to \infty$. Of course, the semi-simple pairs real part equal to zero correspond to harmonic oscillations.

In conclusion, it has been shown that:

1. Modal Analysis is computationally efficient, as it allows transforming the ODE system into diagonal, thus uncoupled, equations.

2. It provides highly valuable information on the model stability.

With some remarks to be taken into account when non proportionally damped systems are considered.

2.2.3 Damped parametric problems

Consider now a parametric model such as the one introduced in Eq. (2.2). Solving such dynamic model means finding the dynamic response for each $\mu \in I_\mu$, that is $\boldsymbol{u}(t, \mu)$. Modal

Analysis can be used, of course, to solve this problem for each value of the parameter. To this end, Eq. (2.8) is rewritten taking into account the parametric dependence:

$$\boldsymbol{P}\dot{\boldsymbol{v}}(t) + \boldsymbol{Q}(\mu)\boldsymbol{v}(t) = \boldsymbol{0}, \tag{2.14}$$

where $\boldsymbol{Q}(\mu)$ depends on the parameter via the damping matrix (recall Eq. (2.2)). Therefore for each value of the parameter we will have a different set of eigenvectors, $\boldsymbol{V}(\mu)$. The solution is sought as:

$$\boldsymbol{v}(t,\mu) = \boldsymbol{V}(\mu)\boldsymbol{\alpha}(t). \tag{2.15}$$

Hence, the question is how to compute $\boldsymbol{V}(\mu)$ not for a particular parameter value as explained in Section 2.2.2, but for all of them. Afterwards, $\boldsymbol{\alpha}(t)$ could be readily computed by integrating a diagonal system of equations.

In what follows, we are exploring the possibility of applying Model Order Reduction (MOR) techniques to compute an approximation of $\boldsymbol{V}(\mu)$, not necessarily the eigenbasis. A first approach based on POD is presented in [115]. It proceeds as follows:

1. Choose a sampling $\mathcal{P} = \{\mu_1, \dots, \mu_s\} \subset I_\mu$.
2. Compute $\boldsymbol{V}(\mu_1), \dots, \boldsymbol{V}(\mu_s)$ as explained in Section 2.2.2 and collect them in the sampling matrix $\tilde{\boldsymbol{V}} \in \mathbb{C}^{2N \times ms}$, assuming $m \leq 2N$ eigenvectors are kept for each sample.
3. Compute the left $\boldsymbol{X} \in \mathbb{C}^{2N \times \ell}$ and right $\boldsymbol{W} \in \mathbb{C}^{ms \times \ell}$ singular vectors of the sampling matrix by performing a Singular Value Decomposition (SVD), such that $\tilde{\boldsymbol{V}} = \boldsymbol{X}\boldsymbol{W}^H$. Of course, $\ell \leq \min(2N, ms)$, and in case of strict inequality, a POD truncation error is introduced.

Observe that any column of the sampling matrix can be written as a linear combination of left singular vectors, i.e.

$$\tilde{\boldsymbol{v}}^i = \boldsymbol{X}\boldsymbol{\psi} \quad \text{with} \quad \boldsymbol{\psi} = \boldsymbol{w}_i^H,$$

where $\tilde{\boldsymbol{v}}^i$ is the i-th column of $\tilde{\boldsymbol{V}}$ and $\boldsymbol{w}_i$ is the i-th row of $\boldsymbol{W}$. Equivalently, each set of eigenvectors $k \leq s$, associated to the parameter value μ_k, can be written as a linear combination of the left singular vectors:

$$\boldsymbol{V}(\mu_k) = \tilde{\boldsymbol{V}}^{\xi_1 \leq i \leq \xi_2} = \boldsymbol{X}\boldsymbol{\Psi} \quad \text{with} \quad \boldsymbol{\Psi} = \boldsymbol{W}^H_{\xi_1 \leq i \leq \xi_2},$$

with $\xi_1 = m(k-1)+1$ and $\xi_2 = mk$. Therefore we can require $\boldsymbol{V}(\mu)$ to live in the subspace generated by the left singular vectors:

$$\forall \mu \in I_\mu : \quad \boldsymbol{V}(\mu) \in \boldsymbol{\mathcal{S}}_\ell := \text{span}\{\boldsymbol{x}^{1 \leq i \leq \ell}\} \quad \Leftrightarrow \quad \boldsymbol{V}(\mu) = \boldsymbol{X}\boldsymbol{\Psi}(\mu). \tag{2.16}$$

It is worth to remark that $\boldsymbol{\Psi}(\mu)$ represents some linear combination coefficients that depend on the parameter; they are not the right singular vectors unless $\mu \in \mathcal{P}$. Inserting Eq. (2.16)

into (2.15), we see that the solution can be written as a linear combination of the left singular vectors through some coefficients depending on the parameter and some others depending on time:

$$\boldsymbol{v}(t,\mu) = \boldsymbol{X}\boldsymbol{\Psi}(\mu)\boldsymbol{\alpha}(t), \tag{2.17}$$

where $\boldsymbol{\Psi} \in \mathbb{C}^{\ell \times m}$ and $\boldsymbol{\alpha} \in \mathbb{R}^{m \times 1}$. Displacements can be recovered as shown in Eq. (2.13). Observe that normally $\ell > m$ and therefore $\boldsymbol{\Psi}$ can also be seen as a change to a smaller basis. The usual practice consists in omitting such change of basis; therefore the solution is expressed as $\boldsymbol{v}(t,\mu) = \boldsymbol{X}\boldsymbol{\zeta}(t,\mu)$. On projection,

$$\boldsymbol{X}^T\boldsymbol{P}\boldsymbol{X}\dot{\boldsymbol{\zeta}}(t,\mu) + \boldsymbol{X}^T\boldsymbol{Q}(\mu)\boldsymbol{X}\boldsymbol{\zeta}(t,\mu) = \boldsymbol{X}^T\boldsymbol{g}(t), \tag{2.18}$$

where $\boldsymbol{g}(t) = [\boldsymbol{0} \quad \boldsymbol{f}(t)]^T$. Performing the usual time integration, coefficients $\boldsymbol{\zeta}(t,\mu)$ can be computed.

However, the Eq. (2.18) is not, in general, diagonal and therefore the great advantage of Modal Analysis is lost. Still if $\ell \ll 2n$, the computational complexity is reduced with respect to integrating directly the original system of equations. In exchange, the price to be paid is:

1. Perform a sampling of the parametric space, i.e. solve a QEP for each sampling value of the parameter.

2. Compute the SVD of a full, potentially large matrix.

The latter operations are neither computationally nor conceptually simple. Besides, one might wonder how the procedure could be extended when not one but several parameters are of interest.

In Section 2.4, we present a simpler method that aims at solving parametric problems using a different MOR technique, the PGD. In addition, it allows solving multi-parametric problems naturally, as PGD was conceived to this purpose.

2.3 Approaches based on Harmonic Analysis

Fourier's theory has been largely studied and successfully applied to a wide number of applications in physics and engineering. Among them, Harmonic Analysis is the technique resulting of applying Fourier's theory to obtain the forced dynamic response of a structure. Hence, unforced problems are left aside in this section. Furthermore, we shall consider a general damping that may be proportional or not.

2.3.1 Forced response of generally damped problems

Consider that our dynamic system is submitted to some dynamic action represented by the generalized force vector, $\boldsymbol{f}(t)$. Under rather weak conditions, it is possible to obtain a

frequency representation, $\hat{\boldsymbol{f}}(\omega)$, of the generalized force vector via the direct Fourier Transform. And *vice versa*, from the frequency representation it is possible to recover the *time representation* of the generalized force vector via the inverse Fourier Transform. The pair of direct/inverse transforms is defined as follows:

$$\hat{\boldsymbol{f}}(\omega) = \int_{-\infty}^{+\infty} \boldsymbol{f}(t) \exp\left(-\mathrm{i}\omega t\right) dt \quad \text{and} \quad \boldsymbol{f}(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \hat{\boldsymbol{f}}(\omega) \exp\left(\mathrm{i}\omega t\right) d\omega.$$

Quantities defined in the frequency domain are denoted by a hat, "$\hat{\cdot}$". Analogously, we may also consider the frequency representation of the generalized displacement vector, $\hat{\boldsymbol{u}}(\omega)$, that may be computed from

$$\boldsymbol{A}(\omega)\hat{\boldsymbol{u}}(\omega) = \hat{\boldsymbol{f}}(\omega), \quad \text{with} \quad \boldsymbol{A}(\omega) = -\omega^2\boldsymbol{M} + \mathrm{i}\omega\boldsymbol{C} + \boldsymbol{K}. \tag{2.19}$$

Notice that Eq. (2.19) is obtained by taking the Fourier transform over Eq. (2.1). Matrix $\boldsymbol{A}(\omega)$ has already been defined in Eq. (2.7); here we simply make $\lambda = \mathrm{i}\omega$. It can be seen that the dynamic response depends parametrically on ω. This parametric dependence is intrinsic to Harmonic Analysis and renders it very unattractive when compared to Modal Analysis or even Direct Integration. In practice, Harmonic Analysis is only used to compute the response against rather simple —with few frequencies— periodic forcing terms. If arbitrary (non-periodic) forcing terms are interest, a continuous frequency spectrum should be considered. Although conceptually possible, from a computational point of view this implies solving Eq. (2.19) for each frequency.

Remark 2.2 (*Frequency-dependent damping*). In some fields of engineering, such as in soil mechanics or rheology of suspensions, equations are transformed into the frequency domain by first assuming a linear damping (or viscous) coefficient. Then, when it is observed that the damping behaviour cannot be fitted using that simple model, frequency-dependent damping coefficients are considered. As discussed in [45], this is of course wrong because the time domain model, obtained by performing the inverse Fourier transform of the modified equations, does not respect causality. In spite of that, a nonlinear dependence on frequency of the damping coefficient results in a model that remain linear in the frequency domain, where frequency is only a parameter, or as it will be explained later, a coordinate in the PGD framework. Fractional calculus has been suggested to avoid this incoherent, frequency-dependent modeling of the viscous behaviour [2, 74, 78].

Let us examine in detail the number of solves solves of Eq. (2.19) to be done in practice. Suppose that $\boldsymbol{f}(t)$ is sampled at a sampling frequency $f_t = 1/\Delta t$ Hz in $I_t = [0, T]$. One may be interested in keeping the same time resolution provided by a certain integration scheme[1]. In that case, the sampling frequency may be that corresponding to the time step of the integration scheme. The Fourier transform of the sampled forcing term provides, at most,

[1] Recall that time increment is, in general, subjected to stability/accuracy restrictions so that there is a critical time step, Δt_{crit}, that should not be exceeded.

a maximum frequency which is, by the Nyquist-Shannon theorem [122], half the sampling frequency: $f_{max} = 1/2f_t$. Therefore, if the forcing term is sampled at a frequency $f_t = (2N_\omega - 1)/T$, the forcing term may be written as the linear combination of N_ω harmonics,

$$\boldsymbol{f}(t) = \hat{\boldsymbol{F}} \exp(\mathrm{i}\boldsymbol{\Omega} t), \tag{2.20}$$

$$\text{with} \quad \hat{\boldsymbol{F}} \in \mathbb{C}^{N \times N_\omega} \quad \text{and} \quad \boldsymbol{\Omega} = \operatorname{diag}(\omega_1, \dots, \omega_{N_\omega}), \tag{2.21}$$

which implies by linearity that Eq. (2.19) must be solved N_ω times. Displacements can be written as a linear combination of N_ω harmonics:

$$\boldsymbol{u}(t) = \hat{\boldsymbol{U}} \exp(\mathrm{i}\boldsymbol{\Omega} t) \quad \text{with} \quad \hat{\boldsymbol{U}} = [\hat{\boldsymbol{u}}^1 \cdots \hat{\boldsymbol{u}}^{N_\omega}] \in \mathbb{C}^{N \times N_\omega}, \tag{2.22}$$

where the displacement amplitude vectors $\hat{\boldsymbol{u}}^{1 \le i \le N_\omega}$ are computed from Eq. (2.19) for $\omega = \omega_{1 \le i \le N_\omega}$ and $\hat{\boldsymbol{f}} = \hat{\boldsymbol{f}}^{1 \le i \le N_\omega}$, the columns of $\hat{\boldsymbol{F}}$. Equivalently, Eq. (2.22) may be substituted in Eq. (2.1), and recalling the orthonormality of the Fourier basis, it is easy to see that the problems associated to each frequency can be solved independently. At this point, it can be clearly seen that unless the forcing term is formed of very few frequencies, Eq. (2.19) has to be solved many times.

Remark 2.3 (*Scalar modulation of the generalized force vector*). Sometimes there exist a scalar function $g(t)$ that for every $t \in I_t$ scales the loads applied on the structure by the same factor. Hence, the generalized force vector can be written in this particular case as $\boldsymbol{f}(t) = \boldsymbol{f}_s\, g(t)$, being $\boldsymbol{f}_s$ the force amplitude distribution on the structure. Eq. (2.20) can be rewritten as follows:

$$\boldsymbol{f}(t) = \boldsymbol{f}_s \left(\exp(\mathrm{i}\boldsymbol{\omega} t)\, \hat{\boldsymbol{\alpha}}\right), \tag{2.23}$$

$$\text{with} \quad \boldsymbol{f}_s \in \mathbb{R}^N, \quad \boldsymbol{\omega} = (\omega_1, \dots, \omega_{N_\omega})^T \quad \text{and} \quad \hat{\boldsymbol{\alpha}} \in \mathbb{C}^{N_\omega},$$

where $\boldsymbol{\alpha}$ are the coefficients of the discrete Fourier transform of $g(t)$. Observe that displacements can be recovered as explained in Eq. (2.22) provided that each harmonic is affected by its own weight:

$$\boldsymbol{A}(\omega_i)\hat{\boldsymbol{u}}^i = \boldsymbol{f}_s \quad 1 \le i \le N_\omega, \quad \text{then} \quad \boldsymbol{u}(t) = \hat{\boldsymbol{U}} \exp(\mathrm{i}\boldsymbol{\Omega} t)\, \hat{\boldsymbol{\alpha}}.$$

In consequence, there is no need of recomputing $\hat{\boldsymbol{u}}^i$ when the scaling function $g(t)$ changes; the only operation to be performed is to recombine the columns of $\hat{\boldsymbol{U}}$ with the new Fourier coefficients. This provides a great numerical advantage which will be exploited in the numerical examples.

2.3.2 Forced response of damped parametric problems

It has been shown in Section 2.3.1 that Harmonic Analysis is intrinsically parametric on the frequency. If an additional parametric dependence wants to be considered as we did in Section 2.2.3, Eq. (2.19) turns into

$$\boldsymbol{A}(\omega, \mu)\hat{\boldsymbol{u}}(\omega, \mu) = \hat{\boldsymbol{f}}(\omega), \quad \text{with} \quad \boldsymbol{A}(\omega, \mu) = -\omega^2 \boldsymbol{M} + \mathrm{i}\omega \boldsymbol{C}(\mu) + \boldsymbol{K}, \tag{2.24}$$

which is doubly parametrized. Let us assume that, as before, the solution is written as a linear combination of N_ω harmonics. However, now there is a parameter dependence to be taken into account:

$$\boldsymbol{u}(t) = \hat{\boldsymbol{U}}(\mu) \exp(\mathrm{i}\boldsymbol{\Omega} t). \tag{2.25}$$

Hence, even in the case in which instead of a continuous spectrum only N_ω frequencies are considered, each time the parameter changes, a total of N_ω, N-sized, complex-valued problems must be solved. Recall that a Fourier inverse transform must be performed as well afterwards to recover the time response. POD approaches analogous to the one briefly introduced in Section 2.2.3 are, of course, possible. However, the same disadvantages apply.

2.4 Space-frequency separated representations

This section is concerned with the solution of parametrized structural dynamics models applying the PGD method. The technique here presented is closely related to Harmonic Analysis. It has been shown in §2.3 that Harmonic Analysis introduces a intrinsic parametric dependence on the frequency. However, with the PGD it is possible to obtain quite efficiently the generalized displacements —in the frequency space, thus harmonic amplitudes— for any forcing frequency, and eventually, any parameter value inside predefined ranges. This is achieved by computing space-frequency separated representations. Since the technique here presented is frequency-based, it is only concerned with the long-term forced response.

In exchange, we shall show that the following advantages apply:

1. There is no need to solve the unforced (eigenvalue) parametrized problem.
2. The space-time solution can be recovered with a simple Fourier inverse transform. There is no need of performing a time integration.
3. The PGD solution is valid not only for one particular forcing term but for any forcing term that can be written as a combination of the considered frequencies (linearity).
4. The solution is available for any value of the parameter.

As it has been shown in Chapter 1, by separated representation we understand that the solution can be written as sum of few terms —low-rank— of separated functions, that is, each one of a different coordinate. This assumption is, in fact, quite natural in structural dynamics. Consider for instance Eq. (2.15) found in the context of Modal Analysis. It is clearly a time-parameter separated representation. Or Eq. (2.17) that results from combining Modal Analysis and POD, which separates explicitly space, time and parameter. In Harmonic Analysis, space-frequency or space-frequency-parameter also appear quite naturally in Eq. (2.22) and Eq. (2.25), respectively. The aforementioned equations share the

same difficulty, which is the lack of an efficient treatment of their parametric nature. In Section 2.4.1, we introduce an appropriate framework to treat parametric problems.

2.4.1 A framework for parametric problems: PGD tensor approximations

The technique proposed in this work is built upon Harmonic Analysis and allows computing the complex displacement amplitude vector in a frequency band for any value of the parameter in a certain range. Frequency and parameter are also seen as coordinates of the problem, and thus the dimensionality is increased. The computational domain includes not only the physical coordinates, i.e. the structure, but also the frequency I_ω and parameter ranges I_μ. Suppose that they are discretized using N_ω and N_μ nodes, respectively. Rather than using a brute force approach, that would imply the resolution of $N_\omega \times N_\mu$ direct problems, it is preferable to formulate the problem in tensor spaces because:

1. A brute-force approach lacks of the concept of approximation, as it provides the solution at some points but there is no information about what happens in between. Methods formulated in tensor spaces, like PGD, provide an approximation of the solution on the parametric space.

2. Tensor methods are in general designed to solve high dimensional problems. They become computationally cheaper that brute-force approaches as the dimension of the parametric space increases.

Complex displacements belong to a tensor space $V = V_s \otimes V_\omega \otimes V_\mu$, where in fact $V_s \equiv \mathbb{C}^N$, the Euclidean space of dimension N, since equations of motion have been considered in a discrete matrix form throughout all this work. Recall that N is the number of degrees of freedom of the structure. Equivalently, we are stating that each component of the complex displacement vector belongs to $V_\omega \otimes V_\mu$. To keep coherence with the discrete formulation, let us consider that $V_\omega := \text{span}\{v_\omega^{1 \leq i_\omega \leq N_\omega}\}$ and $V_\mu := \text{span}\{v_\mu^{1 \leq i_\mu \leq N_\mu}\}$ are approximation spaces of finite dimension, such as finite element spaces for instance. Therefore,

$$V_\omega \otimes V_\mu = \text{span}\{v_\omega^{i_\omega} \otimes v_\mu^{i_\mu} \,;\, 1 \leq i_\omega \leq N_\omega,\, 1 \leq i_\mu \leq N_\mu\}.$$

Each component of the displacement vector can be written in the following form:

$$\hat{u}_{i_s}(\omega,\mu) \in V_\omega \otimes V_\mu \quad \Leftrightarrow \quad \hat{u}_{i_s}(\omega,\mu) = \sum_{i_\omega=1}^{N_\omega} \sum_{i_\mu=1}^{N_\mu} u_{i_\omega,i_\mu}\, v_\omega^{i_\omega} \otimes v_\mu^{i_\mu}, \quad 1 \leq i_s \leq N,$$

where $\hat{u}_{i_s}$ is the i_s-th component of the generalized displacement vector. In order to ease the exposition, let us denote $V_s := \text{span}\{\boldsymbol{v}_s^{1 \leq i_s \leq N}\}$ the canonical basis in $\mathbb{C}^N$. Then, abusing of

notation:

$$\hat{\boldsymbol{u}}(\omega,\mu)\in V \quad \Leftrightarrow \quad \hat{\boldsymbol{u}}(\omega,\mu)=\sum_{i_s}^{N}\sum_{i_\omega}^{N_\omega}\sum_{i_\mu}^{N_\mu} u_{i_s,i_\omega,i_\mu}\,\boldsymbol{v}_{\boldsymbol{s}}^{i_s}\otimes v_\omega^{i_\omega}\otimes v_\mu^{i_\mu}, \tag{2.26}$$

where the coefficient defines the (i_s, i_ω, i_μ)-th entry of the displacement tensor $\hat{\boldsymbol{u}}\in\mathbb{C}^N\otimes\mathbb{C}^{N_\omega}\otimes\mathbb{C}^{N_\mu}$, which is still denoted in the same manner for the sake of simplicity. Observe that Eq. (2.26) defines an approximation of the displacements in the parametric space, provided that we are able to compute the tensor $\hat{\boldsymbol{u}}$. In Section 2.4.2, we show how PGD can be used to obtain such approximation.

2.4.2 PGD algorithm to solve the parametric problem

Let us consider the weighted residual form of Eq. (2.24), extended to the parametric space. Using a Galerkin approach, the problem is formulated as follows: find $\hat{\boldsymbol{u}}\in V$ such that

$$\int_{I_\omega}\int_{I_\mu} \hat{\boldsymbol{w}}^H\boldsymbol{A}\hat{\boldsymbol{u}}-\hat{\boldsymbol{w}}^H\hat{\boldsymbol{f}}\,d\omega\,d\mu=0,\quad \forall\hat{\boldsymbol{w}}\in V. \tag{2.27}$$

Matrix $\boldsymbol{A}=-\omega^2\boldsymbol{M}+\mathrm{i}\omega\boldsymbol{C}(\mu)+\boldsymbol{K}$ depends parametrically on the pulsation and the parameter. In general, it is not difficult to find an affine decomposition of the damping matrix. Let us assume without loss of generality the simplest affine decomposition, that is $\boldsymbol{C}(\mu)\equiv\mu\boldsymbol{C}$. Since a Galerkin approach has been used, $\hat{\boldsymbol{w}}(\omega,\mu)$ is expressed on the same tensor basis as Eq. (2.26). Substituting both $\hat{\boldsymbol{u}}(\omega,\mu)$ and $\hat{\boldsymbol{w}}(\omega,\mu)$ into Eq. (2.27) and integrating we arrive to the following algebraic tensor system:

$$\boldsymbol{A}\,\hat{\boldsymbol{u}}=\hat{\boldsymbol{f}},\quad \text{with}\quad \hat{\boldsymbol{u}}\in\mathbb{C}^N\otimes\mathbb{C}^{N_\omega}\otimes\mathbb{C}^{N_\mu}, \tag{2.28}$$

where $\boldsymbol{A}$ is a tensor operator with three modes defined as follows:

$$\boldsymbol{A}=-\boldsymbol{M}\otimes\boldsymbol{B}_\omega\otimes\boldsymbol{M}_\mu+\mathrm{i}\boldsymbol{C}\otimes\boldsymbol{D}_\omega\otimes\boldsymbol{D}_\mu+\boldsymbol{K}\otimes\boldsymbol{M}_\omega\otimes\boldsymbol{M}_\mu.$$

See Appendix A for details on the definition of these operators. It is important to observe that $\boldsymbol{A}$ is not self-adjoint and therefore this will influence the choice of the PGD algorithm, among those discussed in §1.4.2 and §1.4.3. We discuss these aspects in together with the numerical examples in §2.5.

The solution of Eq. (2.28) is too hard to use a direct solver and needs to be approximated. The PGD method assumes the displacement tensor can be well approximated in the subset of tensors that can be written as a sum of rank-one tensors:

$$\hat{\boldsymbol{u}}\approx\hat{\boldsymbol{u}}^{(M)}=\sum_{m=1}^{M}\hat{\boldsymbol{u}}^m, \tag{2.29}$$

$$\text{with}\quad \hat{\boldsymbol{u}}^m=\boldsymbol{u}_s^m\otimes\boldsymbol{u}_\omega^m\otimes\boldsymbol{u}_\mu^m\quad\text{and}\quad \boldsymbol{u}_s^m\in\mathbb{C}^N,\ \boldsymbol{u}_\omega^m\in\mathbb{C}^{N_\omega},\ \boldsymbol{u}_\mu^m\in\mathbb{C}^{N_\mu}\quad\text{for}\quad 1\le m\le M.$$

We denote by $\hat{\boldsymbol{u}}^{(M)}$ a rank-M approximation of the solution and by $\hat{\boldsymbol{u}}^m$ the m-th rank-one tensor of the approximation. Vectors $\boldsymbol{u}_s^m$, $\boldsymbol{u}_\omega^m$ and $\boldsymbol{u}_\mu^m$ are also called space, pulsation and parameter *modes*, respectively. A Greedy approach is used to correct the approximation successively by adding a rank-one tensor at a time. After computing M terms, the residual is defined as:

$$\boldsymbol{r}^M = \boldsymbol{f} - \sum_{m=1}^{M} \boldsymbol{A}\,\hat{\boldsymbol{u}}^m,$$

and the new term, still denoted by $\hat{\boldsymbol{u}}$, is computed from

$$\boldsymbol{A}\,\hat{\boldsymbol{u}} = \boldsymbol{r}^M. \tag{2.30}$$

Eq. (2.30) defines a nonlinear problem that is generally solved by implementing an Alternating Directions algorithm. We stop adding terms when the norm of the residual is small enough, or when some error estimator is satisfied, see references [4, 87]. It is worth to recall that at convergence, an approximation of the displacements for every frequency in a frequency band and every parameter value inside a range will be available. This approximation will be referred as the Generalized Transfer Function (GTF).

2.4.3 Using the GTF to compute the dynamic response

In this Section we show how the GTF can be used to compute the forced dynamic response due to some loading applied on the structure. It will be shown that exploring the parametric space only requires an inexpensive post-processing. Let us assume that the GTF has already been computed, and suppose that we are interested in recovering the dynamic response of the degree of freedom $i_s^\star$, for a certain value of the parameter $\mu \in I_\mu$. Assume without loss of generality that the index associated to the parameter value is $i_\mu^\star \in [1, N_\mu]$.

Remark 2.4 (*Intermediate values of the parameter*). The approximation can be evaluated at any intermediate value thanks to the underlaying approximation (finite elements, for instance) defined in Eq. (2.26).

Assume that convergence of the PGD solution was reached in M terms, that is a rank-M separated representation. First step consists in particularizing the GTF at indices $i_s^\star$ and $i_\mu^\star$ (index i_ω is set free):

$$\hat{\boldsymbol{u}}^\star = \hat{\boldsymbol{u}}^{(M)}_{i_s^\star,\cdot,i_\mu^\star} = \sum_{m=1}^{M} \hat{u}^m_{i_s^\star}\hat{u}^m_{i_\mu^\star}\,\hat{\boldsymbol{u}}^m_\omega. \tag{2.31}$$

Vector $\hat{\boldsymbol{u}}^\star \in \mathbb{C}^{N_\omega}$ contains the complex displacement amplitudes associated to each pulsation. Notice that Eq. (2.31) simply expresses the linear combination of the pulsation modes. The dynamic response can be recovered by taking the Fourier inverse transform, or equivalently,

from Eq. (2.22):

$$u^\star(t) = \sum_{i_\omega=1}^{N_\omega} \hat{u}^\star_{i_\omega} \exp\left(\mathrm{i}\omega_{i_\omega} t\right). \tag{2.32}$$

From a computational point of view, the inverse Fast Fourier transform (IFFT) can be used [44]. If the response corresponding to a new parameter value wants to be computed, Eq. (2.31) must be particularized again and Eq. (2.32) must be recomputed. However, it is worth to emphasize that both equations are inexpensive. The present approach is therefore ideally suited for a multi-query context in which the dynamical behaviour has to be optimized with respect to some parameter.

Consider the particular case in which the load is scaled in time, as explained in §2.3.1. Observe that the GTF is even more general: it is valid not only for any frequency and parameter value but also for any scaling function. The dynamic response can be recovered as follows:

$$u^\star(t) = \sum_{i_\omega=1}^{N_\omega} \hat{\alpha}_{i_\omega} \hat{u}^\star_{i_\omega} \exp\left(\mathrm{i}\omega_{i_\omega} t\right), \tag{2.33}$$

where $\hat{\alpha}_{i_\omega}$ is the i_ω-th entry of vector $\hat{\boldsymbol{\alpha}}$, already defined in §2.3.1.

Remark 2.5 (*Continuous frequency approach*). To keep coherence with the rest of the Chapter, this Section has been presented using a discrete approach. This is also coherent with the numerical implementation of the method here presented. The frequency band has been discretized and therefore instead of Fourier direct/inverse transforms, we work with Fourier discrete direct/inverse transforms. However, the continuous formulation reveals some details that might go unnoticed with the discrete formulation. Consider the dynamics of a solid occupying a region of the space $I \subset \mathbb{R}^{d\leq 3}$. The continuous equivalent of the displacement tensor, that is, the multivariate function $\hat{\boldsymbol{u}}(\boldsymbol{x}, \omega, \mu)$. For some position $\boldsymbol{x}^\star \in I$ and parameter $\mu^\star \in I_\mu$ of interest, we can particularize to extract the complex displacement amplitudes: $\hat{\boldsymbol{u}}^\star(\omega) = \hat{\boldsymbol{u}}(\boldsymbol{x}^\star, \omega, \mu^\star)$. The time evolution can be recovered by performing a Fourier inverse transform:

$$\boldsymbol{u}^\star(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \hat{\boldsymbol{u}}^\star(\omega) \exp(\mathrm{i}\omega t)\, d\omega.$$

If the scalar modulation of the force vector applies —see §2.3.1— let $\hat{g}(\omega)$ be the Fourier transform of the scaling function $g(t)$. To recover the dynamic response in the time domain:

$$\hat{\boldsymbol{u}}^\star(t) = \mathcal{F}^{-1}\left[\hat{\boldsymbol{u}}^\star(\omega)\, \hat{g}(\omega)\right] = \int_0^t \boldsymbol{u}^\star(t-\tau)\, g(\tau)\, d\tau, \tag{2.34}$$

which is the convolution of the particularization of the GTF and the scaling function. This result is of great interest for real-time applications because:

1. In such applications, the scaling function is not a priori given; it is sampled online and we need to output the dynamic response up to the current time. Eq. (2.34) perfectly fits with this approach.
2. It reflects causality: the response at time t only depends on what has already happened.
3. From a computational point of view, rather than transforming the scaling function and then recover the response by an inverse transform, it is preferable to inverse once the GTF and perform its convolution with the scaling function in the time domain.
4. It is rather inexpensive to compute, and therefore the dynamic response can be outputted at high refresh rates, that are hardly attainable by other methods when non-proportionally damped or parametric problems are of interest.

2.4.4 Reduced Basis approach to recover the transient regime

The frequency approach presented here is only concerned with the long-term forced response and consequently, it does not depend on the initial conditions of the structure. If one is interested in evaluating the transient regime, it is always possible to derive a Reduced Basis approach from the Generalized Transfer Function computed with the PGD.

A reduced basis can be formed by keeping the only the real part of the GTF space modes:

$$\boldsymbol{V} := \mathrm{real}\{\hat{\boldsymbol{U}}_s\} \quad \text{with} \quad \hat{\boldsymbol{U}}_s := [\boldsymbol{u}_s^1\, \boldsymbol{u}_s^2\, \cdots\, \boldsymbol{u}_s^M].$$

This basis is well-suited for any parameter value $\mu \in I_\mu$ simply by the definition of the GTF. The transient solution is sought by requiring the solution to live in the linear space generated by these reduced basis at any time and and for any parameter value:

$$\forall t \in I_t \quad \text{and} \quad \forall \mu \in I_\mu : \quad \boldsymbol{u}(t,\mu) \in \boldsymbol{\mathcal{V}}_M := \mathrm{span}\{\boldsymbol{v}^{1\leq m\leq M}\} \quad \Leftrightarrow \quad \boldsymbol{u}(t,\mu) = \boldsymbol{V}\boldsymbol{\alpha}(t,\mu).$$

Coefficients $\boldsymbol{\alpha}$ are computed by Galerkin projection of the full system onto the subspace $\boldsymbol{\mathcal{V}}_M$. This yields a reduced system which must be solved online, as it is standard in a Reduced Basis approach. Of course, to be efficient, the Galerkin projection must be computed only once which in practice requires that we are able to find an affine decomposition of the damping matrix. This requirement is completely standard in the Reduced Basis community, and it is also shared by the PGD method, as expressed in Hypothesis 1.1.

2.5 Numerical examples

To illustrate the technique proposed in this Chapter, we consider different examples. We restrict ourselves to solid dynamics and therefore the mass, damping and stiffness matrices come from the FE discretization of a solid $I \subset \mathbb{R}^{d\leq 3}$. In consequence, the equation of equilibrium of momentum is solved

$$\rho \ddot{\boldsymbol{u}}(t) - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}(t) = \boldsymbol{0},$$

with a Kelvin-Voigt model, defined by:

$$\boldsymbol{\sigma} = \boldsymbol{D} : (\boldsymbol{\varepsilon} + \mu \dot{\boldsymbol{\varepsilon}}),$$

where $\boldsymbol{D}$ is the linear elasticity strain tensor. In consequence, the structural damping comes from the material viscous behaviour. To quantify the amount of damping present in the system, we introduce the damping factor ξ, that for a Kelvin-Voigt model is expressed as follows [129]:

$$\xi = \frac{1}{2}\mu\omega_0, \quad \text{with} \quad \omega_0 = \sqrt{\lambda_0}, \tag{2.35}$$

being λ_0 is the smallest eigenvalue that results from solving the Generalized Eigenvalue Problem defined in Section 2.2.1. For $\xi = 0\%$ there is no damping, i.e. a pure elastodynamics behaviour is considered, whereas $\xi = 100\%$ means that all vibration modes are damped out.

Reference solutions are in all cases computed using a classic Newmark method [101] using parameters $\beta = 1$ and $\gamma = 0.5$. Note that other methods such as those introduced in [70, 72] might be also used.

Three numerical examples are designed to illustrate the performance of the technique:

1. A simple one-dimensional example by means of which the validity of a space-frequency GTF for different loadings is demonstrated. We also show the importance of the amount of damping present in the structure.

2. A two-dimensional example using a space-frequency GTF.

3. A two-dimensional example with non-proportional damping using a space-frequency-parameter GTF.

2.5.1 Dynamic response of a rod

Let us start with a very simple example. Consider a rod of unitary length and linear elastic behaviour with mass density $\rho = 1$ kg/m^3 and Young modulus 10^4 Pa. The left end of the rod is clamped and a dynamic load is applied on its right end. It can be easily computed (even analytically) that the first natural frequency is located at 25 Hz. A damping factor of 10% is chosen in this example.

A uniform spatial discretization is used with 25 linear two-node elements of size 0.04m. The time step used for the reference solution is $\Delta t = 10^{-4}$s. To recover the same time resolution, a maximum frequency of $f_{max} = 5 \cdot 10^3$Hz needs to be considered. Therefore, the space-frequency separated representation is computed in $I_\omega = 2\pi\,[0, 5 \cdot 10^3]\text{s}^{-1}$. The frequency domain is discretized using a mesh which consists of 2500 $\mathcal{C}^0$-continuous linear elements.

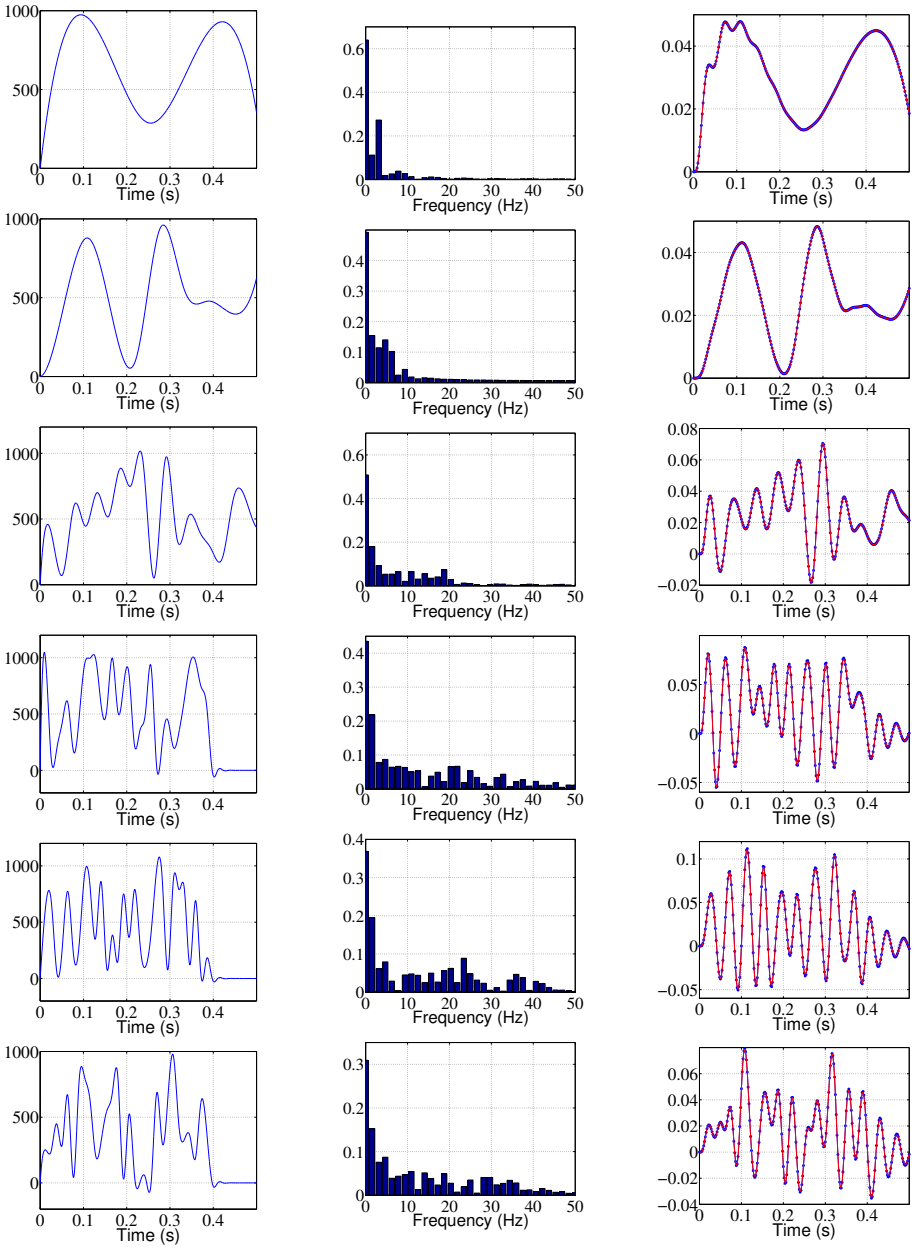
Once the transfer function is available, we compute the response for different randomly generated forces. These forces are generated by setting some random points in the time domain and then fitting a cubic spline. Fig. 2.1a shows six different excitations and Fig. 2.1b their corresponding frequency spectrum up to 50Hz. Despite the excitations look very different, they are all described with low-frequency harmonics which means that, in fact, a narrower frequency band could be considered to compute the frequency function.

Fig. 2.1c shows a very good matching between the reference solution (red solid line) and the solution recovered after the online step (blue markers).

The damping ratio has a key influence on the accuracy of the technique. As explained before, the rod has a natural frequency at 25Hz, which is inside the frequency band for which the transfer function is computed. It is worth to point out that the computation of the transfer function is only possible if some amount of damping is present in the system. Considering no damping at all, i.e. $\xi = 0$, is equivalent to try to solve a singular equation, which is of course not possible using PGD nor any other resolution technique. The influence of the damping ratio is studied in Fig. 2.3. To this end, we apply a triangular load during 0.1s and we observe the response of the system until 0.5s; see Fig. 2.2. We see that the solution deteriorates as the damping ratio tends to zero, see Fig. 2.3a. Apart from this fact, the expected behaviour is observed. For small amounts of damping the rod keeps oscillating after the excitation has vanished (underdamped behaviour). When the damping is increased, the oscillations decay more rapidly and, finally, for $\xi = 100\%$ we observe a critically damped behaviour, that is, the smallest amount of damping for which system does not oscillate.

2.5.2 Dynamic response of a two-dimensional plate

Consider a two-dimensional $I = [-0.5, 0] \times [0.5, 0.5]\text{m}^2$ plate whose bottom side is clamped and submitted to a point-load applied on the middle point of the top side. In consequence, the point where the load applies is $\boldsymbol{x}_F = (0, 0.5)$m. This arrangement is shown in Fig. Fig.



(a) Force amplitudes (b) Frequency spectrum (c) Midway node displ.

Figure 2.1: Dynamic response at the mid-side node for several randomly generated excitations. Comparison between the reference solution (red solid line) and the PGD solution (blue markers).

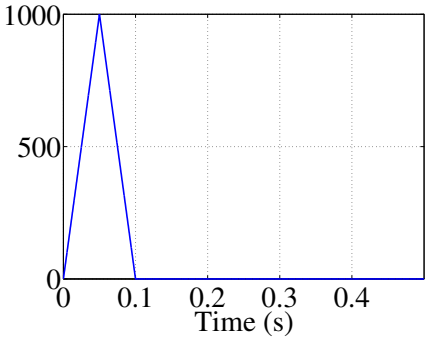
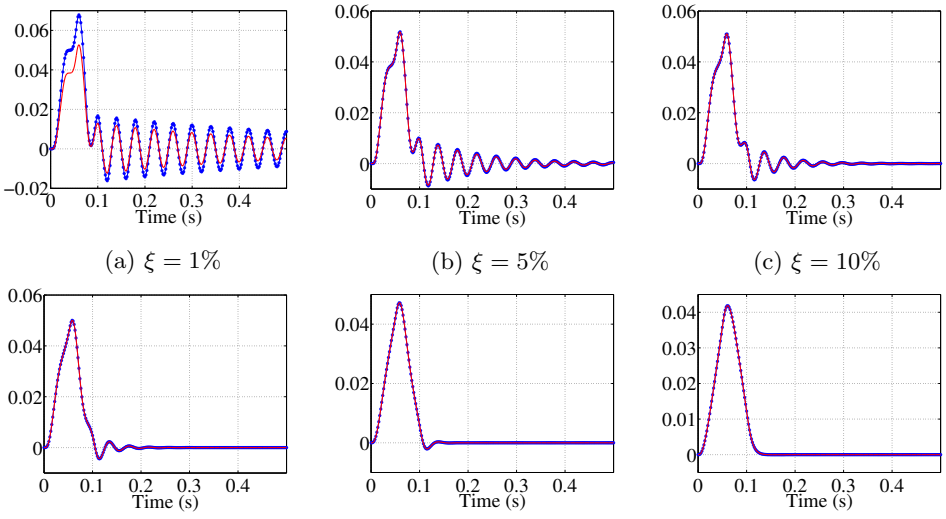


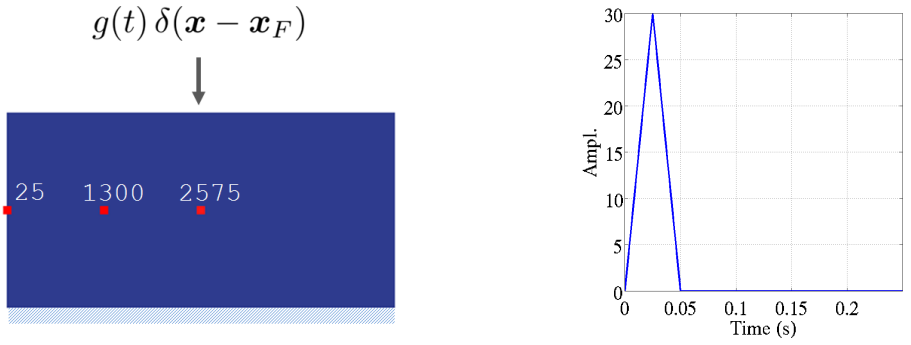
Figure 2.2: Triangular excitation used to study the influence of the damping ratio.



(a) $\xi = 1\%$ (b) $\xi = 5\%$ (c) $\xi = 10\%$

(d) $\xi = 20\%$ (e) $\xi = 50\%$ (f) $\xi = 100\%$

Figure 2.3: Mid-side response for different damping ratios. Comparison between the reference solution (red solid line) and the PGD solution (blue markers).



(a) Geometry

(b) Load amplitude, $g(t)$

Figure 2.4: Dynamics of a 2D plate: geometry, loading amplitude and comparison nodes

2.4a. The nodes where the solution will be compared to the reference are highlighted in red. Table 2.1 shows their numbering and location. The amplitude of the point load is

Node Numbering	**Point** (x,y)
25	$(-0.50, 0.25)$
1300	$(-0.25, 0.25)$
2575	$(0.00, 0.25)$

Table 2.1: Location of the comparison nodes

modulated by a triangular function whose evolution is shown in Fig. 2.4b and its peak value is 30N.

A structured mesh made of four-node quadrilaterals is used for the plate, containing 5000 elements and 10100 degrees of freedom. The characteristic mesh size is $H = 0.014$m. A unit mass density is considered, $\rho = 1\text{kg/m}^3$, whereas the Young modulus is $E = 10^2$Pa and the Poisson's coefficient is $\nu = 1/3$. The speed of sound in a plate made of such material is:

$$c = \sqrt{\frac{E}{\rho}} = 10\text{m/s}.$$

We select a time step $\Delta t = 10^{-3}\text{s} = 0.71 \times \text{H/c}$. In consequence, the elastic waves travel each time step a distance which is less (about 70%) than the characteristic element size. The damping coefficient is set to $\mu = 10^{-2}$s, which using Eq. (2.35), yields a damping ratio of $\xi = 8.1\%$.

To recover the same time resolution, a maximum frequency of $f_{max} = 5 \cdot 10^2$Hz needs to be considered. Therefore, the space-frequency separated representation is computed in

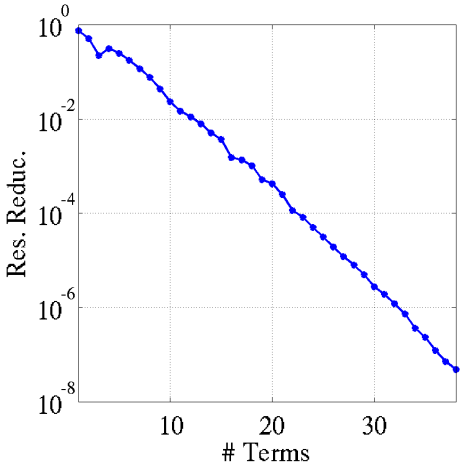
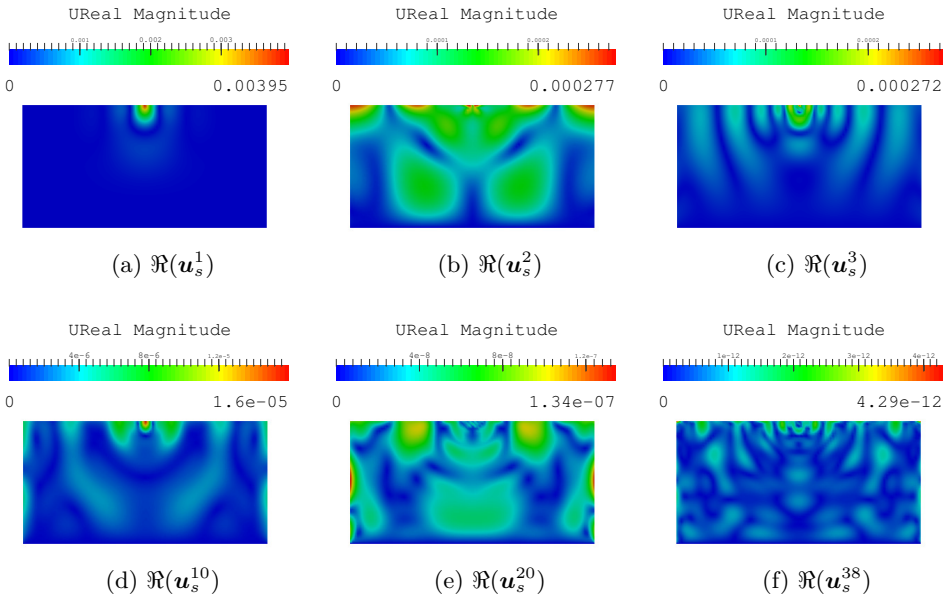


Figure 2.5: GTF convergence: residual reduction convergence of the GTF computation.

$I_\omega = 2\pi\,[0, 5 \cdot 10^2]\text{s}^{-1}$. The frequency domain is discretized using the same number of two-node elements than in §2.5.1, that is 2500 $\mathcal{C}^0$-continuous linear elements.

Once the problem parametrization has been established and detailed, we shall first analyze the results referred to the space-frequency GTF computation, and then we analyze the post-processing part in which the time response is recovered. In spite that the structural dynamics operator in the frequency domain is not self-adjoint, a standard PGD algorithm succeeded to converge to a predefined tolerance of $5 \cdot 10^{-8}$. Recall that the convergence is measured as the residual reduction in Euclidean norm. Fig. 2.5 shows the convergence as a function of the rank of the separated representation. An exponential but non-monotonic convergence can be observed. In particular, it can be observed that the fourth mode increases the residual instead of reducing it. The PGD computation converged with $M = 38$ terms, which is the rank of the final separated representation. The total complexity of the space-frequency solution is roughly $N_s \times N_\omega \approx 25 \cdot 10^6$ complex unknowns. The PGD representation of the solution takes $M \times (N_s + N_\omega) \approx 5 \cdot 10^5$, which is more than 50 times more efficient than the explicit representation. Fig. 2.6 and Fig. 2.7 show the real and imaginary part of the space modes, respectively. In particular, the 1st, 2nd, 3rd, 10th, 20th and 38th (the last mode) are depicted. In general it can be seen that the modes capture progressively the main features of the solution. Last modes tend to capture small features to correct locally the solution where is needed. Space modes are not normalized and therefore it is interesting to look at magnitude, which progressively decreases. This qualitative observation can be better appreciated by looking at Fig. 2.8. It shows the 1st, 2nd, 3rd, 10th, 20th and 38th frequency modes; the real part is depicted in blue while the imaginary part is depicted in red. Frequency modes are shown normalized. The y-axis is set in logarithmic scale so



(a) $\Re(\boldsymbol{u}_s^1)$ (b) $\Re(\boldsymbol{u}_s^2)$ (c) $\Re(\boldsymbol{u}_s^3)$

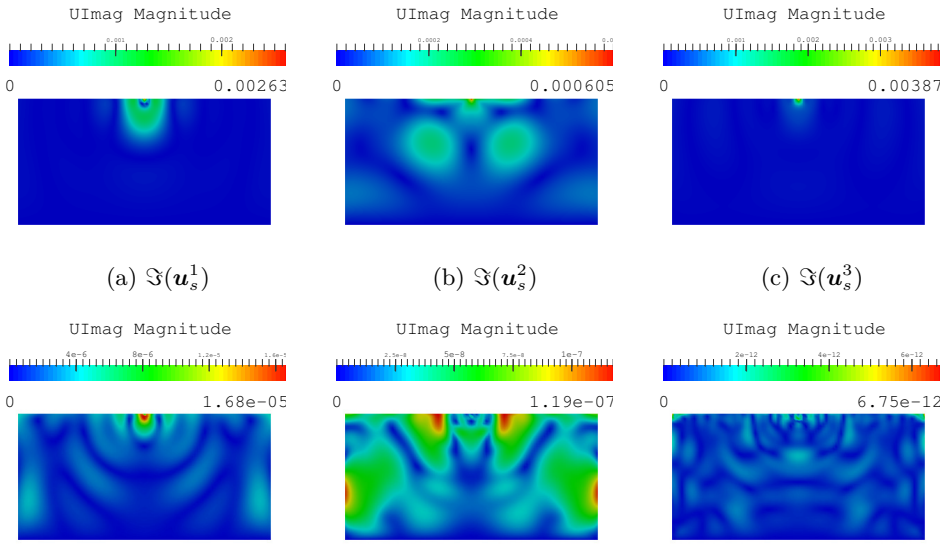
(d) $\Re(\boldsymbol{u}_s^{10})$ (e) $\Re(\boldsymbol{u}_s^{20})$ (f) $\Re(\boldsymbol{u}_s^{38})$

Figure 2.6: GTF calculation: real part of the space modes.

as to better appreciate their shape. As it has been pointed out before, the more modes are added to the solution, the sharper they become.

Once the results of the GTF have been shown and analyzed, we shall study the results obtained after post-processing it to recover the time response. To this end, we perform the convolution of the GTF with the loading amplitude function $g(t)$, as explained in Remark 2.5. The displacement magnitude obtained in such manner is compared to the reference solution in Fig. 2.9a at the comparison nodes, defined in Fig. 2.4a. Observe that the degrees of freedom associated to these particular nodes can be isolated from the GTF and convolved with the loading amplitude; that is, we do not need to operate with the whole system of equations to get the time response at some region, as we do in other methods. Of course, the velocities and the accelerations may be recovered from the displacement field. Although any differentiation technique could be used, here we apply the Newmark scheme, the same used to compute the reference solution. The Newmark method requires operating with the all degrees of freedom, which in practice may not be the best option. Still we proceed in such manner make a fair comparison of the results.

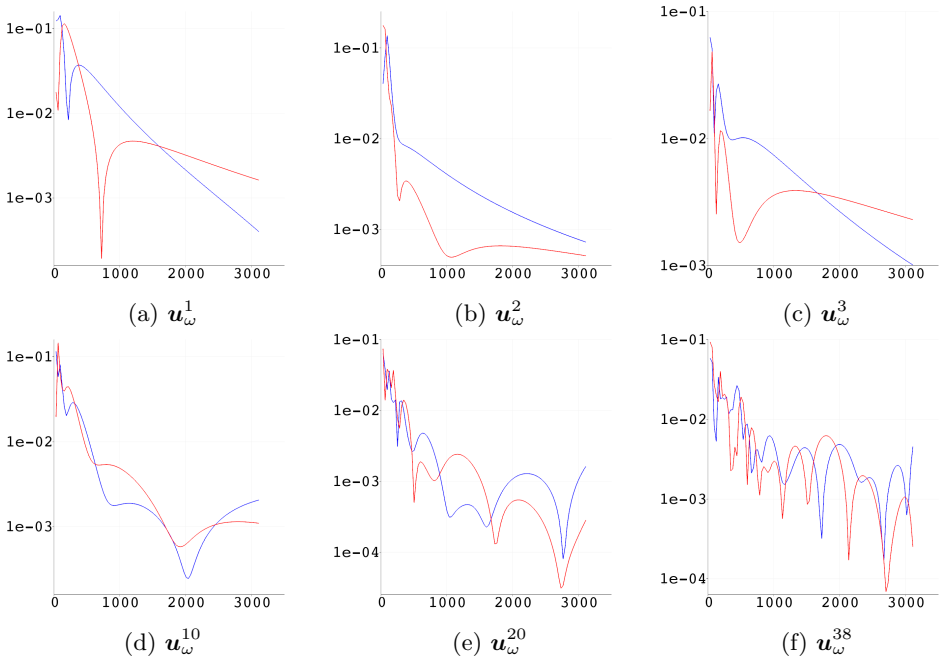
It has to be highlighted that a very good agreement between the reference and PGD solutions is obtained, even for the recovered velocities and accelerations.



(a) $\Im(\boldsymbol{u}_s^1)$ (b) $\Im(\boldsymbol{u}_s^2)$ (c) $\Im(\boldsymbol{u}_s^3)$

(d) $\Im(\boldsymbol{u}_s^{10})$ (e) $\Im(\boldsymbol{u}_s^{20})$ (f) $\Im(\boldsymbol{u}_s^{38})$

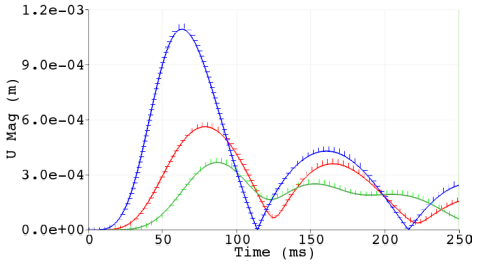
Figure 2.7: GTF calculation: imaginary part of the space modes.



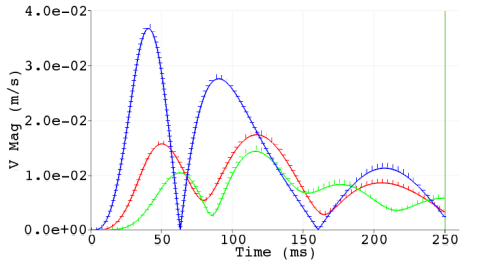
(a) $\boldsymbol{u}_\omega^1$ (b) $\boldsymbol{u}_\omega^2$ (c) $\boldsymbol{u}_\omega^3$

(d) $\boldsymbol{u}_\omega^{10}$ (e) $\boldsymbol{u}_\omega^{20}$ (f) $\boldsymbol{u}_\omega^{38}$

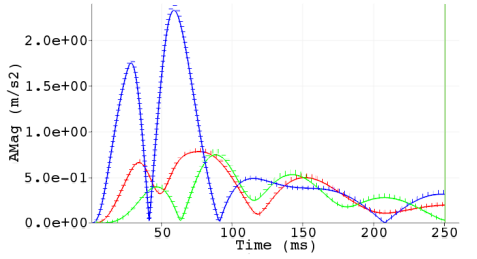
Figure 2.8: GTF calculation: real (blue) and imaginary (red) parts of the frequency modes.



(a) Displacement Magnitude



(b) Velocity Magnitude



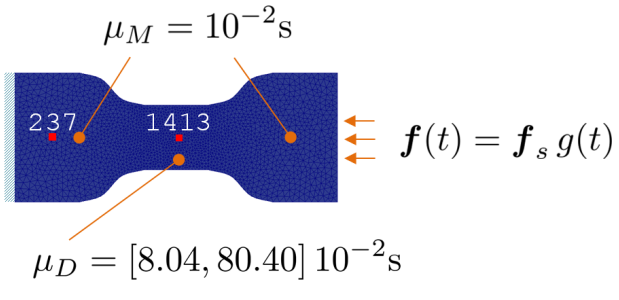
(c) Acceleration Magnitude

Figure 2.9: Comparison of displacements, velocities and accelerations at the comparison nodes, defined in Fig. 2.4a. Green is node 25, red is node 1300 and blue is node 2575; solid line corresponds to the reference solution while dotted line corresponds to the PGD solution.

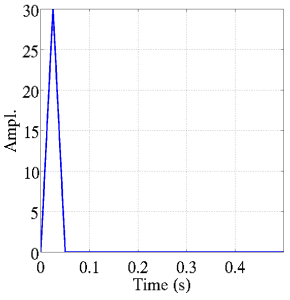
2.5.3 Parametric dynamic response of a damper

In this section, we consider a damper clamped on the left-hand side and submitted to a pressure load of 0.10m length applied on the right-hand side on a segment defined by $\Gamma_F = (1.00, 0.15) \times (1.00, 0.25)$m. The structure is non-proportionally damped. The central part of the damper is made of a material with damping coefficient μ_D whereas in the two extremes damping is $\mu_M = 10^{-2}$s. This arrangement is shown in Fig. Fig. 2.10a. A space-frequency-parameter GTF wants to be computed to analyze the influence of $\mu_D \in I_\mu := [\mu_0, \mu_f] = [8.04, 80.40]\, 10^{-2}$s. If the whole piece was made of a material with damping coefficient equal to μ_0, the damping ratio would be $\xi_0 = 10\%$; with μ_f, the damping ratio would be $\xi_0 = 100\%$. Since the damping zone occupies an area with is less than a third of the total, the effective damping ratio is smaller.

The load is scaled by a triangular modulation function defined in Fig. 2.10b. The time period of interest reaches now 500ms.



(a) Geometry



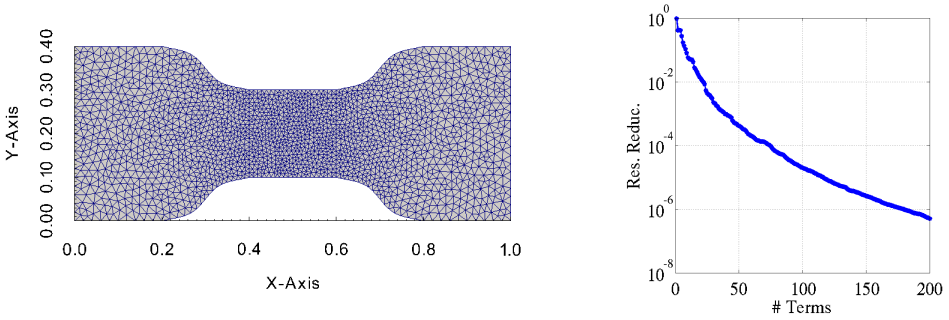
(b) Load amplitude, $g(t)$

Figure 2.10: Dynamics of a damper: geometry, loading amplitude and damping distribution

An unstructured mesh made of three-node triangles is used for the damper. A total of 2258 elements are used which yields 4212 degrees of freedom. The characteristic mesh size is $H = 0.012$m. The same mass density, Young modulus and Poisson's coefficient as in §2.5.2 are considered. In consequence, the speed of sound is 10m/s. We select a time step $\Delta t = 10^{-3}\text{s} = 0.82 \times \text{H/c}$. In consequence, the elastic waves travel each time step a distance which is about the 80% of the characteristic element size. The mesh is shown in Fig. 2.11a.

To recover the same time resolution, a maximum frequency of $f_{max} = 5 \cdot 10^2$Hz needs to be considered. Therefore, the space-frequency separated representation is computed in $I_\omega = 2\pi\,[0, 5 \cdot 10^2]\text{s}^{-1}$. The frequency domain is discretized using the same number of two-node elements than in §2.5.2, that is 2500 $\mathcal{C}^0$-continuous linear elements. The parameter domain, I_μ, is discretized with the same kind of mesh. A total of 100 elements were used.

Once the problem setting has been presented, we shall first analyze the results referred



(a) Computational mesh

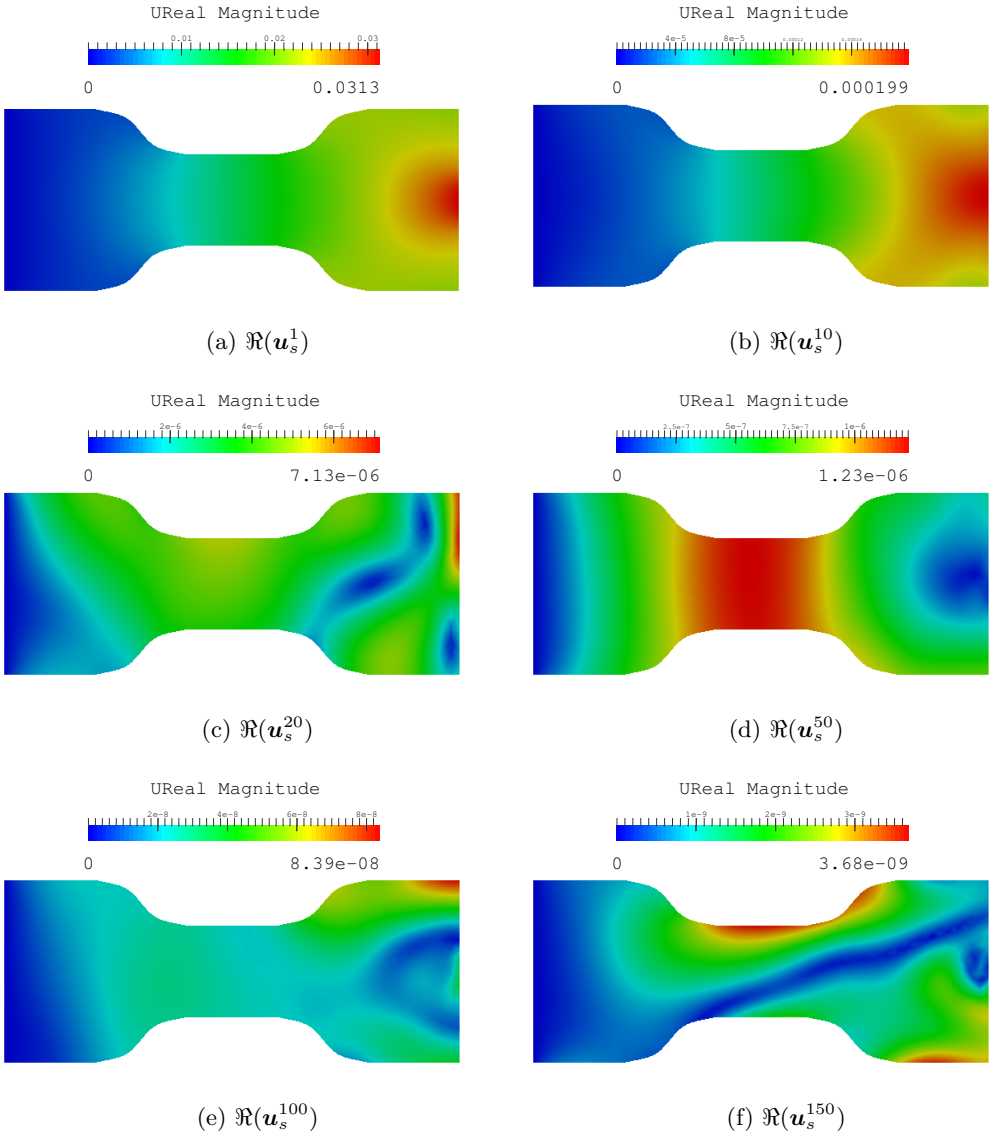
(b) GTF convergence

Figure 2.11: Dynamics of a damper: computational mesh and GTF convergence.

to the computation of the space-frequency-parameter separated representation, and then we analyze the post-processing part in which the time response is recovered. As in §2.5.2, a standard PGD algorithm is applied to compute such separated representation. It succeeded to converge to a predefined tolerance of $5 \cdot 10^{-7}$ —that is an order of magnitude less than in the previous example— and considerable difficulties were encountered. Fig. 2.5 shows the residual reduction as a function of the rank of the separated representation. The first issue to be remarked is that the convergence is not exponential anymore. This is because the problem now is three-dimensional instead of two-dimensional. Convergence is not monotonic, the same as in the previous example. The PGD computation was stopped with $M = 200$ terms, which is relatively high.The total complexity of the space-frequency solution is roughly $N_s \times N_\omega \times N_\mu \approx 1.1 \cdot 10^9$ complex unknowns. The PGD representation of the solution takes $M \times (N_s + N_\omega + N_\mu) \approx 1.4 \cdot 10^6$, which is more than 780 times more efficient than the explicit representation.

Fig. 2.12 and Fig. 2.13 show the real and imaginary part of the space modes, respectively. In particular, the 1st, 10th, 20th, 50th, 100th and 150th are depicted. Fig. 2.14 and Fig. 2.15 show the frequency and parameter modes, respectively. In both cases, the real part is depicted in blue while the imaginary part is depicted in red. The y-axis is set in logarithmic scale so as to better appreciate their shape.

The dynamic time response is recovered from the GTF, particularized for the desired value of μ_D, by performing a simple post-processing. Displacements, velocities and accelerations are compared to the reference solution in Fig. 2.16. Both velocities and accelerations are recovered from displacements using a Newmark scheme, as explained in the previous example. The comparison is made at two nodes: node 237, located at mid height on the left-hand side of the damper, and node 1413, located at the center of the damper. Both nodes are depicted in Fig. 2.10a. In each case, the reference and PGD solutions are depicted in solid and dotted line, respectively. The GTF is particularized for two different values

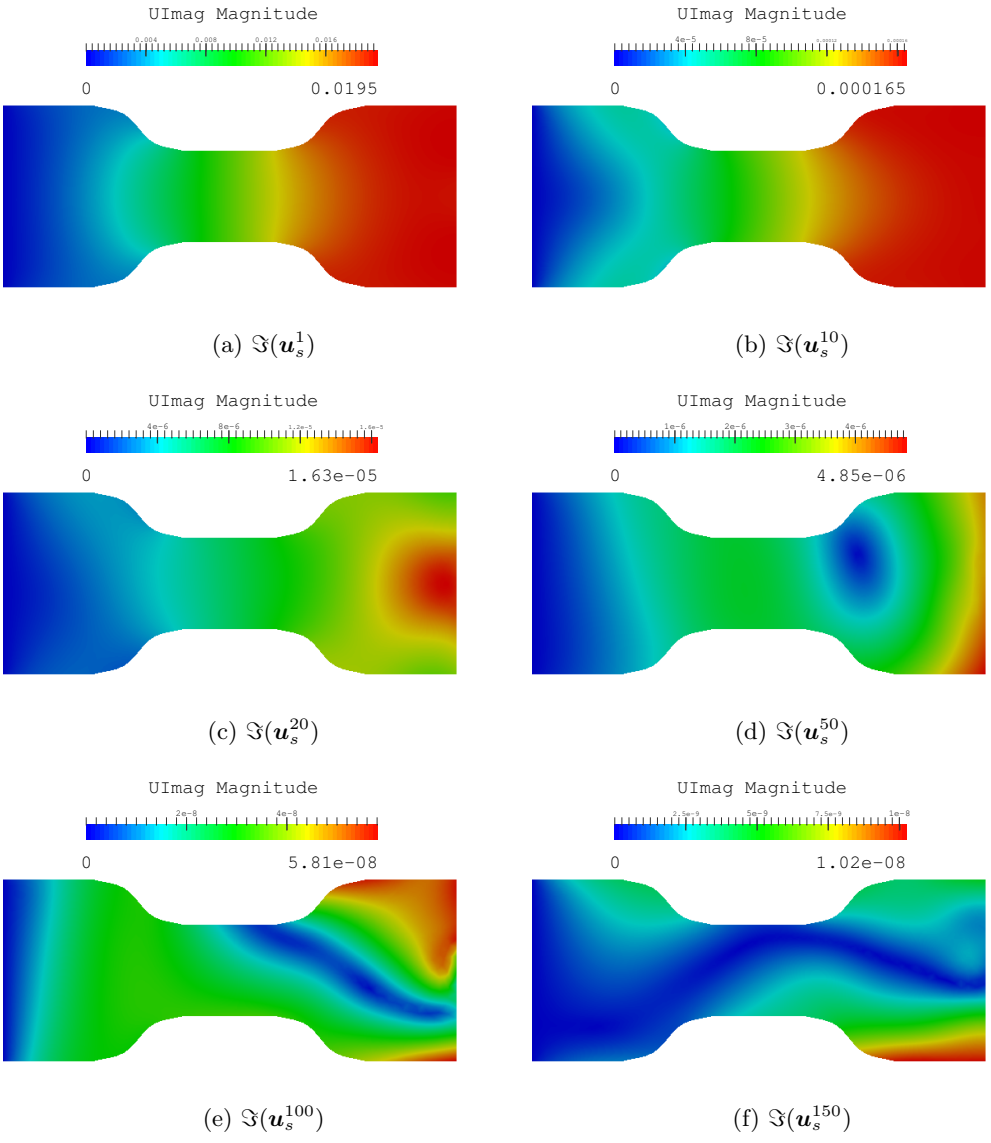


(a) $\Re(\boldsymbol{u}_s^1)$ (b) $\Re(\boldsymbol{u}_s^{10})$

(c) $\Re(\boldsymbol{u}_s^{20})$ (d) $\Re(\boldsymbol{u}_s^{50})$

(e) $\Re(\boldsymbol{u}_s^{100})$ (f) $\Re(\boldsymbol{u}_s^{150})$

Figure 2.12: GTF calculation: real part of the space modes.

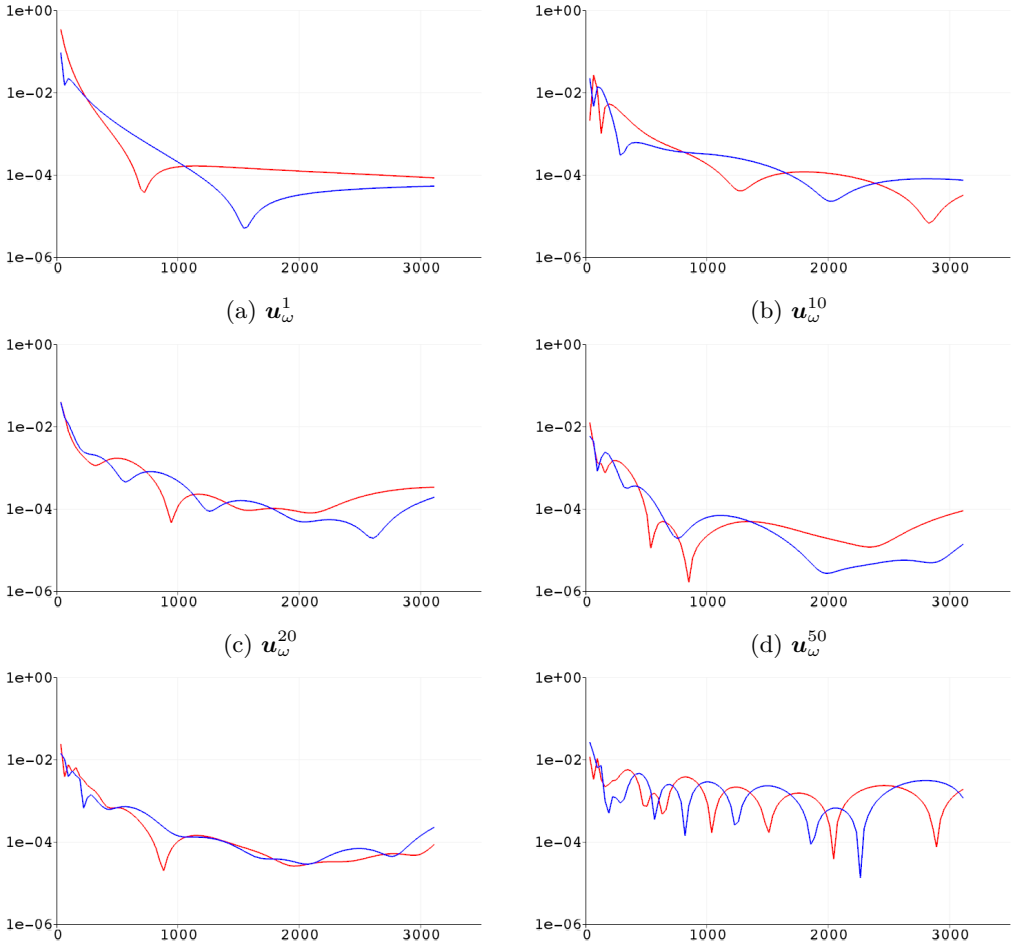


(a) $\Im(\boldsymbol{u}_s^1)$ (b) $\Im(\boldsymbol{u}_s^{10})$

(c) $\Im(\boldsymbol{u}_s^{20})$ (d) $\Im(\boldsymbol{u}_s^{50})$

(e) $\Im(\boldsymbol{u}_s^{100})$ (f) $\Im(\boldsymbol{u}_s^{150})$

Figure 2.13: GTF calculation: real part of the space modes.

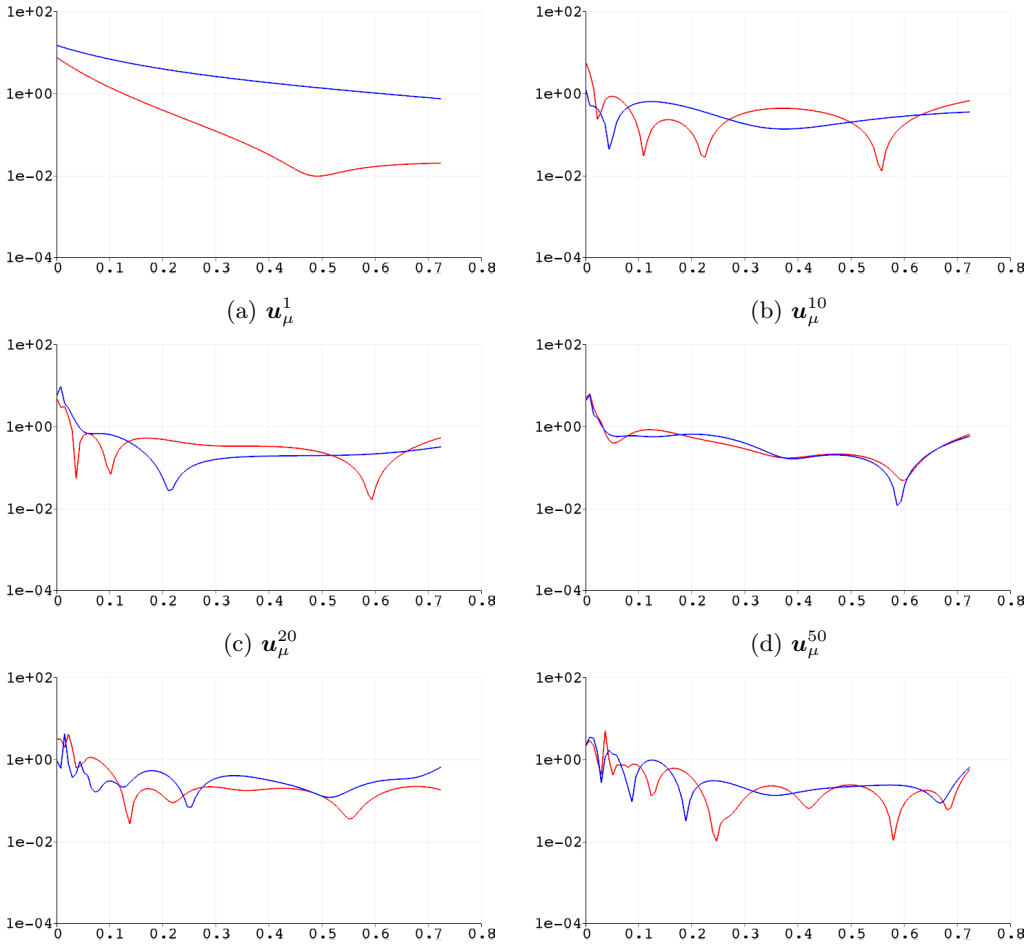


(a) $\boldsymbol{u}_\omega^1$ (b) $\boldsymbol{u}_\omega^{10}$

(c) $\boldsymbol{u}_\omega^{20}$ (d) $\boldsymbol{u}_\omega^{50}$

(e) $\boldsymbol{u}_\omega^{100}$ (f) $\boldsymbol{u}_\omega^{150}$

Figure 2.14: GTF calculation: real (blue) and imaginary (red) parts of the frequency modes.



(a) $\boldsymbol{u}_\mu^1$ (b) $\boldsymbol{u}_\mu^{10}$

(c) $\boldsymbol{u}_\mu^{20}$ (d) $\boldsymbol{u}_\mu^{50}$

(e) $\boldsymbol{u}_\mu^{100}$ (f) $\boldsymbol{u}_\mu^{150}$

Figure 2.15: GTF calculation: real (blue) and imaginary (red) parts of the damping modes.

of damping: $\mu_D = 8.04 \cdot 10^{-2}$s, which corresponds to blue curves, and $\mu_D = 40.20 \cdot 10^{-2}$s, which corresponds to red curves. Recall that μ_D is the damping coefficient of the material placed on the central part of the damper. It can be observed that bigger damping coefficients produce less displacement as well as a delay on the response. Fig. 2.16 demonstrates a good agreement between the reference and the PGD solution.

2.6 Nonlinear harmonic structural dynamics

The application of the frequency approach to solve nonlinear structural dynamics problems is, in general, quite complicated. Let us assume a nonlinear mechanical behaviour of the material such that the stiffness of the structure depends on the generalized displacement vector:

$$\boldsymbol{M}\ddot{\boldsymbol{u}}(t) + \boldsymbol{C}\dot{\boldsymbol{u}}(t) + \boldsymbol{f}_m[\boldsymbol{u}](t) = \boldsymbol{f}(t), \tag{2.36}$$

where $\boldsymbol{f}_m \in \mathbb{R}^N$ represents the vector of generalized internal, or mechanical, forces. Its nonlinear dependence on the generalized displacement vector is denoted in brackets. If Eq. (2.36) wants to be solved in the frequency domain, using the Fourier transform we obtain:

$$-\omega^2 \boldsymbol{M}\hat{\boldsymbol{u}}(\omega) + \mathrm{i}\omega \boldsymbol{C}\hat{\boldsymbol{u}}(\omega) + \hat{\boldsymbol{f}}_m[\boldsymbol{u}](\omega) = \hat{\boldsymbol{f}}(\omega), \tag{2.37}$$

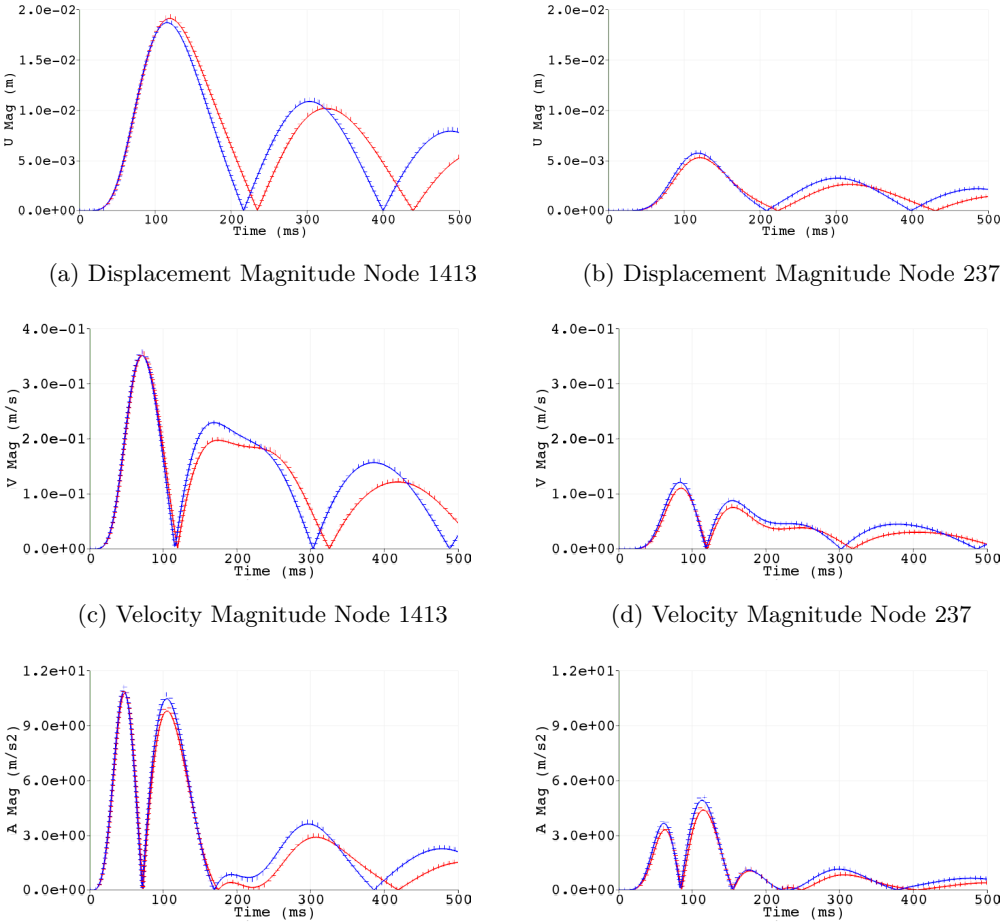
which of course precludes the use of harmonic superposition. In addition, the nonlinearity generates an energy exchange between the different harmonics which makes the problem to be coupled, that is, we cannot solve each harmonic amplitude independently. This is a very well-known issue which is classically addressed by coming back by evaluating the nonlinear term in the time domain rather than in the frequency domain. Of course, it has to be Fourier transformed to come back to the frequency domain. Numerically, this approach is not straightforward because the nonlinear term tends to transfer energy to the higher frequencies and thus, a wider frequency band has to be considered. Otherwise, *aliasing* may be experienced due to the high-frequency energetic content neglected, introducing numerical pollution in the low-frequencies. Anti-aliasing filters could of course be applied.

The fact of the nonlinear term introducing a coupling between the harmonics can be simply illustrated as follows. Consider a Newton linearization of the nonlinear problem:

$$\boldsymbol{M}\ddot{\boldsymbol{u}}(t) + \boldsymbol{C}\dot{\boldsymbol{u}}(t) + \boldsymbol{K}_T[\boldsymbol{u}](t)\boldsymbol{u}(t) = \boldsymbol{f}(t), \tag{2.38}$$

where $\boldsymbol{K}_T[\boldsymbol{u}](t)$ denotes the tangent stiffness matrix. Since it depends on the generalized displacement vector, the stiffness matrix is time-dependent. Recall that the following property of the Fourier transform: given two functions $f(t)$ and $g(t)$ that fulfill the requirements to be Fourier transformed, the transform of their product in the time domain becomes the convolution product of the transformed functions in the frequency domain. That is:

$$\mathcal{F}[f(t)\, g(t)] = \hat{f}(\omega) * \hat{g}(\omega) = \int_{-\infty}^{+\infty} \hat{f}(\omega - \nu) g(\nu)\, d\nu.$$



(a) Displacement Magnitude Node 1413

(b) Displacement Magnitude Node 237

(c) Velocity Magnitude Node 1413

(d) Velocity Magnitude Node 237

(e) Acceleration Magnitude Node 1413

(f) Acceleration Magnitude Node 237

Figure 2.16: Comparison of displacements, velocities and accelerations at the nodes 1413 (left) and 237 (right), for two different values of the parameter, $\mu_D = 8.04 \cdot 10^{-2}$s (red curves) and $\mu_D = 40.20 \cdot 10^{-2}$s (blue curves). Solid line corresponds to the reference solution while dotted line corresponds to the PGD solution.

Using this property, Eq. (2.38) becomes:

$$-\omega^2 \boldsymbol{M}\hat{\boldsymbol{u}}(\omega) + \mathrm{i}\omega \boldsymbol{C}\hat{\boldsymbol{u}}(\omega) + \int_{-\infty}^{+\infty} \boldsymbol{K}_T(\omega-\nu)\hat{\boldsymbol{u}}(\nu)\, d\nu = \hat{\boldsymbol{f}}(\omega),$$

which is clearly coupling every possible frequency. This fact, along with the impossibility of applying superposition, explain the fact that in this work we have not considered an extension of our frequency-based approach to nonlinear problems.

Remark 2.6 (*A special type of nonlinearity*). In some fields of engineering, such as in soil mechanics [30], a special kind of nonlinear formulation is used. It has been observed experimentally that when applying a harmonic excitation the soil response is still in the same frequency, but it cannot be accurately described using a linear model. In fact, superposition principle does not hold, i.e. the response to series of harmonics is not the superposition of the responses to each one of them. Models in which the nonlinearity only applies in the frequency domain are commonly used in the soil mechanics field. In consequence, the effect of the nonlinear behaviour is to modulate the amplitude of each harmonic. This approach, even if it is formally arguable, provides consistent results with regard to the experimental observations. PGD can be applied to solve this kind of nonlinear problem very efficiently. For instance, if the constitutive nonlinear behaviour makes both the shear modulus of the soil and its damping factor depend on the displacement amplitude [73, 100], a GTF in which both parameters are included as coordinates might be computed. From such GTF, a fast nonlinear solver can be built because each nonlinear iteration yields only particularizations of the GTF, which are costless. See [56] for further details on this approach.

Chapter 3

A frequency-based approach to overcome non-separability of moving thermal loads

Moving excitations can be thought as waves and therefore they are intrinsically non-separable in space and time. The classic PGD approach to transient problems, which consists in computing space-time separated representations, is highly penalized in this case simply due to the computational cost of integrating the right-hand side of the equation. In this Chapter, we present a frequency-based formulation that allows overcoming the aforementioned separability issues. This technique inherits many aspects from the frequency approach introduced in Chapter 2 and takes advantage of some additional properties that were not explored there. We focus on the transient heat equation, although the technique here presented could be applied to other problems, including structural dynamics.

The *reciprocity principle* will be proven thanks to the symmetrization introduced by the frequency formulation. This property yields a direct application for the real-time monitoring of thermal processes.

Contents

3.1 Monitoring thermal processes

Many thermal manufacturing processes require monitoring temperature and, moreover, being able to determine from these measurements if the process is running as designed. Typically these measurements are provided by some sensors (e.g. thermocouples) strategically placed. Consider, for instance, an industrial thermal process involving an external excitation, a heat source, moving on the surface of the considered part. This is typical of composite manufacturing processes. Today it is still a challenge to post-process these temperature measurements to monitor in real-time the correct evolution of the thermal process, or to control the heat source, or to identify defects, material properties or power oscillations, among others.

Real-time simulation-based control of thermal processes is a big challenge because high-fidelity numerical simulations are costly and cannot be used, in general, for real-time decision making. Very often, processes are monitored or controlled with a few measurements at some specific points. Thus, the strategy presented here is centered on fast evaluation of the response only where it is needed.

3.1.1 Time and frequency approaches for the heat equation

Representations in the frequency domain are appealing for analyzing responses of structures subjected to dynamic excitations, as it has been shown in Chapter 2. However, it is important to note that in spite of the large amount of scientific contributions using a frequency domain description in solid dynamics, this approach is not standard for thermal models subjected to dynamical forced thermal loads. Indeed, frequency domain approaches for thermal studies are scarce [54, 99, 110, 121, 123]. This is, of course, because time domain approximations of thermal models have been proven to be, in general, both efficient and robust. Moreover, a posteriori model order reduction has been successfully applied in this setting [23, 39, 57, 58, 95, 108, 113, 131]. Regarding a priori model reduction, PGD has also been successfully applied in several works to thermal problems formulated in the time domain; see [113] for instance.

However, time domain approximations cannot be applied so efficiently when travelling excitations are to be taken into account. Prulière et al. used in [113] a SVD to compute a space-time separated representation of the excitation. In order to keep a moderate computational cost, this approach required computing the SVD of a coarse description of the excitation, i.e. using a coarse mesh. Then, the space-time separated representation was projected back in a finer mesh, the one used to approximate the solution. Using 15 terms, the space-time separated representation of the excitation was approximated with an error of 0.03%. The error of the temperature field was in the order of 1%, which was accurate enough for the engineering purposes described that work.

An alternative approach was proposed in [39]. It consisted in adopting an Eulerian

approach in which the reference system was kept fixed to the thermal load and, therefore, the material moved at (constant) speed in the opposite sense. Two main hypothesis were made: the thermal load has to move at constant speed and the influence of the boundary conditions can be neglected. For those applications in which such hypothesis are acceptable, the Eulerian approach is simple and very effective. A final objection concerns the difficulty of considering both complex geometries and online corrections of the thermal load such as power oscillations or corrections of the orientation.

3.1.2 A new frequency-based PGD approach

The strategy that we propose in this Chapter revolves on the reciprocity principle [91] extensively used in mechanics, dynamics, electromagnetic or wave scattering problems. Note that, in general, this principle is not applicable for the heat equation because of the lack of symmetry introduced by the first time-derivative. This work proposes to adopt the reciprocity principle also to the heat equation. For this, the heat equation is recast in the frequency domain.

PGD combined with the reciprocity principle yield a fast strategy for real-time evaluation of the temperature at a specific point of a thermal system due to an arbitrary transient heat source travelling along a Neumann or Robin boundary. The strategy is this based on an offline and online phase. In the offline phase, PGD pre-computes a transfer function that encloses the output response of the system. This is achieved by solving the frequency-domain heat equation very much as shown in Chapter 2). This transfer function is computed offline and only once. The response at the monitoring point can be recovered performing a computationally inexpensive post-processing step. This last step can be performed online because it involves only a convolution between the generalized transfer function and a given arbitrary external thermal excitation. As it will be shown, the online approximation is so fast that it can be used for control purposes in real-time and on deployed devices.

The approach here presented has been published in [3], from where this Chapter is mostly withdrawn.

3.2 Monitoring temperature at a surface point

This section analyzes the representation of temperature at an arbitrary point of the boundary under an arbitrary transient external excitation, for instance, a travelling external heat source such as a moving laser. More specifically, the representation of the solution at the point of interest is first studied by means of a Green's function in the space-time domain. Then the applicability of the reciprocity principle for the transient heat equation with a forced excitation is discussed both in time and frequency domain. The use of reciprocity (only valid in the frequency domain) allows determining a representation of temperature at

the desired point for any arbitrary external heat source.

3.2.1 Model problem in the space-time domain

Formally, the problem under consideration is described as follows. Given a time interval $I :=]0, T[$ (T can be taken arbitrarily large) and a body $\Omega \subset \mathbb{R}^d, d \leq 3$, whose boundary $\partial\Omega$ is partitioned into Dirichlet, Γ_D, and Robin/Neumann, Γ_N, frontiers such that $\overline{\partial\Omega} = \overline{\Gamma}_D \cup \overline{\Gamma}_N$ and $\Gamma_D \cap \Gamma_N = \emptyset$; temperature evolution $u(\boldsymbol{x}, t)$, for $\boldsymbol{x} \in \Omega$ and $t \in I$, is described by the transient heat equation:

$$\begin{cases} \rho c_p \partial_t u - \nabla \cdot \mathbf{K} \nabla u = 0 & \text{in } \Omega \times I, \\ u = u_D & \text{on } \Gamma_D \times I, \\ \boldsymbol{n} \cdot \mathbf{K} \nabla u = -\ell(u - u_{\mathtt{ext}}) + q & \text{on } \Gamma_N \times I, \\ u = u_0 & \text{on } \Omega \times \{0\}, \end{cases} \tag{3.1}$$

where ρ is density (kg/m^3), c_p is specific heat capacity (J/(kg K)), $\mathbf{K}$ is the thermal conductivity matrix (W/(m K)), ℓ is the heat transfer coefficient (W/(m^2 K)), $u_{\mathtt{ext}}$ is the external temperature (K), $\boldsymbol{n}$ is the exterior unit normal to Γ_N (dimensionless) and $q = q(\boldsymbol{x}, t)$, for $(\boldsymbol{x}, t) \in \Gamma_N \times I$, is the inflow forcing excitation (W/m^2). The international system of units of measurement is also employed for length (m) and time (s).

As noted in the introduction, q is typically the heat flux imposed by a laser. The objective here is to determine a (fast) computable representation of the temperature at an arbitrary boundary point $\boldsymbol{x}_0 \in \Gamma_N$ and at any instant t_0, with $0 < t_0 < T$.

Since the problem is linear, Eq. (3.1) is further simplified. Thermal diffusivity (thermal conductivity divided by density and specific heat capacity) can be considered the only material constant in the partial differential equation. Moreover, the increment of temperature with respect to the external one, i.e. $(u - u_{\mathtt{ext}})$, can be defined as the unknown of the problem (in practice, impose $u_{\mathtt{ext}} = 0$).

Finally, for the clarity of the presentation, the model problem studied is further simplified. However, these simplifications (considering unitary values of the coefficients) do not compromise the validity of the following developments.

More precisely, the following assumptions are used to define the model problem: homogeneous Dirichlet boundary conditions, canonical dimensionless form with an isotropic homogeneous material, and no convective heat exchanges. Note however, that these simplifications, are only done to simplify the presentation and do not hinder the application of the proposed methodology to real problems described by Eq. (3.1) as it will be shown in

Section 3.5. Under these assumptions, Eq. (3.1) becomes

$$\begin{cases} \partial_t u - \nabla^2 u = 0 & \text{in } \Omega \times I, \\ u = 0 & \text{on } \Gamma_D \times I, \\ \boldsymbol{n} \cdot \nabla u = q & \text{on } \Gamma_N \times I, \\ u = u_0 & \text{on } \Omega \times \{0\}. \end{cases} \tag{3.2}$$

3.2.2 Green's function and reciprocity in the space-time domain

The objective is to obtain the solution at an arbitrary point and time, $(\boldsymbol{x}_0, t_0)$. Ideally the desired value $u(\boldsymbol{x}_0, t_0)$ could be readily evaluated if the adjoint Green's function were known. The adjoint Green's function is the solution of

$$\begin{cases} \partial_t G + \nabla^2 G = 0 & \text{in } \Omega \times]0, t_0[, \\ G = 0 & \text{on } \Gamma_D \times]0, t_0[, \\ \boldsymbol{n} \cdot \nabla G = \delta(\boldsymbol{x} - \boldsymbol{x}_0)\delta(t - t_0) & \text{on } \Gamma_N \times]0, t_0[, \\ G = 0 & \text{on } \Omega \times [t_0, T[, \end{cases} \tag{3.3}$$

where the notation of $G(\boldsymbol{x}, t; \boldsymbol{x}_0, t_0)$ clearly identifies the parametric dependence on $(\boldsymbol{x}_0, t_0)$. The representation of the desired temperature is then

$$u(\boldsymbol{x}_0, t_0) = \int_{\Gamma_N} \int_0^{t_0} G(\boldsymbol{x}, t; \boldsymbol{x}_0, t_0)\, q(\boldsymbol{x}, t) dt d\Gamma + \int_\Omega u_0(\boldsymbol{x})\, G(\boldsymbol{x}, 0; \boldsymbol{x}_0, t_0) d\Omega.$$

See Appendix B for a detailed presentation. However, in general, the computation of the Green's function is by no means a trivial task, for instance when confronted to an arbitrary domain or inhomogeneous material properties. Consequently, this approach is not used in practice.

An alternative is to use the reciprocity property [14, 91, 128]. However, it is also well-known that it is not applicable to the heat equation because the operator is not self-adjoint. In order to recall this, the variational problem equivalent to Eq. (3.2) is presented: find $u \in \mathcal{S}$ such that

$$B(u, v) = L(q; v) \quad \forall v \in \mathcal{V}, \tag{3.4}$$

with the appropriate spaces introducing the required regularity in space and time [107, 124]

$$\mathcal{V} := \big\{ v : v(\cdot, t) \in \mathcal{H}^1(\Omega),\, v(\boldsymbol{x}, \cdot) \in \mathcal{L}^2(I),\, v = 0 \text{ on } \Gamma_D \times I \big\} \cap \big\{ v : v(\cdot, t) \in \mathcal{H}^{-1}(\Omega),\, v(\boldsymbol{x}, \cdot) \in \mathcal{H}^1(I) \big\},$$

$$\mathcal{S} := \big\{ v : v \in \mathcal{V},\, v(\boldsymbol{x}, 0) = u_0 \big\},$$

and

$$B(u, v) = \int_\Omega \int_I v\, \partial_t u\, dt d\Omega + \int_\Omega \int_I \nabla u \cdot \nabla v\, dt d\Omega, \quad L(q; v) = \int_{\Gamma_N} \int_I q v\, dt d\Gamma.$$

Given two excitations q_1 and q_2, the corresponding solutions of Eq. (3.4) are denoted u_1 and u_2, respectively. Since both u_1 and u_2 belong to $\mathcal{S} \subset \mathcal{V}$ the following expressions are also verified:

$$B(u_1, u_2) = L(q_1; u_2) \text{ and } B(u_2, u_1) = L(q_2; u_1). \tag{3.5}$$

However, the bilinear form is non-symmetric because of the time derivative, i.e. $B(u, v) \neq B(v, u)$. Thus, subtracting both expressions in Eq. (3.5), standard reciprocity is not satisfied because the left hand side terms do not cancel out. In conclusion,

$$L(q_1; u_2) \neq L(q_2; u_1).$$

Therefore, in the space-time domain a Green's function approach or a reciprocity property cannot be used in practice to determine temperature at a surface point and instance, say $(\boldsymbol{x}_0, t_0) \in \Gamma_N \times I$.

3.2.3 Space-frequency problem for an arbitrary excitation

Another alternative for studying this forced excitation problem is to consider harmonic analysis. In order to work in the frequency domain the Fourier transform and its inverse are used, namely

$$\hat{v}(\boldsymbol{x}, \omega) = \mathcal{F}[v] = \int_{-\infty}^{+\infty} v(\boldsymbol{x}, t)\, e^{-i\omega t} dt \tag{3.6a}$$

and

$$v(\boldsymbol{x}, t) = \mathcal{F}^{-1}[\hat{v}] = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \hat{v}(\boldsymbol{x}, \omega)\, e^{i\omega t} d\omega. \tag{3.6b}$$

Remark 3.1 (*Fourier transform properties*). Fourier transforms have been largely studied and they hold a large number of properties (viz. linearity, translation, etc). In what follows it is important to recall that, in general, $\hat{v} \in \mathbb{C}$; but, for an even function in time v_e, i.e. $v_e(\boldsymbol{x}, -t) = v_e(\boldsymbol{x}, t)$, $\mathcal{F}[v_e] = \hat{v}_e \in \mathbb{R}$; whereas, for an odd function in time v_o, i.e. $v_o(\boldsymbol{x}, -t) = -v_o(\boldsymbol{x}, t)$, $\mathcal{F}[v_o]$ is imaginary, i.e. $\mathcal{F}[v_o] \in i\mathbb{R}$. For an odd function in time v_o, $\hat{v}_o$ is redefined as the imaginary part of $\mathcal{F}[v_o]$. Thus, $\mathcal{F}[v_o] = i\hat{v}_o$ with $\hat{v}_o \in \mathbb{R}$.

Applying the Fourier transform to Eq. (3.2), it becomes

$$\begin{cases} i\omega\hat{u} - \nabla^2\hat{u} = 0 & \text{in } \Omega, \\ \hat{u} = 0 & \text{on } \Gamma_D, \\ \boldsymbol{n} \cdot \nabla\hat{u} = \hat{q} & \text{on } \Gamma_N. \end{cases} \tag{3.7}$$

Remark 3.2 (*Long-term forced solution*). Note that the harmonic solution is only concerned with the long-term forced solution and consequently it does not depend on the initial condition. Standard approaches should be used for evaluating the transient regime, which, in practice, decays rapidly to the obtained long-term solution.

The variational form associated to the strong form problem described in Eq. (3.7) reads: find $\hat{u} \in \mathcal{H}^1_{\Gamma_D} := \{v \in \mathcal{H}^1(\Omega) : v = 0 \text{ on } \Gamma_D\}$ such that

$$\big(\nabla\hat{u}, \nabla\hat{v}\big) + i\omega\big(\hat{u}, \hat{v}\big) = \big\langle \hat{q}, \hat{v} \big\rangle \quad \forall \hat{v} \in \mathcal{H}^1_{\Gamma_D}, \tag{3.8}$$

where

$$\big(\hat{u}, \hat{v}\big) = \int_\Omega \hat{u}\,\hat{v}^* d\Omega \ , \ \big(\nabla\hat{u}, \nabla\hat{v}\big) = \int_\Omega \nabla\hat{u}\cdot\nabla\hat{v}^* d\Omega \text{ and } \big\langle \hat{u}, \hat{v} \big\rangle = \int_{\Gamma_N} \hat{u}\,\hat{v}^* d\Gamma \tag{3.9}$$

denote, respectively, the $\mathcal{L}^2$ scalar product of functions $\hat{u}$ and $\hat{v}$ and gradients in Ω and its traces over Γ_N. Note also, that $\hat{v}^*$ indicates the complex conjugate of $\hat{v}$, since both $\hat{u}$ and $\hat{v}$ are, in general, in $\mathbb{C}$.

It is important to observe that Eq. (3.8) is non-Hermitian but it is symmetric, the later property proves sufficient and also crucial for reciprocity.

3.2.4 Arbitrary excitation implies solving two problems with real excitation

Given any arbitrary excitation, $q(\boldsymbol{x}, t)$, it can always be decomposed in the sum of an even and odd function, namely

$$q(\boldsymbol{x}, t) = q_e(\boldsymbol{x}, t) + q_o(\boldsymbol{x}, t) = \frac{1}{2}\big(q(\boldsymbol{x}, t) + q(\boldsymbol{x}, -t)\big) + \frac{1}{2}\big(q(\boldsymbol{x}, t) - q(\boldsymbol{x}, -t)\big).$$

Recalling the Fourier transform properties: $\mathcal{F}[q] = \hat{q} = \mathcal{F}[q_e] + \mathcal{F}[q_o] = \hat{q}_e + i\hat{q}_o$ with $\hat{q}_e(\boldsymbol{x}, \omega) \in \mathbb{R}$ and $\hat{q}_o(\boldsymbol{x}, \omega) \in \mathbb{R}$. Moreover, it is important to notice that, the decomposition of $q(\boldsymbol{x}, t)$ in the sum of an even and odd excitation produces two identical problems with a real excitation whose solutions are the real and the imaginary part of the solution of Eq. (3.7). More precisely, because of the linearity of Eq. (3.7) and Remark 3.3, the solution $\hat{u}$ of Eq. (3.7) can be decomposed as $\hat{u} = \hat{u}_e + i\hat{u}_o$, with $\hat{u}_e$ and $\hat{u}_o$ solutions of

$$\begin{cases} i\omega\hat{u}_e - \nabla^2\hat{u}_e = 0 & \text{in } \Omega, \\ \hat{u}_e = 0 & \text{on } \Gamma_D, \\ \boldsymbol{n}\cdot\nabla\hat{u}_e = \hat{q}_e & \text{on } \Gamma_N, \end{cases} \text{ and } \begin{cases} i\omega\hat{u}_o - \nabla^2\hat{u}_o = 0 & \text{in } \Omega, \\ \hat{u}_o = 0 & \text{on } \Gamma_D, \\ \boldsymbol{n}\cdot\nabla\hat{u}_o = \hat{q}_o & \text{on } \Gamma_N. \end{cases} \tag{3.10}$$

Then, the original solution in the time domain can be recovered by means of the inverse Fourier transform, namely $u = \mathcal{F}^{-1}[\hat{u}] = \mathcal{F}^{-1}[\hat{u}_e] + \mathcal{F}^{-1}[i\hat{u}_o] = u_e + u_o$.

In summary, it is possible to find the solution of Eq. (3.7) —or Eq. (3.8)— for any arbitrary excitation $\hat{q}$ solving twice the same problem but with different *real* excitations, one corresponding to $\hat{q}_e(\boldsymbol{x},\omega)$ and the other to $\hat{q}_o(\boldsymbol{x},\omega)$.

Remark 3.3 (*Imaginary excitation*). Suppose u is solution of the following problem

$$\begin{cases} i\omega u - \nabla^2 u = 0 & \text{in } \Omega, \\ u = 0 & \text{on } \Gamma_D, \\ \boldsymbol{n}\cdot\nabla u = q & \text{on } \Gamma_N. \end{cases}$$

Then, $v = iu$ is solution of

$$\begin{cases} i\omega v - \nabla^2 v = 0 & \text{in } \Omega, \\ v = 0 & \text{on } \Gamma_D, \\ \boldsymbol{n}\cdot\nabla v = iq & \text{on } \Gamma_N. \end{cases}$$

Hint: replace $u = -iv$ in the first problem and obtain the second one. Obviously, this follows directly from linearity but it is explicitly recalled for didactic purposes.

3.2.5 Reciprocity in space-frequency holds for a real excitation

Given two real harmonic excitations $\hat{q}_1$ and $\hat{q}_2$, the corresponding solutions of Eq. (3.8) are denoted by $\hat{u}_1$ and $\hat{u}_2$, respectively. Then, since both solutions belong to $\mathcal{H}^1_{\Gamma_D}$, the following expressions hold:

$$\big(\nabla\hat{u}_1, \nabla\hat{u}_2\big) + i\omega\big(\hat{u}_1, \hat{u}_2\big) = \big\langle \hat{q}_1, \hat{u}_2 \big\rangle, \tag{3.11a}$$

$$\big(\nabla\hat{u}_2, \nabla\hat{u}_1\big) + i\omega\big(\hat{u}_2, \hat{u}_1\big) = \big\langle \hat{q}_2, \hat{u}_1 \big\rangle. \tag{3.11b}$$

Substracting Eq. (3.11b) from Eq. (3.11a) gives

$$\big(\nabla\hat{u}_1, \nabla\hat{u}_2\big) - \big(\nabla\hat{u}_2, \nabla\hat{u}_1\big) + i\omega\big[\big(\hat{u}_1, \hat{u}_2\big) - \big(\hat{u}_2, \hat{u}_1\big)\big] = \big\langle \hat{q}_1, \hat{u}_2 \big\rangle - \big\langle \hat{q}_2, \hat{u}_1 \big\rangle, \tag{3.12}$$

which clearly shows that reciprocity is satisfied in the frequency domain, namely

$$\big\langle \hat{q}_1, \hat{u}_2 \big\rangle = \big\langle \hat{q}_2, \hat{u}_1 \big\rangle, \tag{3.13}$$

if the following conditions hold

$$\big(\nabla\hat{u}_1, \nabla\hat{u}_2\big) = \big(\nabla\hat{u}_2, \nabla\hat{u}_1\big) \text{ and } \big(\hat{u}_1, \hat{u}_2\big) = \big(\hat{u}_2, \hat{u}_1\big). \tag{3.14}$$

This is precisely the case when $\hat{q}_1$ and $\hat{q}_2$ are real. See Appendix C for a detailed proof of Eq. (3.13).

Remark 3.4 (*Reciprocity with convective heat flux*). In the general case when convective heat fluxes are considered, see Eq. (3.1), reciprocity also holds. Linearity is exploited solving for $(u - u_{\mathtt{ext}})$ instead of the original temperature u. Accordingly, Eq. (3.11) are modified as follows:

$$\big(\nabla\hat{u}_1, \nabla\hat{u}_2\big) + i\omega\big(\hat{u}_1, \hat{u}_2\big) + \big\langle \ell\hat{u}_1, \hat{u}_2\big\rangle = \big\langle \hat{q}_1, \hat{u}_2\big\rangle,$$
$$\big(\nabla\hat{u}_2, \nabla\hat{u}_1\big) + i\omega\big(\hat{u}_2, \hat{u}_1\big) + \big\langle \ell\hat{u}_2, \hat{u}_1\big\rangle = \big\langle \hat{q}_2, \hat{u}_1\big\rangle.$$

Subtracting both equations shows that reciprocity also holds for convective heat because the same conditions described by Eq. (3.14) are obtained. Note that if Eq. (3.14) are verified then symmetry also hold for the traces, i.e. $\big\langle \hat{u}_1, \hat{u}_2\big\rangle = \big\langle \hat{u}_2, \hat{u}_1\big\rangle$.

3.2.6 Using reciprocity to monitor temperature

Recall that the final objective is to monitor temperature at a given point for an arbitrary external excitation, i.e. evaluate $u(\boldsymbol{x}_0, t_0)$ for $(\boldsymbol{x}_0, t_0) \in \Gamma_N \times I$. For this purpose, it is necessary to determine $\hat{u}(\boldsymbol{x}_0, \omega)$ for an arbitrary external excitation, $\hat{q}$, in the space-frequency domain. The conclusion of Section 3.2.4 is that two problems, which are shown in strong form by Eq. (3.10), with real excitations $\hat{q}_e$ and $\hat{q}_o$, such that $\hat{q} = \hat{q}_e + i\hat{q}_o$, must be solved to find $\hat{u}(\boldsymbol{x}_0, \omega) = \hat{u}_e(\boldsymbol{x}_0, \omega) + i\hat{u}_o(\boldsymbol{x}_0, \omega)$.

Suppose, that for each frequency, $\omega \in \mathbb{R}$, one could also determine the corresponding solution $\hat{h}(\boldsymbol{x}, \omega; \boldsymbol{x}_0)$ under a Dirac flux imposed at the monitoring point $\boldsymbol{x}_0$, $\delta(\boldsymbol{x} - \boldsymbol{x}_0)$, namely

$$\begin{cases} i\omega\hat{h} - \nabla^2\hat{h} = 0 & \text{in } \Omega, \\ \hat{h} = 0 & \text{on } \Gamma_D, \\ \boldsymbol{n}\cdot\nabla\hat{h} = \delta(\boldsymbol{x} - \boldsymbol{x}_0) & \text{on } \Gamma_N. \end{cases} \tag{3.15}$$

Since all excitations are real, reciprocity, see Eq. (3.13), holds and consequently

$$\big\langle \delta(\boldsymbol{x} - \boldsymbol{x}_0), \hat{u}_e\big\rangle = \big\langle \hat{q}_e, \hat{h}\big\rangle \text{ and } \big\langle \delta(\boldsymbol{x} - \boldsymbol{x}_0), \hat{u}_o\big\rangle = \big\langle \hat{q}_o, \hat{h}\big\rangle,$$

that is,

$$\hat{u}_e(\boldsymbol{x}_0, \omega) = \big\langle \hat{h}(\,\cdot\,, \omega; \boldsymbol{x}_0), \hat{q}_e(\,\cdot\,, \omega)\big\rangle \text{ and } \hat{u}_o(\boldsymbol{x}_0, \omega) = \big\langle \hat{h}(\,\cdot\,, \omega; \boldsymbol{x}_0), \hat{q}_o(\,\cdot\,, \omega)\big\rangle.$$

This implies, see Remark 3.5, that

$$\hat{u}(\boldsymbol{x}_0, \omega) = \int_{\Gamma_N} \hat{h}(\boldsymbol{x}, \omega; \boldsymbol{x}_0)\, \hat{q}(\boldsymbol{x}, \omega)\, d\Gamma. \tag{3.16}$$

Once temperature is monitored at the desired point in the space-frequency domain, the inverse Fourier transform is employed to obtain the desired final representation of temperature in the space-time domain:

$$u(\boldsymbol{x}_0, t) = \mathcal{F}^{-1}\big[\hat{u}(\boldsymbol{x}_0, \omega)\big] = \int_{\Gamma_N} \mathcal{F}^{-1}\big[\hat{h}(\boldsymbol{x}, \omega; \boldsymbol{x}_0)\, \hat{q}(\boldsymbol{x}, \omega)\big]\, d\Gamma.$$

This expression can be further simplified using the *convolution theorem* [49, 76], recall

$$\mathcal{F}^{-1}\big[\hat{h}(\boldsymbol{x},\omega;\boldsymbol{x}_0)\,\hat{q}(\boldsymbol{x},\omega)\big] = \int_0^t h(\boldsymbol{x},\tau;\boldsymbol{x}_0)\,q(\boldsymbol{x},t-\tau)d\tau,$$

where

$$h(\boldsymbol{x},\tau;\boldsymbol{x}_0) = \mathcal{F}^{-1}[\hat{h}](\boldsymbol{x},\tau;\boldsymbol{x}_0) = \frac{1}{2\pi}\int_{-\infty}^{+\infty}\hat{h}(\boldsymbol{x},\omega;\boldsymbol{x}_0)\,e^{i\omega t}d\omega. \tag{3.17}$$

Thus, the representation of the temperature at the desired point $\boldsymbol{x}_0 \in \Gamma_N$ is

$$u(\boldsymbol{x}_0,t) = \int_{\Gamma_N}\int_0^t\left(\frac{1}{2\pi}\int_{-\infty}^{+\infty}\hat{h}(\boldsymbol{x},\omega;\boldsymbol{x}_0)\,e^{i\omega t}d\omega\right)q(\boldsymbol{x},t-\tau)\,d\tau d\Gamma,$$

which can be written in a more compact form and for any instance $t_0 \in I =]0,T[$ as

$$u(\boldsymbol{x}_0,t_0) = \int_0^{t_0}\big\langle h(\,\cdot\,,\tau;\boldsymbol{x}_0), q(\,\cdot\,,t_0-\tau)\big\rangle\,d\tau. \tag{3.18}$$

Recall that $q \in \mathbb{R}$.

This is a compact and useful expression, it only requires knowledge of the external imposed excitation q up to the desired monitored instant t_0 (causality). Moreover, and this is a major point and advantage, Eq. (3.18) does not reflect the decomposition of the excitation in even and odd contributions. Thus, if the transfer function, $h(\boldsymbol{x},\tau;\boldsymbol{x}_0)$, is known, Eq. (3.18) can be applied directly for any arbitrary excitation $q(\boldsymbol{x},t)$.

However this expression also presents a major drawback: the inverse Fourier transform of $\hat{h}$, solution of Eq. (3.15), must be known. This implies solving Eq. (3.15) for every frequency ω in the range needed by the arbitrary excitation. Thus, in general, the representation Eq. (3.18) cannot be used in practice.

The next section circumvents this drawback and proposes a methodology to obtain an expression for the generalized transfer function $\hat{h}$, solution of Eq. (3.15), for the all range of realistic frequencies. This expression can then be substituted in Eq. (3.18) to determine the desired temperature. Moreover, Eq. (3.18) can be evaluated in real-time.

Remark 3.5 (*Reconstruction of solution*). To determine Eq. (3.16) it is important to recall that $\hat{q}_e$ and $\hat{q}_o$ are real. Thus, the scalar products on Γ_N,

$$\hat{u}(\boldsymbol{x}_0,\omega) = \hat{u}_e(\boldsymbol{x}_0,\omega) + i\hat{u}_o(\boldsymbol{x}_0,\omega) = \big\langle\hat{h}(\,\cdot\,,\omega;\boldsymbol{x}_0),\hat{q}_e(\,\cdot\,,\omega)\big\rangle + i\big\langle\hat{h}(\,\cdot\,,\omega;\boldsymbol{x}_0),\hat{q}_o(\,\cdot\,,\omega)\big\rangle,$$

can be rewritten as

$$\begin{aligned}\hat{u}(\boldsymbol{x}_0,\omega) &= \int_{\Gamma_N}\hat{h}(\boldsymbol{x},\omega;\boldsymbol{x}_0)\,\hat{q}_e(\boldsymbol{x},\omega)\,d\Gamma + i\int_{\Gamma_N}\hat{h}(\boldsymbol{x},\omega;\boldsymbol{x}_0)\,\hat{q}_o(\boldsymbol{x},\omega)\,d\Gamma\\ &= \int_{\Gamma_N}\hat{h}(\boldsymbol{x},\omega;\boldsymbol{x}_0)\big(\hat{q}_e(\boldsymbol{x},\omega) + i\hat{q}_o(\boldsymbol{x},\omega)\big)d\Gamma = \int_{\Gamma_N}\hat{h}(\boldsymbol{x},\omega;\boldsymbol{x}_0)\,\hat{q}(\boldsymbol{x},\omega)\,d\Gamma,\end{aligned}$$

and do not present any complex conjugate. In fact, it is important to note that

$$\hat{u}(\boldsymbol{x}_0,\omega) = \int_{\Gamma_N}\hat{h}(\boldsymbol{x},\omega;\boldsymbol{x}_0)\,\hat{q}(\boldsymbol{x},\omega)\,d\Gamma = \big\langle\hat{h}(\,\cdot\,,\omega;\boldsymbol{x}_0),\hat{q}^*(\,\cdot\,,\omega)\big\rangle \neq \big\langle\hat{h}(\,\cdot\,,\omega;\boldsymbol{x}_0),\hat{q}(\,\cdot\,,\omega)\big\rangle.$$

Remark 3.6 (*Inverse Fourier Transform of the generalized transfer function*). The inverse fourier transform of the *generalized transfer function*, $\hat{h}(\boldsymbol{x}, \omega; \boldsymbol{x}_0)$, see Eq. (3.17) is computed using the FFT algorithm. Only periodic signals with a finite number of harmonics can be exactly represented with the discrete Fourier transform, and thus non-periodic signals involving a continuous spectrum of frequencies can only be approximated. The range of frequencies included in the generalized transfer function must be chosen accordingly.

Remark 3.7 (*Convective heat flux*). In the general case described in Eq. (3.1), when convective fluxes are present, the generalized transfer function problem originally described by Eq. (3.15) is modified. As noted in Remark 3.4, reciprocity also holds when convective fluxes are considered. Thus, the generalized transfer function must have on the Neumann boundary the convective heat flux, namely

$$\begin{cases} i\omega\hat{h} - \nabla^2\hat{h} = 0 & \text{in } \Omega, \\ \hat{h} = 0 & \text{on } \Gamma_D, \\ \boldsymbol{n}\cdot\nabla\hat{h} = -\ell\hat{h} + \delta(\boldsymbol{x}-\boldsymbol{x}_0) & \text{on } \Gamma_N. \end{cases}$$

Following the procedure described previously, Eq. (3.16) and Eq. (3.18) also hold.

3.3 Computing the generalized transfer function

This section is aimed at computing a generalized transfer function, $\hat{h}(\boldsymbol{x}, \omega; \boldsymbol{x}_0)$, for a desired and predefined range of frequencies, I_ω. This transfer function needs to be computed only once, and preferably offline. Since it is determined in the frequency domain, its inverse Fourier transform is later evaluated in order to use Eq. (3.18) as a simple and inexpensive post-process of any given excitation $q(\boldsymbol{x}, t)$.

As it has been shown in Chapter 1, major contribution of the PGD approach is to view frequency, ω, as a new coordinate [40]. Thus, instead of solving Eq. (3.15) for each frequency, the objective is to solve, only once, a more general problem with ω as an extra coordinate, namely find $\hat{h}(\boldsymbol{x}, \omega; \boldsymbol{x}_0)$ satisfying

$$\begin{cases} i\omega\hat{h} - \nabla^2\hat{h} = 0 & \text{in } \Omega \times I_\omega, \\ \hat{h} = 0 & \text{on } \Gamma_D \times I_\omega, \\ \boldsymbol{n}\cdot\nabla\hat{h} = \delta(\boldsymbol{x}-\boldsymbol{x}_0) & \text{on } \Gamma_N \times I_\omega, \end{cases} \tag{3.19}$$

where I_ω is the predefined range of variation of ω. The weak problem equivalent to Eq. (3.19) is obtained using a weighted residual argument, namely, find $\hat{h}$ for all $\hat{v}$ in the selected appropriate functional space such that

$$A(\hat{h}, \hat{v}) = L(\hat{v}) \tag{3.20a}$$

with

$$A(\hat{h},\hat{v}) := \int_{I_\omega} (\nabla\hat{h},\nabla\hat{v})d\omega + \int_{I_\omega} i\omega(\hat{h},\hat{v})d\omega \tag{3.20b}$$

$$L(\hat{v}) := \int_{I_\omega} \hat{v}^*(\boldsymbol{x}_0,\omega)d\omega. \tag{3.20c}$$

Note that formally, the required functional spaces must account for the singularity of the Dirac flux. Nevertheless, in practice, the Dirac delta is mollified and this allows to use the standard finite element functional setup.

The PGD approach assumes that the solution of Eq. (3.20), $\hat{h}(\boldsymbol{x},\omega;\boldsymbol{x}_0)$, can be approximated by a rank-n separable function, $\hat{h}^n(\boldsymbol{x},\omega;\boldsymbol{x}_0)$, namely,

$$\hat{h}(\boldsymbol{x},\omega;\boldsymbol{x}_0) \approx \hat{h}^n(\boldsymbol{x},\omega;\boldsymbol{x}_0) = \sum_{s=1}^{n} X^s(\boldsymbol{x})\, W^s(\omega) = \hat{h}^{n-1}(\boldsymbol{x},\omega;\boldsymbol{x}_0) + X^n(\boldsymbol{x})\, W^n(\omega), \tag{3.21}$$

where $X^s \in \mathcal{H}^1_{\Gamma_D}$ and $W^s \in \mathcal{L}^2(I_\omega)$ for $s = 1,\dots,n$. Recall that these functions give values in $\mathbb{C}$.

A standard rank-one corrections (Greedy) PGD algorithm [88] is used to construct this approximation, that is, to determine the unknown functions X^s and W^s in Eq. (3.21). The sequence is stopped with an appropriate error estimator [4, 87, 98]. Since each new term implies the computation of a product of unknown functions, X^n and W^n, a nonlinear scheme must be designed. The standard choice is the alternating directions algorithm because it has proven robust in former works [38, 40, 106]. To simplify notation each iterate approximating X^n and W^n is denoted by R and S. Hence, the nonlinear problem to solve for each new term of $\hat{h}^n(\boldsymbol{x},\omega;\boldsymbol{x}_0)$ is obtained substituting Eq. (3.21) in Eq. (3.20a), in order to compute R and S (iterates of X^n and W^n) such that

$$A(R\,S,\hat{v}) = L(\hat{v}) - A(\hat{h}^{n-1},\hat{v}) \tag{3.22}$$

with trial functions on the tangent manifold

$$\hat{v} = \hat{v}_R(\boldsymbol{x})\, S(\omega) + R(\boldsymbol{x})\, \hat{v}_S(\omega) \quad \forall \hat{v}_R \in \mathcal{H}^1_{\Gamma_D} \text{ and } \forall \hat{v}_S \in \mathcal{L}^2(I_\omega).$$

The alternating direction scheme, detailed below, consists in, for instance, updating the space function R from a given S assumed known, and then compute S from the just updated function R. This iteration continues until reaching convergence of both R and S. That is, the two stages for each iteration are:

1. Find $R \in \mathcal{H}^1_{\Gamma_D}$ (S assumed known) such that

$$A(R\,S,\hat{v}_R\,S) = L(\hat{v}_R\,S) - A(\hat{h}^{n-1},\hat{v}_R\,S) \quad \forall \hat{v}_R \in \mathcal{H}^1_{\Gamma_D}. \tag{3.23a}$$

2. Find $S \in \mathcal{L}^2(I_\omega)$ (R assumed known) such that

$$A(R\,S,R\,\hat{v}_S) = L(R\,\hat{v}_S) - A(\hat{h}^{n-1},R\,\hat{v}_S) \quad \forall \hat{v}_S \in \mathcal{L}^2(I_\omega). \tag{3.23b}$$

Then at convergence, X^n and W^n are updated by R and S.

3.3.1 Updating the space function

For each new term in the series defined by Eq. (3.21) and each iteration described by Eq. (3.23), Eq. (3.23a) must be solved. Taking advantage of the separated structure of the solution and also of $A(\,\cdot\,,\,\cdot\,)$, see Eq. (3.20b), Eq. (3.23a) can be rewritten as, find $R \in \mathcal{H}^1_{\Gamma_D}$ for all $\hat{v}_R \in \mathcal{H}^1_{\Gamma_D}$ (S assumed known) such that

$$\alpha^S\big(\nabla R, \nabla \hat{v}_R\big) + i\beta^S\big(R, \hat{v}_R\big) = \gamma^S \hat{v}_R^*(\boldsymbol{x}_0) - \sum_{s=1}^{n-1} \alpha_s^S\big(\nabla X^s, \nabla \hat{v}_R\big) + i\beta_s^S\big(X^s, \hat{v}_R\big), \tag{3.24}$$

where the coefficients, which must be computed for each instance of S, are defined as

$$\alpha^S = \big(S, S\big)_{I_\omega},\ \beta^S = \big(\omega\, S, S\big)_{I_\omega},\ \gamma^S = \big(1, S\big)_{I_\omega},$$
$$\alpha_s^S = \big(W^s, S\big)_{I_\omega}, \text{ and } \beta_s^S = \big(\omega\, W^s, S\big)_{I_\omega}.$$

Recall that $\big(\,\cdot\,,\,\cdot\,\big)_{I_\omega}$ denotes the $\mathcal{L}^2$ scalar product of complex functions in I_ω.

After the corresponding discretization of the spatial domain with a standard combination of piecewise linear shape functions, the system of linear equations induced by Eq. (3.24) presents a conductivity and a mass matrix. These matrices are computed only once because they are constant for each iteration and for each term. Moreover, it is important to note that they are symmetric but non-Hermitian, which will preclude, for instance, Cholesky or conjugate gradient schemes.

3.3.2 Updating the frequency function

Similarly, the second stage, described by Eq. (3.23b), can also be rewritten using the separated structure of the solution and also of $A(\,\cdot\,,\,\cdot\,)$ as, find $S \in \mathcal{L}^2(I_\omega)$ for all $\hat{v}_S \in \mathcal{L}^2(I_\omega)$ (R assumed known from the previous stage) such that

$$\alpha^R\big(S, \hat{v}_S\big)_{I_\omega} + i\beta^R\big(\omega\, S, \hat{v}_S\big)_{I_\omega} = \gamma^R\big(1, \hat{v}_S\big)_{I_\omega} - \sum_{s=1}^{n-1} \alpha_s^R\big(W^s, \hat{v}_S\big)_{I_\omega} + i\beta_s^R\big(\omega\, W^s, \hat{v}_S\big)_{I_\omega}, \tag{3.25}$$

where the coefficients, which must be computed for each instance of R, are defined as $\mathcal{L}^2$ products over the spatial domain,

$$\alpha^R = \big(\nabla R, \nabla R\big),\ \beta^R = \big(R, R\big),\ \gamma^R = R^*(\boldsymbol{x}_0),$$
$$\alpha_s^R = \big(\nabla X^s, \nabla R\big), \text{ and } \beta_s^R = \big(X^s, R\big).$$

The lack of derivatives with respect to ω in Eq. (3.19) induces an point-wise algebraic equation for S. Piecewise discontinuous approximations of S will induce uncoupled scalar equations. Whereas continuous approximations over the one-dimensional range of frequencies lead to a symmetric but non-Hermitian matrix on the left-hand-side of Eq. (3.25), as in the previous case.

3.4 Extension to multi-parametric and inverse problems

The approach presented here has a potentiality that exceeds real-time monitoring of temperature at a given location and can also be used for other thermal studies such as optimization, inverse analysis, nondestructive testing, etc. Here, two simple extensions are presented.

3.4.1 Multi-parametric models

As a simple demonstrator, thermal conductivity is chosen as an extra parameter. The underlying idea, already exploited in [40, 112], is to solve multi-parametric models capitalizing the advantages of the PGD framework. A multi-parametric model is an extension of the procedure detailed in the previous section. Besides frequency as an extra-coordinate the generalized transfer function can encompass other parameters as extra coordinates. For instance, parameters characterizing the geometry, the constitutive behavior or the boundary conditions could be incorporated. The PGD methodology allows to compute efficiently a multi-parametric solution defined in a high-dimensional space (spatial coordinates, frequency and other parameters). Multi-parametric models are of great interest in science and engineering because they make possible real-time simulation, optimization and inverse analysis, as illustrated in [6, 40].

For instance, the thermal example that motivates this work can also be posed as an inverse analysis to find the actual conductivity of a certain material from the temperature provided by a thermocouple placed at location $\boldsymbol{x}_0$. Then, independently of the inverse identification method used, it is obvious that fast identification procedures can be envisaged if an approximation of temperature at the monitoring point $\boldsymbol{x}_0$ for any instance t can be computed in real-time for any conductivity k, i.e. $u(\boldsymbol{x}_0, t, k)$.

This approach can also be used for nondestructive testing. The monitored temperature being different from the computed one (obtained with the undamaged material parameters) triggers the inverse analysis to determine the "damaged" material parameter, which can be solved readily because a generalized solution for any material parameter is available.

The inverse problem is not solved in detail at this point because it is outside the scope of this work. Nevertheless, here, the generalized solution for any conductivity is provided. Once this solution is known, any standard inverse algorithm could be implemented. The key point is to determine the generalized transfer function, $\hat{h}$, for any value of the conductivity $k \in I_k$, where I_k is the desired range of conductivities. Eq. (3.19) is now rewritten with the new parameter, conductivity, considered as an extra-coordinate. Thus the new problem consist in finding $\hat{h}(\boldsymbol{x}, \omega, k; \boldsymbol{x}_0)$ that satisfies

$$\begin{cases} i\omega\hat{h} - \nabla \cdot (k\nabla\hat{h}) = 0 & \text{in } \Omega \times I_\omega \times I_k, \\ \hat{h} = 0 & \text{on } \Gamma_D \times I_\omega \times I_k, \\ \boldsymbol{n} \cdot k\nabla\hat{h} = \delta(\boldsymbol{x} - \boldsymbol{x}_0) & \text{on } \Gamma_N \times I_\omega \times I_k. \end{cases}$$

The PGD approach must now determine an approximation $\hat{h}^n(\boldsymbol{x}, \omega, k; \boldsymbol{x}_0)$ to the solution of the previous problem, namely

$$\hat{h}(\boldsymbol{x}, \omega, k; \boldsymbol{x}_0) \approx \hat{h}^n(\boldsymbol{x}, \omega, k; \boldsymbol{x}_0) = \sum_{s=1}^{n} X^s(\boldsymbol{x})\, W^s(\omega)\, K^s(k),$$

where extra separated functions must be determined; more precisely, those directly linked to conductivity, namely $K^s(k)$ for $s = 1, \ldots, n$. The same greedy approach described earlier can be applied with now an extra stage in the nonlinear solve to determine each $K^s(k)$. Once the multi-parametric solution has been computed, it can be postprocessed in the same way explained in Section 3.2.6 in order to recover the solution at $\boldsymbol{x}_0$ and any time t. The objective is that temperature at the monitoring point $\boldsymbol{x}_0$ for any instance t can be computed in real-time for any conductivity k; that is, Eq. (3.18) is extended to approximate $u(\boldsymbol{x}_0, t, k)$, namely

$$u(\boldsymbol{x}_0, t, k) = \mathcal{F}^{-1}[\hat{u}](\boldsymbol{x}_0, t, k) = \int_0^t \left\langle h(\,\cdot\,, \tau, k; \boldsymbol{x}_0), q(\,\cdot\,, t_0 - \tau) \right\rangle d\tau, \tag{3.26}$$

where $h = \mathcal{F}^{-1}[\hat{h}]$.

As noted earlier, the crucial point is to determine a reasonable approximation of the generalized transfer function $\hat{h}(\boldsymbol{x}, \omega, k; \boldsymbol{x}_0)$. The example presented in Section 3.5 is proposed to demonstrate that such an approximation can be evaluated.

3.4.2 Inverse problem: an amplitude time-modulated calibration of excitation

This inverse problem considers that the amplitude of the power given by a laser varies with time because of uncontrolled power supply and has to be calibrated. Suppose an excitation defined by $\alpha(t)q(\boldsymbol{x}, t)$ where, as assumed in previous sections, $q(\boldsymbol{x}, t)$ is given (and, thus, known) while its amplitude, which varies with time $\alpha(t)$ is not know. Temperature at the monitoring point $\boldsymbol{x}_0$ for any instance t is obtained following the procedure described in Section 3.2.4 for this new excitation, Eq. (3.18) becomes

$$u(\boldsymbol{x}_0, t) = \int_0^t \alpha(t_0 - \tau) \int_{\Gamma_N} h(\boldsymbol{x}, \tau; \boldsymbol{x}_0)\, q(\boldsymbol{x}, t - \tau)\, d\Gamma d\tau.$$

Then, given a discretization of the unknown function $\alpha(t)$, for instance

$$\alpha(t) = \sum_{j=1}^{n_{\text{fit}}} \alpha_j N_j(t), \tag{3.27}$$

where $N_j(t)$ are known interpolation functions, the coefficients $\boldsymbol{\alpha} \in \mathbb{R}^{n_{\text{fit}}}$ can be determined by a least-squares technique. This implies solving the normal equations $\mathbf{A}\boldsymbol{\alpha} = \mathbf{b}$.

Given the series of instants $\{t_1, t_2, \ldots, t_m\}$ (with $m \geq n_{\text{fit}}$) at which temperature is going to be measured, the matrix of the normal equations is determined once and for all, during the *offline* phase, as

$$\mathbf{A} = [a_{ij}] = \left[\sum_{r=1}^{m} \psi_i(t_r)\psi_j(t_r)\right]$$

with

$$\psi_i(t_r) = \int_0^{t_r} N_i(t_r - \tau) \int_{\Gamma_N} h(\boldsymbol{x}, \tau; \boldsymbol{x}_0)\, q(\boldsymbol{x}, t_r - \tau)\, d\Gamma d\tau.$$

Then, the measured values of temperature, $u^{\texttt{meas}}(\boldsymbol{x}_0, t_r)$, at $\boldsymbol{x}_0$ and for the series of instants $\{t_1, t_2, \ldots, t_m\}$ allow to compute, in the *online* phase, the independent term

$$\mathbf{b} = [b_i] = \left[\sum_{r=1}^{m} \psi_i(t_r) u^{\texttt{meas}}(\boldsymbol{x}_0, t_r)\right].$$

Finally the normal equations $\mathbf{A}\boldsymbol{\alpha} = \mathbf{b}$ are solved.

3.5 Numerical examples

3.5.1 Single-ply composite cylinder: verification of the proposed methodology

Aiming to demonstrate the ability of the proposed method to monitor transient models, a 2D problem, which involves a heat flux moving over the outer boundary of a cylinder, is proposed. The outer boundary is also subjected to heat convection while the other boundaries are adiabatic. Fig. 3.1 depicts the problem statement. The initial boundary value problem is described as

$$\begin{cases} \rho c_p \partial_t u - \nabla \cdot k \nabla u = 0 & \text{in } \Omega \times I, \\ \boldsymbol{n} \cdot k \nabla u = -\ell(u - u_{\texttt{ext}}) + q & \text{on } \Gamma_{\texttt{Out}} \times I, \\ \boldsymbol{n} \cdot k \nabla u = 0 & \text{on } \partial\Omega / \Gamma_{\texttt{Out}} \times I, \\ u = u_0 & \text{on } \Omega \times \{0\}, \end{cases} \tag{3.28}$$

where $\rho = 1\text{kg/m}^3$ is density, $c_p = 1\text{J/(kg K)}$ is specific heat capacity, $k = 1$ W/(m K) is isotropic thermal conductivity and $\ell = 1\text{W/(m}^2\text{ K)}$ is the heat transfer coefficient, $u_{\texttt{ext}} = 298\text{K}$ is the external temperature, $\Gamma_{\texttt{Out}}$ is the outer boundary where the laser impacts and with a radius of 1.0m,

$$q(\xi, t) = 500 \exp\big(-50(2\xi - \pi t)^2\big)\text{W/m}^2 \tag{3.29}$$

is the inflow forcing excitation, and ξ is the local tangent coordinate along $\Gamma_{\texttt{Out}}$. The temperature is measured (i.e. the point where temperature is monitored) at the middle point of the inner boundary. Finally, the thickness of the single-ply is 0.05m.

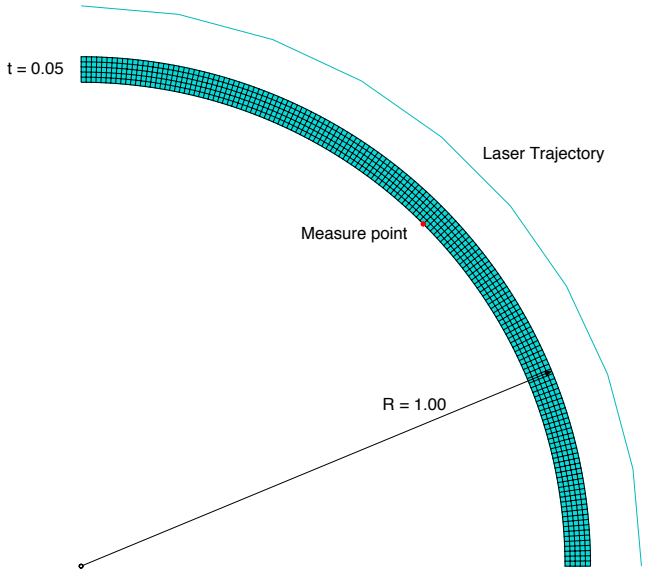


Figure 3.1: Single-ply composite cylinder: problem statement.

The range of frequencies considered is $f \in [-250, 250]$ Hz or, in terms of the angular frequency used in all previous sections, $\omega \in I_\omega := [-500\pi, 500\pi]$. Such a large interval of frequencies has been chosen in order to be able to make a fair comparison with a reference solution. The time-step of a signal and the maximum frequency that can be computed from it are related by the Nyquist-Shannon theorem [122] as follows:

$$f_{\texttt{max}} = 1/2\Delta t. \tag{3.30}$$

The finite element (FE) reference solution is obtained with a standard time-marching Crank-Nicolson scheme whose time-step is chosen for accuracy considerations. In practice, a time-step of 2ms is accurate enough, and thus the maximum frequency to be considered is 250 Hz. With that frequency, the time signal recovered after performing the inverse Fourier transform has the same time-step as the FE reference solution.

The range $[-250, 250]$ Hz is clearly an overkill because the frequency range of the imposed heat flux q is in $[-20, 20]$ Hz. This later range is determined because a Fourier transform of the heat flux seen by a point on $\Gamma_{\texttt{Out}}$ reveals that harmonics of frequency greater than 20 Hz transfer a negligible amount of energy to the system.

Moreover, note that for a particular negative frequency, the transfer function must be the complex conjugate of its symmetric (positive) counterpart. This is also a consequence of the Nyquist-Shannon theorem. Here negative frequencies are also computed to verify numerically that the PGD method reproduces a symmetric real part and an anti-symmetric imaginary part.

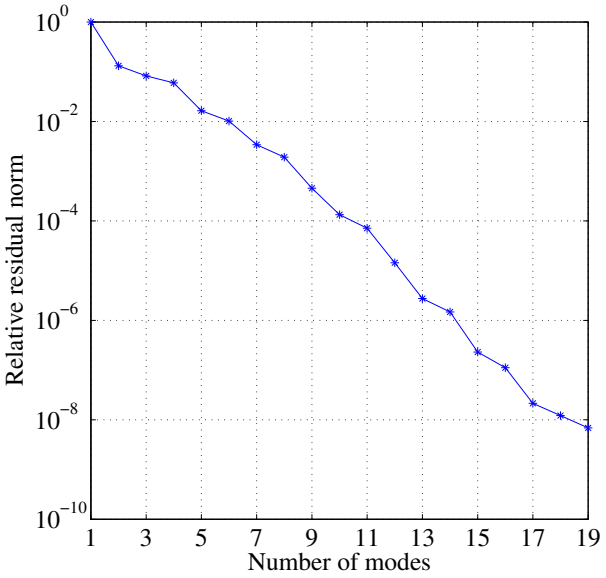


Figure 3.2: Single-ply composite cylinder: convergence of the generalized frequency transfer function.

As discussed in Section 3.3 and depicted in Eq. (3.21), PGD is used to determine an approximation, say $\hat{h}^n$, of the transfer function, $\hat{h}$, solution of Eq. (3.19). In fact, Fig. 3.2 shows the relative residue of Eq. (3.20) as the number of modes n increases. The normalized residue is computed using the $\mathcal{L}^2$ norm of the discrete residue normalized by the norm of the right-hand-side of Eq. (3.20a).

For demonstration purposes tolerances are taken small, beyond engineering accuracy. However, 19 terms induce negligible (below 10^{-8}!) normalized relative residues. The average number of fixed-point iterations is 25 (same tolerance of 10^{-8}). Since 19 terms are necessary to reduce the residual norm below 10^{-8}, the total amount of FE solves in the offline stage is around 475.

Note that spatial modes and frequency modes are localized. Fig. 3.3 and Fig. 3.4 show respectively the first three spatial and frequency modes. These modes are X^1, W^1, X^2, W^2, X^3 and W^3. Since all of them are complex, the real part and the imaginary part are depicted. As expected, the real part of the frequency modes is symmetric, while the imaginary part is anti-symmetric.

Notice that a uniform spatial discretization is used with 770 bilinear quadrilateral elements of size 0.01m. This implies 930 nodes with scalar complex unknowns. A non-uniform discretization is used for frequency because it varies more rapidly near the origin. The mesh consists of 500 $\mathcal{C}^0$-continuous linear elements refined around the zero-frequency using a cubic polynomial ω^3 for the element length. Since frequency modes do not involve any derivative

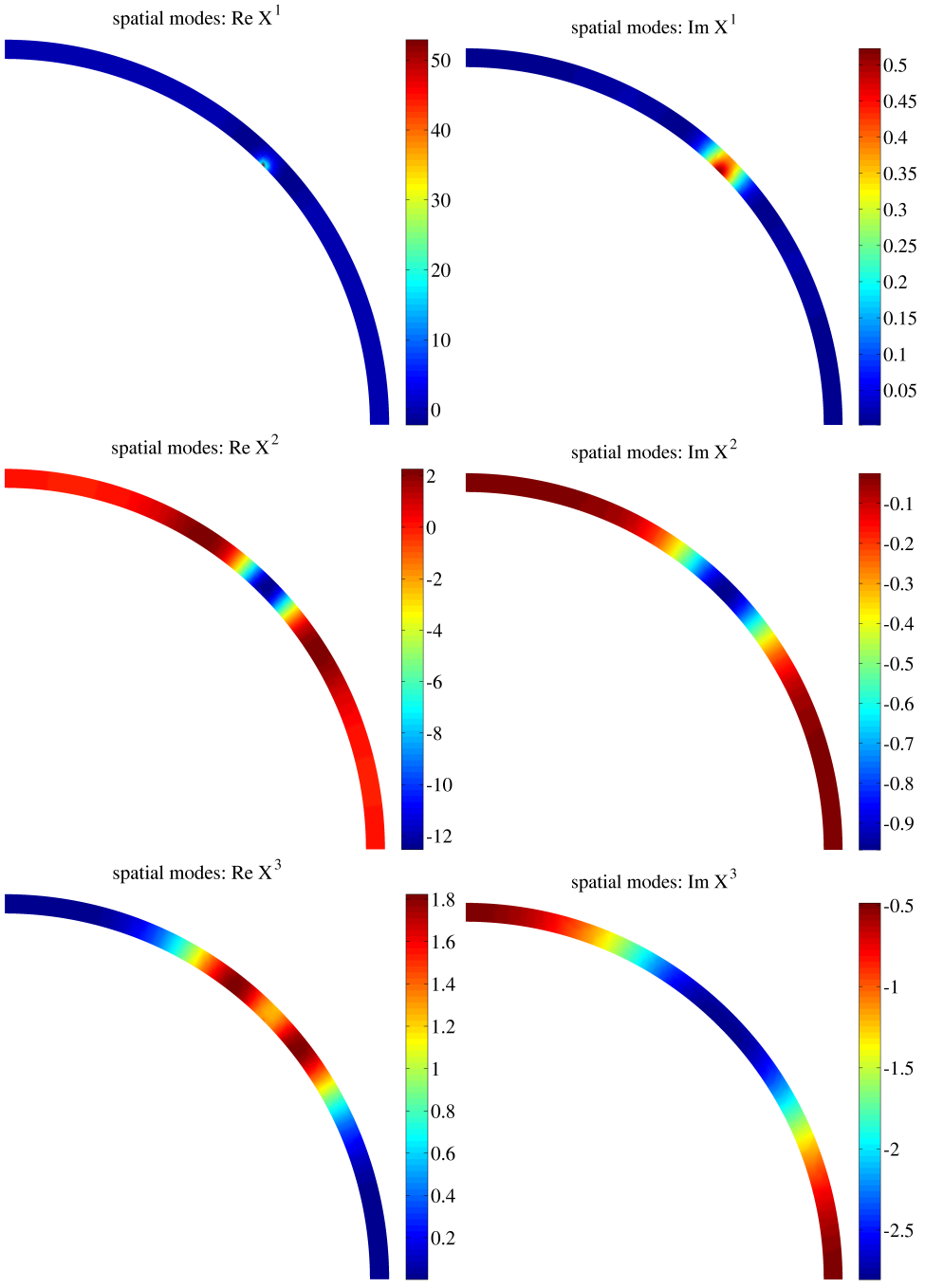


Figure 3.3: Single-ply composite cylinder: first 3 PGD spatial modes real (left) and imaginary (right).

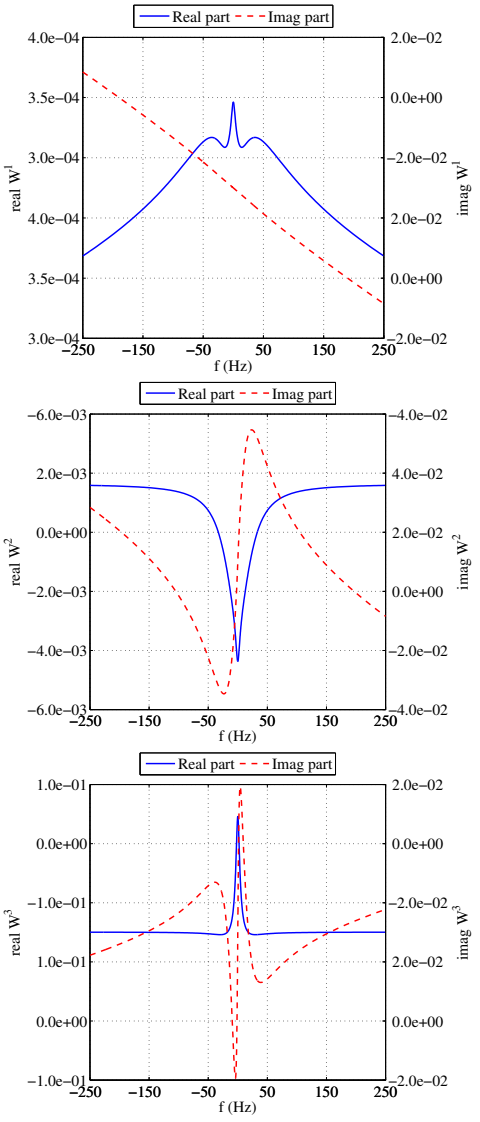


Figure 3.4: Single-ply composite cylinder: first 3 PGD frequency modes.

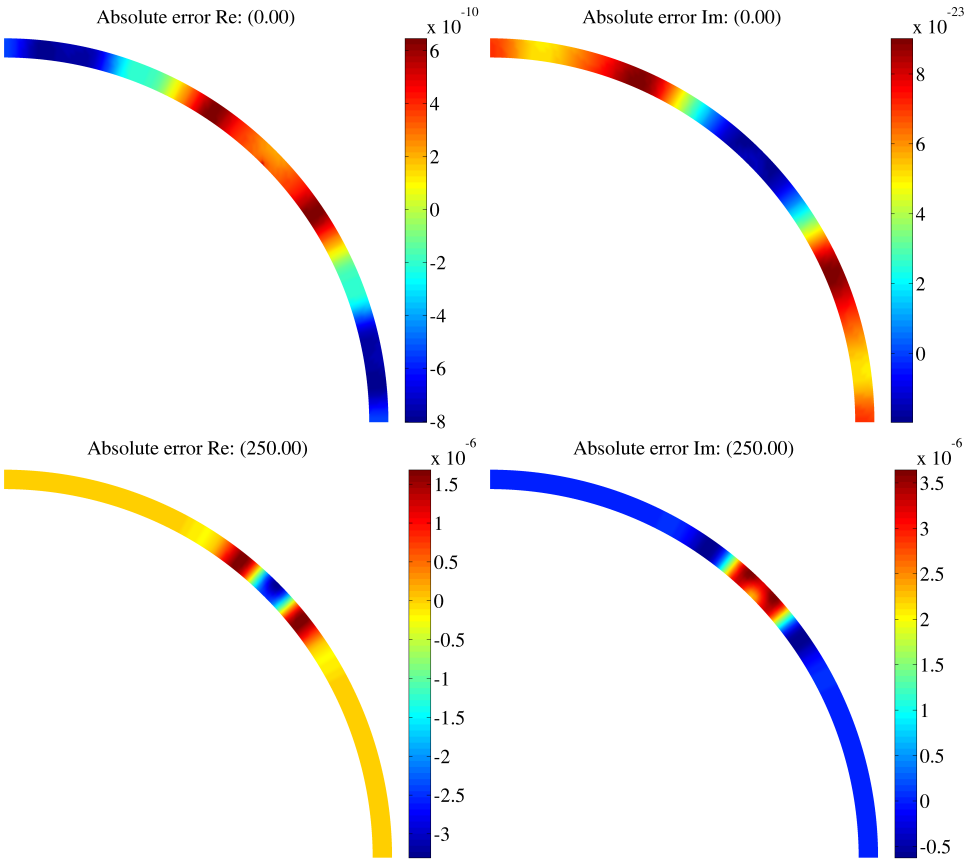


Figure 3.5: Single-ply composite cylinder: difference between PGD and FE solutions, real (left) and imaginary (right) parts, for frequencies 0 Hz (top) and 250 Hz (bottom).

their corresponding algebraic equation can be solved point-wise. However, a FE discretization is introduced to approximate the frequency separated functions in a least-squares sense.

To further verify that the PGD approximation of the generalized transfer function $\hat{h}^n$ is reasonable, it is evaluated at the extreme frequencies 0 and 250 Hz and then compared with a direct FE resolution of Eq. (3.15) for those precise frequencies. Fig. 3.5 depicts the difference between both approximations. The generalized solution gives approximations very close to those obtained with an FE computation, errors are always below 10^{-5}.

Finally, once $\hat{h}^n$ is determined and its inverse Fourier transform computed, $h^n = \mathcal{F}^{-1}[\hat{h}^n]$, Eq. (3.18) is used to determine the temperature at the desired monitoring point $\boldsymbol{x}_0$ for a given excitation. The evaluation of the PGD approximation and its inverse Fourier transform is performed only once for any excitation, it is the offline phase. The actual application of the convolution, see Eq. (3.18), for any excitation in order to determine the temperature at

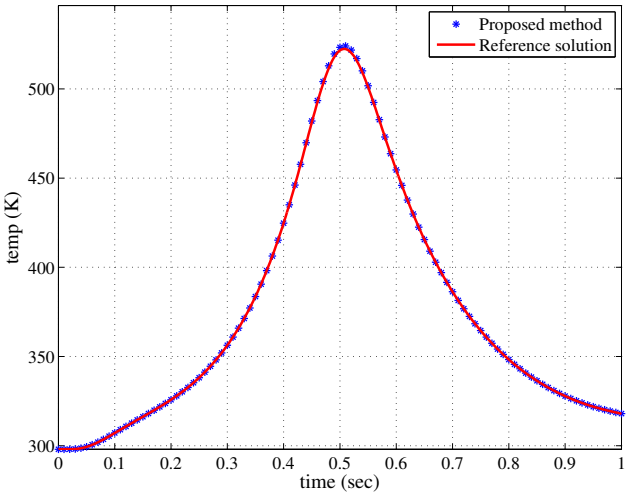


Figure 3.6: Single-ply composite cylinder: comparison of the temperature evolution at point $\boldsymbol{x}_0$ for a moving heat flux on the outer boundary for the proposed PGD-based solution (discontinuous blue) and the standard FE (solid red).

the monitored point is the online or post-process phase. Here this is done for the imposed external flux $q(\xi, t)$, see Eq. (3.29), in the time interval $t \in [0, 1]$ seconds, which is the window of interest. The convolution integral is discretized in time using the time-step that comes naturally from the greatest frequency considered in the generalized transfer function. This follows the previous discussion on the use of the Nyquist-Shannon theorem [122].

The evolution with time of the temperature at $\boldsymbol{x}_0$ is shown in Fig. 3.6. The reference solution (solid red line) is computed with FE and a Crank-Nicolson time-marching scheme. This scheme uses a ratio $\Delta t/\Delta x^2 = 20$ which has errors below $0.4\,10^{-4}$ compared with a reference solution using $\Delta t/\Delta x^2 = 1/2$ to ensure an accurate transient response. Note that each time-step requires the resolution of a system of equations whose dimension is determined by the FE mesh used.

It is clear from this figure that the proposed method produces an accurate response. However, it is more important to note that the online phase for PGD-based approximation of temperature, which is determined at 500 instants (equispaced by 2ms), requires with MATLAB® on a laptop 0.34s, which is almost a third of the physical time 1s. This confirms that given the generalized transfer function h^n and an imposed heat flux $q(\xi, t)$ along the outer boundary, the temperature at a point $\boldsymbol{x}_0$ can be evaluated in real-time (actually faster than real-time!). Note that this is 35 times faster than the full FE solution, which needs of around 12 seconds to be computed with a standard commercial code.

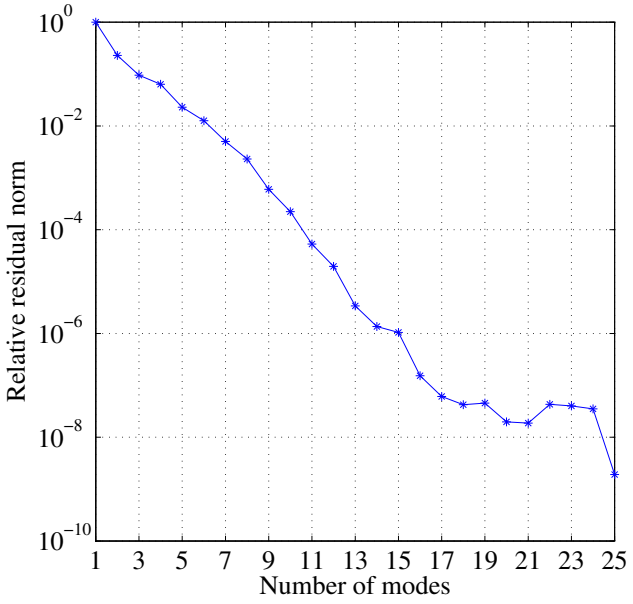


Figure 3.7: Multi-ply composite cylinder: convergence of the generalized frequency transfer function.

3.5.2 Multi-ply composite cylinder: response verification and imperfection influence

An imperfection is introduced in the previously studied problem. That is, the same geometry, see Fig. 3.1, and the same equations, see Eq. (3.28), with the same parameters are considered. However, now a zero thickness imperfection of a prescribed length is introduced in the middle of the ply just between the inner boundary and the outer boundary and centered at the measuring point. This imperfection models a possible delamination of the composite and it is modeled as a perfect adiabatic boundary. Obviously this imperfection affects convergence but not drastically, see Fig. 3.7 for a delamination length of 0.2m. Its influence is more clear when plotting modes of the generalized transfer function. In fact, the same comparison shown earlier (that is, the difference between a PGD approximation of the generalized transfer function $\hat{h}^n$ and the direct FE) is shown in Fig. 3.8 for the extreme frequencies 0 and 250 Hz. Recall that computations are still done for the whole range $[-250, 250]$ Hz to further verify the symmetric nature of the solution. Again errors are always below 10^{-5}.

As discussed at the end of Section 3.2.3, the generalized transfer function Eq. (3.8) is non-Hermitian but it is symmetric. The later property proves sufficient for reciprocity, but it is well known that there is no proof of monotonic convergence for the PGD method when

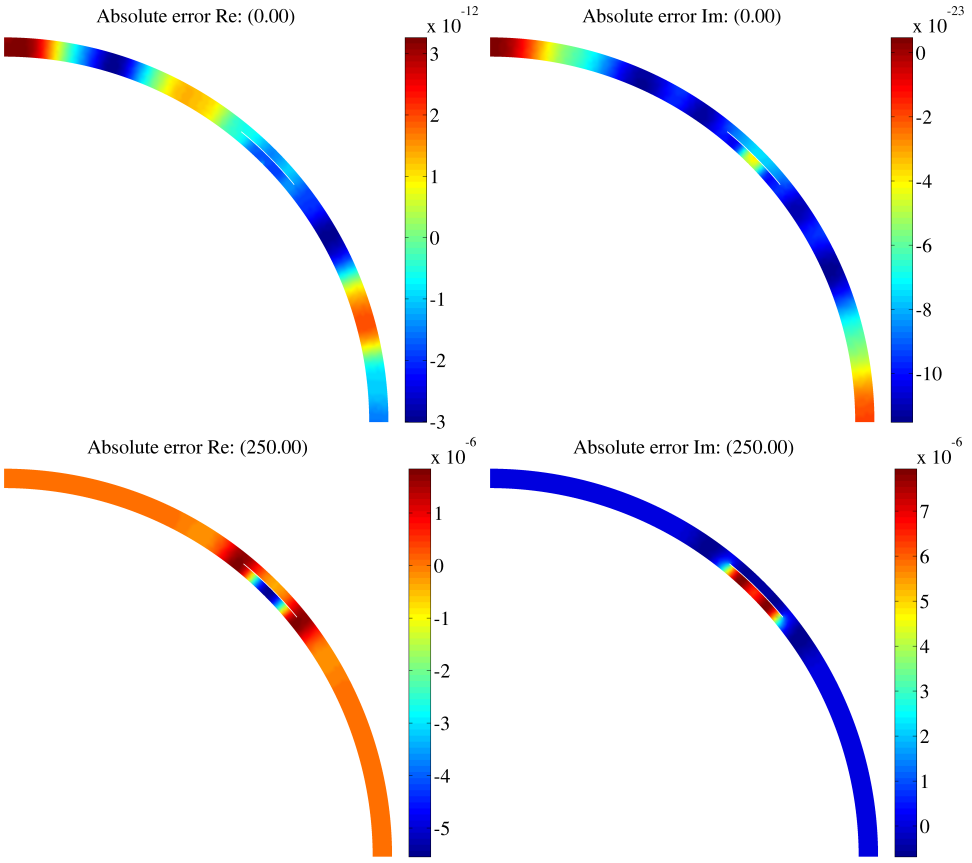


Figure 3.8: Multi-ply composite cylinder: difference between PGD and FE solutions, real (left) and imaginary (right) parts, for frequencies 0 Hz (top) and 250 Hz (bottom).

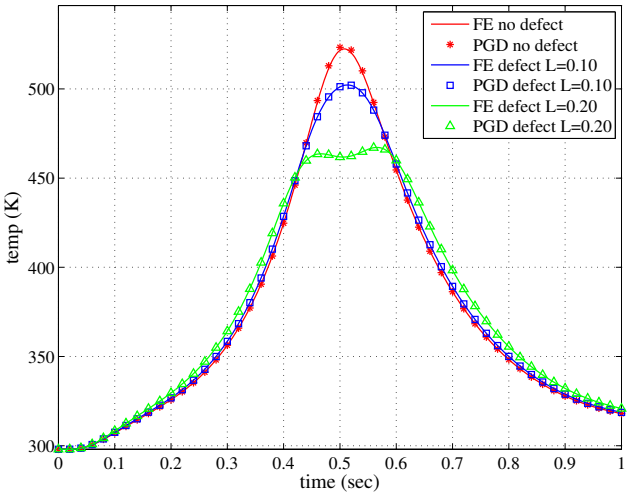


Figure 3.9: Multi-ply composite cylinder: comparison of the temperature evolution at point $\boldsymbol{x}_0$ for the proposed PGD-based solution (discontinuous) and the standard FE (solid) for different defect lengths.

confronted to non-Hermitian operators. Thus, the convergence shown in Fig. 3.7 shows a slight increase of the residual norm for the last computed terms.

The influence of the imperfection is even more clear when plotting temperature evolution at the measuring point. Fig. 3.9 shows three cases with their corresponding (*expensive*) comparison with FE, the no defect case, same curves shown in Fig. 3.6, and two defects, one of length 0.1m, the other with length 0.2m. Since the computer cost of the reference FE solution is similar to the one of the previous section, its overhead with respect to the prosed method is also of the same magnitude. The number of terms used in the PGD expansion is 21 and 25 for the short and long imperfection, respectively, recall that 19 were used with no imperfection. Thus, the imperfection does not increase the number of terms dramatically.

This methodology clearly shows that, in real-time, defects can be detected as the difference between measured and computed temperature values. Moreover, it further opens the possibility to determine the defect nature from its thermal signature.

3.5.3 Multi-parametric extension

As presented in Section 3.4.1 this PGD-based approach presented here has a potentiality that exceeds real-time monitoring of temperature at a given location. More precisely, thermal conductivity, k, can be chosen as an extra parameter. The generalized transfer function is now parametric in k and consequently, temperature at the monitoring point $\boldsymbol{x}_0$ for any instance t can be computed in real-time for any conductivity k, i.e. $u(\boldsymbol{x}_0, t, k)$, see Eq. (3.26).

The single-ply example presented and discussed in Section 3.5.1 is further generalized for any conductivity $k \in [1, 20]$W/(m K). In this section, the range of frequencies is up to 60 Hz instead of 250 Hz because there is no need to use the same time-step as the FE reference solution. In any case, the range of frequencies is taken such that the proper symmetries are recovered in the computed solution. This is clearly seen in the modes shown in Fig. 3.10 and Fig. 3.11.

This problem however is much more challenging because there is an extra parameter. But more important, these difficulties are relevant because variations in thermal conductivity introduce major changes in the real behavior of the thermal field. This is clearly observed in the modes associated to conductivity in Fig. 3.11. Note the large variations introduced close to the lower bound of the thermal range, recall $k \in [1, 20]$W/(m K). That is, for low conductivities solutions must localize close to the heat source. This has a clear influence in the convergence process of PGD, see Fig. 3.12. Although convergence to engineering precision ($0.5\,10^{-2}$) is obtained with 14 modes, the rate of convergence is slower compared to space-frequency separated representations computed in the previous cases. Moreover, if further precision is required (beyond engineering accuracy), the algorithm fails to converge due to the non-Hermitian character of the operator. See [15] for further details and strategies to overcome this issue. Moreover, regarding the fixed-point convergence of the multi-parametric problem (i.e. convergence of each greedy algorithm), the average number of iterations per mode is now increased to 59. Consequently, a total of 826 FE solves are done during the offline phase to compute the necessary 14 terms of the PGD expansion. Note, that a *brute force* approach sampling (no functional approximation) of the generalized transfer function at the 121 frequency and 77 conductivity nodes would imply 9317 FE solves. The cost of the *brute force* computation of the transfer functions depends on the fidelity of the discretization of the frequency and conductivity spaces. Whereas, the cost of the computation of the PGD expansion (and the number of terms in the expansion) is largely independent of the fidelity of the discretization of the frequency and conductivity spaces, assuming the discretization is sufficiently fine.

Nevertheless, the PGD-based scheme converges globally although its local behavior, precisely for low conductivity, shows less precision compared to larger values of k. This is better appreciated in the temperature evolution at the measuring point. Fig. 3.13 shows this evolution for several values of the conductivity with their corresponding (*expensive*) comparison with FE. Results are in very good agreement with the reference FE solution and precision increases as k increases.

Since the largest frequency considered for the evaluation of the generalized transfer function is 60Hz, the response at the measuring point $\boldsymbol{x}_0$ is recovered with a time-step of 8ms, recall Eq. (3.30). Consequently, the computational time needed for the online phase is faster than in previous examples, only 0.09 seconds. Whereas, the FE reference solution is computed with the same time-marching scheme described in previous sections ($\Delta t = 2$ms)

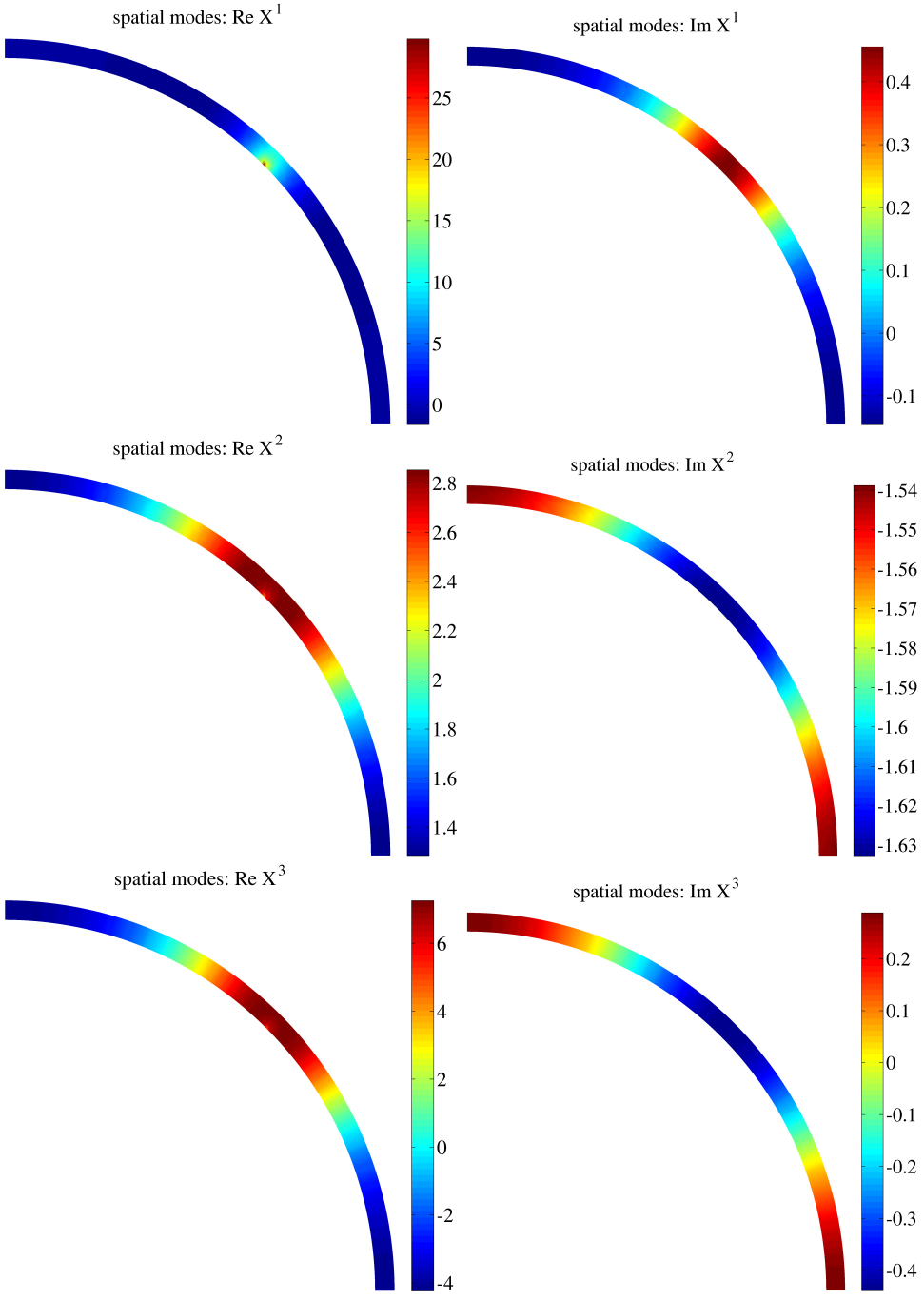


Figure 3.10: Single-ply multi-parametric composite cylinder: first 3 PGD spatial modes real (left) and imaginary (right).

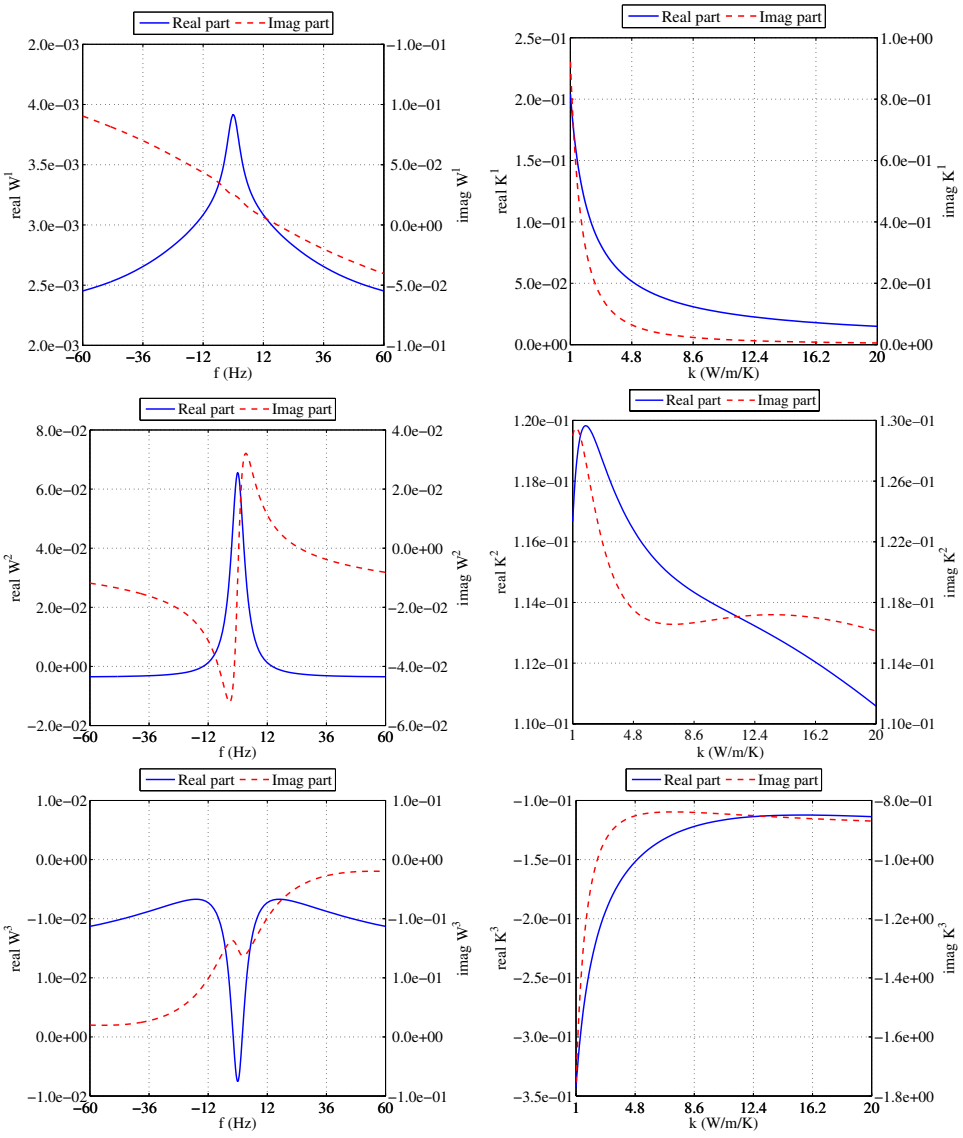


Figure 3.11: Single-ply multi-parametric composite cylinder: first 3 PGD frequency (left) and thermal conductivity (right) modes.

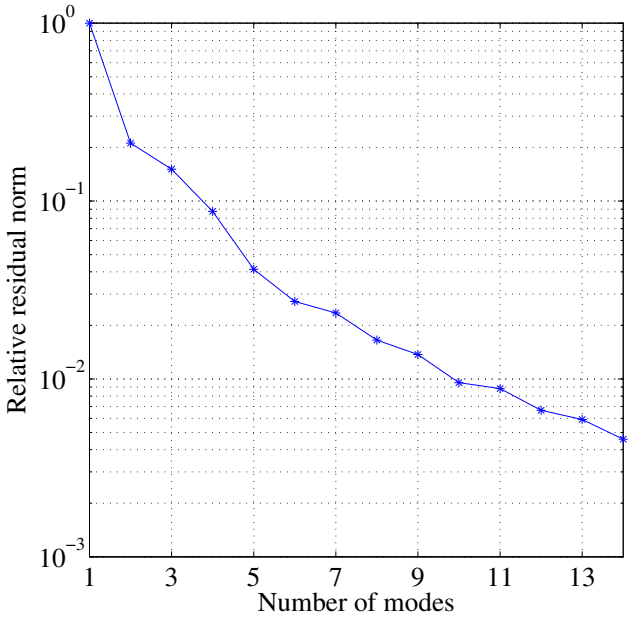


Figure 3.12: Multi-parametric convergence of the generalized frequency transfer function.

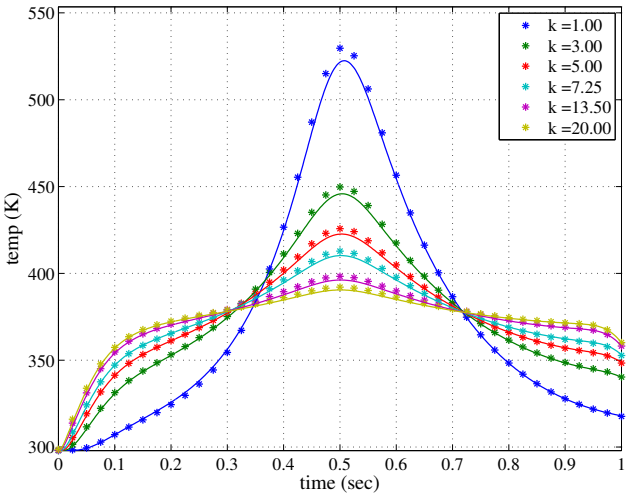


Figure 3.13: Multi-parametric comparison of the temperature evolution at point $\boldsymbol{x}_0$ for the proposed PGD-based solution (discontinuous) and the standard FE (solid) for thermal conductivities.

because this implies that $20 \leq k\Delta t/\Delta x^2 \leq 400$. Therefore, both solutions are not strictly comparable. The computational time associated to the FE solution remains unchanged (around 12 seconds) and the online stage of the proposed approach is now 133 times faster than the FE one.

This example clearly demonstrates the applicability of this approach to inverse problems where the measured temperature can be used to determine thermal conductivity.

3.5.4 Amplitude identification

This example shows the applicability of the proposed approach for the model case of identification discussed in Section 3.4.2. First, some measured temperature is needed, that is, $u^{\texttt{meas}}(\boldsymbol{x}_0, t_r)$ for $r = 1, \ldots, m$. In this case, the "measured" temperature is synthetically generated at $m = 501$ instances ($\Delta t = 2$ms) with an inflow forcing excitation equal to the one defined in Eq. (3.29) whose amplitude is modulated by $[1 + \cos(2\pi t)]/2$. That is, a FE code with a Crank-Nicolson time-marching scheme is used to generate the temperature data at the monitoring point under an external heat source

$$q(\xi, t) = 250[1 + \cos(2\pi t)] \exp\big(-50(2\xi - \pi t)^2\big) \mathrm{W/m}^2.$$

Second, this data is used to determine the laser input amplitude following the procedure described in Section 3.4.2. The amplitude of the excitation is assumed not known and it is approximated following a piecewise linear approximation with $n_{\text{fit}} = 50$ (i.e. a uniform mesh of 51 nodes each 20ms), see Eq. (3.27). Fig. 3.14 shows the synthetically generated temperature at the monitoring point (left) and a comparison (right) between the approximated nodal values (blue markers) and the reference amplitude (solid red line). The coincidence is remarkable.

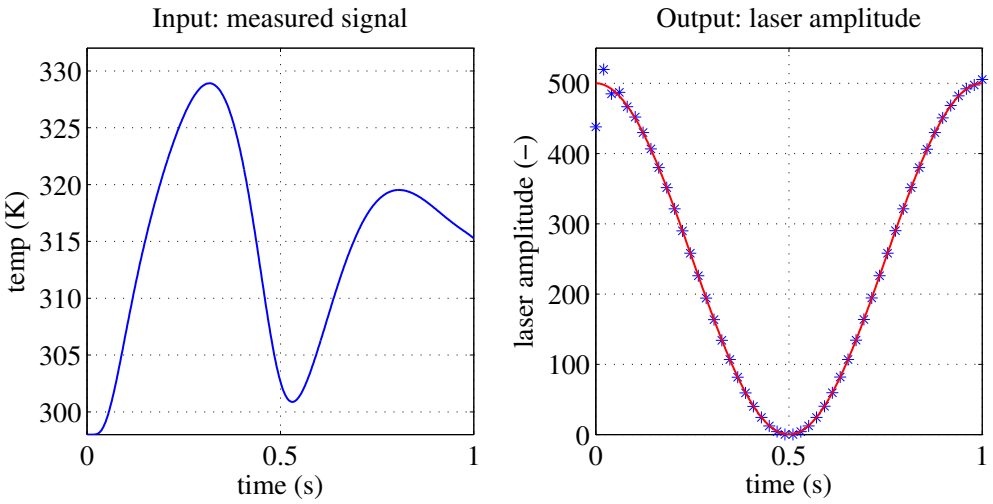


Figure 3.14: Amplitude identification: synthetically generated temperature measurements (left) used to calibrate the amplitude of the heat source (right), calibrated values (blue markers) and reference solution (solid red line).

Chapter 4

An *a priori* Empirical Interpolation Method

A primary requirement of the Proper Generalized Decomposition (PGD) algorithm to be efficient consists in being able to write the problem data in the canonical tensor representation. When this representation is not trivial, there exist several approaches to compute it. Among them, we can distinguish between projection, series expansion and interpolation-based methods.

In this Chapter, we explore these alternatives and specially their application to nonlinear problems solved in the PGD framework. A new interpolation-based method is introduced: the *A priori* Empirical Interpolation Method (AEIM). We shall pay special attention to nonlinear problems as they always introduce The AEIM is inspired from the Empirical Interpolation Method (EIM)[18, 35] in what concerns the election of the interpolation points, but unlike it, the interpolation basis is not known beforehand. The method presented here is *a priori* in such sense: it is able to choose both —and simultaneously— the interpolation basis and the interpolation points so as to minimize the distance between the multivariate function and its separated representation.

Contents

4.1 Nonlinear problems in the PGD framework

Many models encountered in physics and engineering are nonlinear and, ideally, one would like to be able to solve multidimensional models reducing its computational complexity and thus speeding-up simulations in the same manner as achieved for linear models. Unfortunately this is not possible today mainly due to the separability issues introduced by nonlinear equations.

As it has been shown in Chapter 1, the Proper Generalized Decomposition (PGD) is a multilinear solver —differential or algebraic, depending on the interpretation—. Therefore, when nonlinear problems are to be solved, the classic approach consists in coupling PGD with a nonlinear method: fixed-point, Newton, Large Time Increment method (LATIN) [83, 85] or Asymptotic Numerical Method (ANM) [42, 43], just to cite some of the possibilities that have already been tested in former works.

Nonlinear problems are in general difficult and they cannot be treated in an uniform manner: each nonlinear problem requires its own appropriate numerical techniques. In spite of that, nonlinear problems addressed in the PGD framework share some primary difficulties concerning both the separability of both coefficients and operators; see §1.5.3 for an introductory discussion. This Chapter is concerned with these *primary* difficulties. They appear disregarding the nonlinear method chosen when at some point of the implementation the nonlinear term needs to be evaluated at the previous iterate of the solution (thus known). The question is therefore how to evaluate the nonlinear term in separated form. Observe that even if the previous iterate is known in separated form, the nonlinear term cannot be written in separated format trivially.

Therefore, it is important to emphasize that we are not discussing the convenience of choosing a nonlinear method or another but the different methods to evaluate the nonlinear term in separated form.

4.1.1 Illustrating the separated representation of the operator

We shall first consider a linear equation that will be then turned into a nonlinear one to illustrate the difficulties that arise when one tries to set the problem in the usual PGD separated tensor structure. Recall that PGD takes advantage of this structure to split the high-dimensional problem into a series of low-dimensional problems. Low-dimensional problems are only coupled through coefficients computed from inner products in low dimensions. When nonlinear problems are considered, the inner product separation property —see §1.1.3.3— is of no help and computing the coupling coefficients requires performing integrals in high-dimensional domains, which destroys the computational performance of PGD.

4.1.1.1 Linear diffusion problem

Let $I := I_1 \times \cdots \times I_D$ be a multidimensional domain and $\boldsymbol{x} := (x_1, \ldots, x_D) \in I$, a point in it. Consider the variational form of the Laplace equation: find $u \in \mathcal{V}(I)$, an appropriate function space, such that

$$\langle k\nabla u, \nabla w\rangle = 0 \qquad \forall w \in \mathcal{V}(I),$$

with appropriate boundary conditions and k being a linear, homogeneous and isotropic diffusion coefficient. Observe that:

$$\langle k\nabla u, \nabla w\rangle = \sum_{d=1}^{D} \langle k\partial_{x_d} u, \partial_{x_d} w\rangle, \tag{4.1}$$

where ∂_{x_d} stands for $\partial/\partial x_d$. Proceeding very much as in §1.1.3.2, we introduce finite dimensional approximation spaces $V_d := \text{span}\{v_d^{1\leq i\leq N_d}\}$, for $1 \leq d \leq D$, from which a tensor product space $V := \otimes_{d=1}^{D} V_d$ can be built. We denote by N_d the dimension of V_d. Using the Galerkin method, both u and w belong to V. Taking the d-th term of Eq. (4.1) to simplify the notation, it can be written as:

$$\langle k\partial_{x_d} u, \partial_{x_d} w\rangle = \sum_{i_1,j_1=1}^{N_1} \cdots \sum_{i_D,j_D=1}^{N_D} \langle k\partial_{x_d} \bigotimes_{d=1}^{D} v_{i_d}, \partial_{x_d} \bigotimes_{d=1}^{D} v_{j_d}\rangle \, u_{i_1,\ldots i_D} w_{j_1,\ldots j_D},$$

where N_d is the dimension of the approximation space in dimension d. Given i_d, j_d for $1 \leq d \leq D$, the inner product in the last expression can be rewritten as follows:

$$\langle k\partial_{x_d} \bigotimes_{d=1}^{D} v_{i_d}, \partial_{x_d} \bigotimes_{d=1}^{D} v_{j_d}\rangle = \langle kv_{i_1}, v_{j_1}\rangle_1 \cdots \langle \partial_{x_d} v_{i_d}, \partial_{x_d} v_{j_d}\rangle_d \cdots \langle v_{i_D}, v_{j_D}\rangle_D, \tag{4.2}$$

where we have applied the separation property of the inner product, see Eq. (1.18). It is worth to recall that $\langle \cdot, \cdot \rangle_d$ denotes a scalar product in V_d. Eq. (4.2) clearly defines mass matrices for each dimension except the d-th one, where a diffusion matrix is found. The diffusivity parameter has been assigned to the first scalar product by convention. In algebraic form, the problem to be solved is: find $\boldsymbol{u} \in \mathbb{R}^{|\mathcal{N}|}$ such that

$$\boldsymbol{w}^T \boldsymbol{A}\boldsymbol{u} = \boldsymbol{w}^T \boldsymbol{b} \qquad \forall \boldsymbol{w} \in \mathbb{R}^{|\mathcal{N}|},$$

with

$$\boldsymbol{A} = \sum_{t=1}^{D} \otimes_{d=1}^{D} \boldsymbol{A}_d^t, \quad \text{and} \quad \boldsymbol{A}_d^t = \begin{cases} \boldsymbol{M}_d & \text{if} \quad d \neq t \\ \boldsymbol{K}_d & \text{if} \quad d = t \end{cases}, \tag{4.3}$$

We have denoted $\mathcal{N} := N_1 \times \cdots \times N_D$, the total dimension of the tensor product space V, and $|\cdot|$ the cardinal of the set. Besides, $\boldsymbol{M}_d$ and $\boldsymbol{K}_d$ are the mass and the diffusion matrices in dimension d, respectively. Notice that the Laplacian operator with a linear, homogeneous and isotropic diffusion coefficient provides naturally a separated tensor structure as a rank-D operator.

4.1.1.2 Nonlinear diffusion problem

Let us consider now the same problem but with a nonlinear diffusivity parameter, denoted by $k[u]$. Although it is not the aim of this chapter to analyze the different linearization schemes that might be applied to solve such nonlinear problem —see §4.1.2 for a short review—, let us consider for the sake of simplicity and without loss of generality that after ℓ nonlinear iterations we have computed an approximation of the solution, denoted by $\boldsymbol{u}_\ell$. Assume that a rank-R separated representation of such approximation is known; it is denoted by $\boldsymbol{u}_\ell^{(R)}$. Even in such favorable case, observe that $\boldsymbol{k}_\ell := k[\boldsymbol{u}_\ell] \in \mathbb{R}^{|\mathcal{N}|}$ is extremely easy to evaluate but impossible to handle in practice because $|\mathcal{N}|$ is potentially too big. Or equivalently, $\boldsymbol{k}_\ell$ is a tensor whose only known representation is the "explicit" one. Of course, one would like to have a lighter representation such as the canonical or the hierarchical Tucker, for instance, as it has been shown in §1.3.3.

Besides, the lack of tensor structure of the diffusion coefficient implies that the operator $\boldsymbol{A} \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{N}|}$ cannot be written, in principle, in a tensor separated structure similar to that shown in Eq. (4.3). To make it possible, one must compute a tensor representation of $\boldsymbol{k}_\ell$, which in the PGD framework means computing a canonical representation:

$$\boldsymbol{k}_\ell \approx \boldsymbol{k}_\ell^{(M)} := \sum_{m=1}^{M} \boldsymbol{k}^m, \quad \text{with} \quad \boldsymbol{k}^m := \otimes_{d=1}^{D} \boldsymbol{k}_d^m, \tag{4.4}$$

that for small M is a much less expensive representation of $\boldsymbol{k}_\ell$. It will be shown in next lines that Eq. (4.4) yields a separated tensor representation of the operator. To ease the exposition, let us consider a *functional* representation of Eq. (4.4) instead of the tensorial one, since both are equivalent:

$$k_\ell \approx k_\ell^{(M)} := \sum_{m=1}^{M} k^m, \quad \text{with} \quad k^m := \prod_{d=1}^{D} k_d^m.$$

Then, the "nonlinear" counterpart of Eq. (4.1) can be written as follows:

$$\langle k\nabla u, \nabla w\rangle = \sum_{d=1}^{D}\sum_{m=1}^{M} \langle k^m \partial_{x_d} u, \partial_{x_d} w\rangle.$$

After introducing the approximation spaces as in Eq. (4.2), the inner products to be computed in order to obtain the entries of the operator can be written in the following separated form:

$$\langle k^m \partial_{x_d} \bigotimes_{d=1}^{D} v_{i_d}, \partial_{x_d} \bigotimes_{d=1}^{D} v_{j_d}\rangle = \langle k_1^m\, v_{i_1}, v_{j_1}\rangle_1 \cdots \langle k_d^m\, \partial_{x_d} v_{i_d}, \partial_{x_d} v_{j_d}\rangle_d \cdots \langle k_D^m\, v_{i_D}, v_{j_D}\rangle_D. \tag{4.5}$$

It is important to emphasize that in Eq. (4.5) it has been possible to make use of the separation property of the inner product thanks to the separated structure of each term of

the rank-M decomposition, k^m. This yields a rank-$(D \times T)$ operator defined as follows:

$$\boldsymbol{A} = \sum_{t=1}^{D} \sum_{m=1}^{M} \otimes_{d=1}^{D} \boldsymbol{A}_d^{t,m}, \quad \text{and} \quad \boldsymbol{A}_d^{t,m} = \begin{cases} \boldsymbol{M}_d^m & \text{if} \quad d \neq t \\ \boldsymbol{K}_d^m & \text{if} \quad d = t \end{cases},$$

where matrices $\boldsymbol{M}_d^m$ and $\boldsymbol{K}_d^m$ are obtained by computing[1] and assembling[2] the following inner products:

$$\langle k_d^m \, v_{i_d}, v_{j_d} \rangle_d \quad \rightsquigarrow \quad \boldsymbol{M}_d^m \quad \text{and} \quad \langle k_d^m \, \partial_{x_d} v_{i_d}, \partial_{x_d} v_{j_d} \rangle_d \quad \rightsquigarrow \quad \boldsymbol{K}_d^m.$$

Remark 4.1 (*Nonlinear operators*). In this section, we have illustrated the non-separability of operator due to the presence of a nonlinear diffusion coefficient. However, the same kind of issue would have been found if nonlinear differential operators were considered. In what follows, and for the sake of simplicity, only non-separable and nonlinear coefficients shall be considered.

4.1.2 Linearization schemes and their coupling with PGD

As mentioned in §4.1, nonlinear problems are classically treated in the PGD framework by coupling a linearization scheme and the PGD solver. To ease the exposition, let us set the nonlinear problem to be solved in the following abstract algebraic form: find $\boldsymbol{u} \in \mathbb{R}^{|\mathcal{N}|}$ such that

$$\langle \boldsymbol{A}(\boldsymbol{u}), \boldsymbol{w} \rangle = \langle \boldsymbol{b}, \boldsymbol{w} \rangle \qquad \forall \boldsymbol{w} \in \mathbb{R}^{|\mathcal{N}|}.$$

Here $\langle \cdot, \cdot \rangle$ is simply the Euclidean scalar product. Among the most classic linearization schemes, one can cite the following:

Fixed-point linearization. It can be used if the operator can be expressed as the sum of a linear and nonlinear part, that is $\boldsymbol{A}(\boldsymbol{u}) = \boldsymbol{A}_l \boldsymbol{u} + \boldsymbol{A}_n(\boldsymbol{u})$. In such case, if at iteration ℓ an approximation $\boldsymbol{u}_\ell$ has been computed, $\boldsymbol{u}_{\ell+1}$ can be computed by solving

$$\langle \boldsymbol{A}_l \boldsymbol{u}_{\ell+1}, \boldsymbol{w} \rangle = \langle \boldsymbol{b}, \boldsymbol{w} \rangle - \langle \boldsymbol{A}_n(\boldsymbol{u}_\ell), \boldsymbol{w} \rangle$$

with the PGD method, which means that $\boldsymbol{u}_{\ell+1}$ is built from rank-one corrections using an alternating directions algorithm. This approach implies that at each nonlinear iteration, a full PGD problem has to be solved. Besides, the evaluation of is potentially very expensive unless a separated tensor representation is found, as it has been discussed in §4.1.1.2.

[1] In practice, it is done by choosing an integration rule such as the Gauss quadrature. Let us remark that functions k_d^m can be readily evaluated at the quadrature points.

[2] Since the notation is quite cumbersome, it is worth to clarify that here assembling means to loop over every i_d, j_d.

Incremental fixed-point linearization. In this case, the $\ell+1$ iterate is computed by updating $\boldsymbol{u}_\ell$ as follows: $\boldsymbol{u}_{\ell+1} = \boldsymbol{u}_\ell + \boldsymbol{u}$, where $\boldsymbol{u}$ is the *incremental correction*. It is computed by solving

$$\langle \boldsymbol{A}_l \boldsymbol{u}, \boldsymbol{w} \rangle = \langle \boldsymbol{b}, \boldsymbol{w} \rangle - \langle \boldsymbol{A}_l \boldsymbol{u}_\ell, \boldsymbol{w} \rangle - \langle \boldsymbol{A}_n(\boldsymbol{u}_\ell), \boldsymbol{w} \rangle$$

Intermediate PGD projections —see §1.5.1 for the definition of PGD projection— may be convenient to control the rank of the solution. Otherwise, it is expected to grow rapidly due to the incremental formulation which does not restart the separated representation from one nonlinear iteration to another, as in the previous case, but reuses the last iterate. The rank of the correction may be set free, that is, the PGD algorithm computes as much modes as needed to solve the linearized equation. In other cases, the rank of the correction might be predefined. However, this last option is likely to diverge in many cases since $\boldsymbol{u}_{\ell+1}$ does not satisfy the linearized equation. Whatever is the rank of the correction, if a standard PGD algorithm is used to computed, it will be built from successive rank-one corrections.

Newton methods. As in the previous case, the $\ell+1$ iterate of the solution is sought as $\boldsymbol{u}_{\ell+1} = \boldsymbol{u}_\ell + \boldsymbol{u}$, and the correction is computed from:

$$\langle \boldsymbol{J}_\ell \, \boldsymbol{u}, \boldsymbol{w} \rangle = \langle \boldsymbol{b}, \boldsymbol{w} \rangle - \langle \boldsymbol{A}(\boldsymbol{u}_\ell), \boldsymbol{w} \rangle, \tag{4.6}$$

where $\boldsymbol{J}_\ell$ denotes the jacobian matrix at the ℓ iterate. Of course, quasi-Newton methods in which the jacobian is not updated every nonlinear iteration are possible. Newton methods are described in many textbooks such as in [137] for finite element applications. In the PGD context, Newton methods pose several computational difficulties. The first concerns the evaluation of the right-hand side of Eq. (4.6), which in fact represents the weak form of the residual at the ℓ-th iteration. As discussed before, its evaluation is potentially very expensive unless a separated tensor representation is found. Furthermore, since the Newton method requires not only evaluating the residual but also computing the jacobian, the same kind of issue is encountered again when one attempts to compute numerically the jacobian.

Asymptotic Numerical Method (ANM). Let us assume again that the operator can be expressed as the sum of a linear and nonlinear part. Then we introduce a *loading parameter* λ such that

$$\langle \boldsymbol{A}_l \boldsymbol{u}, \boldsymbol{w} \rangle = \langle \boldsymbol{b}, \boldsymbol{w} \rangle - \langle \lambda \boldsymbol{A}_n(\boldsymbol{u}), \boldsymbol{w} \rangle. \tag{4.7}$$

Observe that for $\lambda = 0$ the linear problem whose solution is denoted by $\boldsymbol{u}_0$ is obtained whereas the solution of the nonlinear problem, the one we are really interested in, is obtained for $\lambda = 1$. The ANM defines an asymptotic expansion for both the unknown

and the loading parameter in terms of the expansion parameter, denoted here by α, as follows:

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{u}_0 + \alpha \boldsymbol{u}_1 + \alpha^2 \boldsymbol{u}_2 + \dots \\ \lambda &= \lambda_0 + \alpha \lambda_1 + \alpha^2 \lambda_2 + \dots \end{aligned} \tag{4.8}$$

If the nonlinear term has polynomial structure, then it can be expressed easily as an asymptotic expansion whose coefficients can be recursively identified. Consider the following example to illustrate the procedure:

Example 4.1 (*Illustrating ANM procedure*). Let us assume for the sake of simplicity that $\boldsymbol{A}_n(\boldsymbol{u}) = \boldsymbol{u}^2$. Then from Eq. (4.8) the nonlinear term can also be expressed as the following asymptotic expansion:

$$\boldsymbol{A}_n(\boldsymbol{u}) = \boldsymbol{q}_0 + \alpha \boldsymbol{q}_1 + \alpha^2 \boldsymbol{q}_2 + \dots \quad \text{with} \quad \boldsymbol{q}_p = 2\boldsymbol{u}_0 \boldsymbol{u}_p + \sum_{i=1}^{p-1} \boldsymbol{u}_i \, \boldsymbol{u}_{p-i}, \quad p = 1, 2, \dots .$$

This gives the following linear equation to be solved for each order $p \geq 1$:

$$\langle \boldsymbol{A}_l \boldsymbol{u}_p, \boldsymbol{w} \rangle + \langle 2\lambda \boldsymbol{u}_0 \boldsymbol{u}_p, \boldsymbol{w} \rangle = -\sum_{i=1}^{p} \langle \lambda \boldsymbol{u}_i \boldsymbol{u}_{p-i}), \boldsymbol{w} \rangle.$$

Once the solution, the loading parameter and the nonlinear term have been expressed in terms of the expansion parameter, they can be plugged into Eq. (4.7). Then, the different powers of α must be identified yielding a sequence of linear problems to compute the expansion coefficients $\boldsymbol{u}_p$, where p is the asymptotic expansion order. PGD is used to compute a separated tensor approximation of $\boldsymbol{u}_p$. Two advantages of this method are to be highlighted. First is that the operator remains unchanged during the iterations and does not need to be recomputed, unlike Newton method. This can be observed in the previous example: at any order $p \geq 1$, the operator to be assembled only has to take into account $2\boldsymbol{u}_0$ into account, which the solution of the linear problem. The second advantage is that the right-hand side, at each order, can be computed from the lower order coefficients.

Two disadvantages must also be mentioned. First is that the rank of the right-hand side is expected to grow rapidly with the order of the expansion. Observe again the expression of $\boldsymbol{q}_p$ in the previous example: it involves the multiplication of expansion coefficients which have separated structure because they have been computed using the PGD. Intermediate PGD projections could be used to control the rank growth. The second disadvantage concerns the stability of the method. Asymptotic are convergent inside its convergence disc and need to coupled with continuation strategies [89].

Finally, other strategies —although not covered here— could be also envisaged. Among them, we highlight LATIN strategies, considered in [83, 85], or Sparse Grids, considered in [31].

Remark 4.2 (*Solution's rank as a function of the linearization scheme and PGD coupling*). A very important issue when solving nonlinear problems in the PGD framework is how to couple the linearization scheme and the Greedy enrichment of the PGD solution. It has been observed that in many cases, the rank of the solution is subsidiary of the nonlinear iterative scheme. That is, the separated representation accumulates modes during intermediate stages of the nonlinear process which are not necessarily relevant in the final converged solution. In consequence, separated representations of rank unnecessarily high are obtained. PGD projection can be used to obtain a more compact separated representation of the final solution. However, a bigger computational cost has to be paid during the problem resolution. The investigation of the different coupling alternatives is an open issue not covered in this work.

4.2 Separability of multivariate functions

In what follows, we are simply supposing that we are given a multivariate function, which is admitted known and easy to evaluate, and we shall assume that a separated representation of such function wants to be computed. The multivariate function may represent:

- A coefficient which depends on the separated coordinates, but whose separated representation in terms of these coordinates is not known.
- A solution-dependent (i.e. nonlinear) coefficient in a partial differential equation.
- A nonlinear differential operator.

Let us emphasize that if given a multivariate function, we are able to:

- first, provide its separated representation,
- and second, compute such separated representation at a cost roughly independent on its dimension,

then it is possible to write a separated formulation of the problem adapted to the computational requirements of the PGD. Therefore, given a multivariate function $f : I \subset \mathbb{R}^D \to \mathbb{K}$, either real or complex-valued, the objective is to find a rank-M approximation:

$$f \approx f^{(M)} := \sum_{m=1}^{M} \psi_1^m \cdots \psi_D^m. \tag{4.9}$$

It is important to notice that we are not truly interested in finding the best possible separated representation, which is a very hard problem out of the scope of this thesis and ill-posed in general for the canonical formats; our main concern is to be able to compute such approximation with low computational expenses.

A tensor formulation of the problem is of course possible and equivalent to the *functional* version given before. One may simply introduce a discrete version of the multivariate function which constitutes a tensor in the sense of multidimensional array. Therefore, let us consider a regular cloud of points $\mathcal{M}_{\mathcal{N}}$, with $\mathcal{N} = (N_1, \dots, N_D)$ nodes per direction. Then, we denote by $\boldsymbol{f} \in \mathbb{K}^{|\mathcal{N}|}$ the explicit representation of the tensor version of the multivariate function f. We seek a canonical tensor representation such that:

$$\boldsymbol{f} \approx \boldsymbol{f}^{(M)} := \sum_{m=1}^{M} \boldsymbol{\psi}_1^m \otimes \cdots \otimes \boldsymbol{\psi}_D^m. \tag{4.10}$$

In what follows, we shall make no particular distinction in the use of either Eq. (4.9) or Eq. (4.10). The functional or the tensor version will be chosen depending on the aspects that want to be highlighted.

4.3 State of the art on separated representations of multivariate functions

Several strategies are possible to compute a separated representation such as the one shown in Eq. (4.9). They can be classified in three main categories: projection-based methods and interpolation-based methods.

4.3.1 Projection-based methods

Projection-based methods are formulated so as to minimize the error between $\boldsymbol{f}$ and its low-rank representation $\boldsymbol{f}^*$, defined by:

$$e = \|\boldsymbol{f}^* - \boldsymbol{f}\|^2, \tag{4.11}$$

where $\|\cdot\|$ denotes the Euclidean norm. Other particular norms are analyzed below. A first class of algorithms has been studied in §1.3.4, where explicit algorithms to compute low-rank tensor approximations where presented. In particular, the High-Order Singular Value Decomposition (HOSVD) and the Hierarchical Singular Value Decomposition (HSVD) were analyzed. Both are based on performing matricizations —that is, reshaping the tensor— and performing successively the SVD of such matricizations in order to find the dominant singular vectors. Their principal limitation is that explicit manipulation on the full tensor have to be performed, which in higher dimensions is not possible.

These algorithms provide Tucker and Hierarchical tensor representations, respectively. Both representations provide a separated tensor representation of the original tensor. Although this representation does not coincide with the canonical one, it is possible to take advantage of it in the PGD framework to formulate the problem with the desired separated structure.

The PGD projection algorithm, which has already been discussed in §1.5.1, is an alternative. It simply solves Eq. (4.11) using the PGD algorithm, as explained in §1.4.2. The operator to be used in this case is the identity. It provides a canonical tensor structure, but unlike the previous methods, in the general case, it does not enjoy from any quasi-optimality condition. The PGD projection also requires the explicit manipulation of the full tensor. From a computational point of view, this means that the separation property of the inner product cannot be used and therefore, the coefficients that couple the low-dimensional problems can only be computed by computing inner products in dimension $D-1$.

Since solving Eq. (4.11) using an Euclidean norm is very expensive, other alternatives inspired from *gappy* methods are possible [51, 132]. They are based on the use of the so-called *gappy* norms to reconstruct corrupted data or incomplete fields. Let $\boldsymbol{f}$ be the *true* field and $\boldsymbol{f}^*$ the reconstructed one. The idea is to minimize the error defined by

$$e = \|\boldsymbol{f}^* - \boldsymbol{f}\|_g^2 := \|\boldsymbol{P}^T(\boldsymbol{f}^* - \boldsymbol{f})\|^2, \tag{4.12}$$

where $\boldsymbol{P}$ is a *extractor* matrix whose singleton columns at the known, i.e. non corrupted, entries. In such manner, the error is measured with respect to the reliable data. This method has been tested for instance in the context of *a posteriori*, POD-based model order reduction. In this approach, $\boldsymbol{f}^*$ is assumed to live in a POD reduced basis. The combination coefficients are found precisely solving the minimization problem defined in Eq. (4.12), using the gappy norm.

Hence the question is how to use this approach in the context of *a priori* tensor approximations. Of course, we are not interested in reconstructing data but in taking advantage of the gappy norm to reduce the computational complexity of computing the approximation. The idea is therefore to exploit the gappy norm in order to compute an approximation by evaluating only some points. This is an appealing alternative that, even if it is not covered in this work, it might be interesting to explore in future research. Two questions remain open:

- First is the election of the approximation basis. In the *a posteriori* context, a POD or RBM basis can be chosen, but in the PGD context the election of the approximation basis is not trivial. Of course, the approximation basis should be chosen with canonical tensor structure.

- Second is how to choose the *gappy points*, that is, how to perform the sampling. Is there any particularly efficient strategy?

4.3.2 Interpolation-based methods

Interpolation-based methods aim at computing an approximation of the multivariate function by performing evaluations only at the interpolation points. In such sense, they are in the

same spirit that gappy methods. Two ingredients are needed to perform an interpolation: the interpolation basis and the interpolation points. Let us assume that the interpolation basis is given by a set of functions $\{\psi^{1\leq m\leq M}\}$. The multivariate function is approximated in terms of such basis as:

$$f \approx f^* := \sum_{m=1}^{M} \alpha_m \, \psi^m.$$

Coefficients α_m are chosen such as to enforce that the approximation matches the function at the interpolation points, i.e. $f(\boldsymbol{x}_m) = f^*(\boldsymbol{x}_m)$ for $1 \leq m \leq M$. With that condition, a linear square system of size M has to be solved to compute the coefficients. Of course, the system has to be invertible, which requires that the set of functions that constitute the interpolation basis must be linearly independent and the interpolation points must be different.

4.3.2.1 The Empirical Interpolation Method

The Empirical Interpolation method (EIM) [18, 35] is probably the most popular interpolation based technique in the MOR community, specially in both the POD and Reduced Basis framework. Recall that these communities perform *a posteriori* model reduction. In general, they use Galerkin projection to obtain the reduced system. However, when nonlinear problems are faced, the nonlinear term has to be evaluated repeatedly during the resolution and projected each time, which implies that the computational complexity still depends on the dimension of the full model. EIM is used in this context to avoid projection-based techniques. The interpolation basis is obtained by computing several snapshots not only of the solution, but also of the nonlinear term (that is, the multivariate function). If these snapshots are sufficiently representative, the interpolation basis is expected to represent well the nonlinear term. Being answered the question of how to choose the interpolation basis, we shall now focus on the interpolation points. In the EIM context, they are chosen using the *Magic Points* algorithm.

4.3.2.2 Magic Points

Assume the interpolation basis is ordered in descend order, which is easy if it comes from a POD. The Magic Points [92] associated to that basis are defined progressively as follows. The first interpolation point is set where the first basis function attains its maximum, in absolute value; that is:

$$\boldsymbol{x}_1 = \arg\max_{\boldsymbol{x}\in I} |\psi^1|.$$

Assume that we have already computed the first k interpolation points. Recall that a total of M points have to be computed, being M the size of the interpolation basis. The basic idea to choose the $(k+1)$-th point, associated to ψ^{k+1}, is to try to describe this interpolator

with the k previous basis functions and then locate the point where the discrepancy is the biggest. Therefore, the following square system

$$\sum_{m=1}^{k} \alpha_m \, \psi^m(\boldsymbol{x}_j) = \psi^{k+1}(\boldsymbol{x}_j), \quad 1 \leq j \leq k \quad \Rightarrow \quad \begin{array}{l} \boldsymbol{\alpha} = \boldsymbol{\Psi}^{-1}\boldsymbol{\psi}, \\ \boldsymbol{\Psi} \in \mathbb{K}^{k \times k} \quad \text{and} \quad \boldsymbol{\alpha}, \boldsymbol{\psi} \in \mathbb{K}^k, \end{array}$$

allows computing coefficients α_m such as the basis function ψ^{k+1} is interpolated using the k previous basis functions and their corresponding interpolation points: $\{\psi^j, \boldsymbol{x}_j\}_{1 \leq j \leq k}$. Then, we define the residual of the approximation as:

$$r := \psi^{k+1} - \psi^*, \quad \text{with} \quad \psi^* := \sum_{j=1}^{k} \alpha_j \, \psi^j,$$

where ψ^* is defined from the coefficients previously computed and expresses an approximation of ψ^{k+1}. The interpolation point is chosen at the point that maximizes the absolute value of the residual:

$$\boldsymbol{x}_{k+1} = \arg\max_{\boldsymbol{x} \in I} |r|.$$

Once the set of interpolation points is computed from the interpolation basis, the nonlinear term —or in general, the multivariate function— can be approximated readily by solving a small system of equations:

$$\sum_{m=1}^{M} \alpha_m \, \psi^m(\boldsymbol{x}_j) = f(\boldsymbol{x}_j), \quad 1 \leq j \leq M \quad \Rightarrow \quad \begin{array}{l} \boldsymbol{\alpha} = \boldsymbol{\Psi}^{-1}\boldsymbol{f}, \\ \boldsymbol{\Psi} \in \mathbb{K}^{M \times M} \quad \text{and} \quad \boldsymbol{\alpha}, \boldsymbol{f} \in \mathbb{K}^M. \end{array}$$

4.3.2.3 Extension to the a priori and multidimensional framework

Interpolated-based algorithms are very attractive as they offer an effective way to approximate multivariate functions by paying a relatively low computational cost. However, the EIM's formulation is clearly adapted to the a posteriori model reduction framework and its extension to the a priori framework is not straightforward. The following questions arise:

- How do we choose the interpolation basis? In the EIM procedure described before it is obtained by computing snapshots of the function, which of couse cannot be done in the a priori setting. Besides, for problems defined in several dimensions, snapshots are an explicit tensor representation that might not be possible to handle.

- How do we provide tensor structure to the interpolation basis? Of course, in the PGD framework we need the basis to possess a separated tensor structure and therefore this is another requirement to impose to the interpolation basis.

- Assuming the interpolation basis is known and has separated tensor structure. How do we apply the Magic Points algorithm in multidimensional problems? It is not

straightforward to find the maximum absolute value of a multidimensional residual, even if it has separated tensor structure. A full inspection of the residual is, in general, prohibitive.

A first approach which answered the two first questions but not the third was considered in [40]. It consisted in using as interpolation basis the modes of the solution. A rather simple nonlinear problem was solved using this technique, but several issues were noticed. First was that the nonlinear term was not accurately captured. An second was that the number of modes was relatively high since we were forcing the algorithm to add modes just to have enough functions to capture the nonlinear term. This reveals that the choice of the interpolation basis is far from optimality.

4.4 A new algorithm: *a priori* Empirical Interpolation

In this section a new computational strategy to compute separated representations of multivariate functions is presented. It is inspired from the Empirical Interpolation Method proposed, and it can be seen as its extension the a priori multidimensional framework. The main features of the method are the following:

- It provides the interpolation basis with canonical tensor structure as well as the interpolation points.
- The interpolation basis is constructed progressively using a greedy algorithm.
- Both interpolation basis and interpolation points are computed simultaneously using an alternating directions algorithm which implements a cheap search strategy to find the point where the residual of the approximation is maximum.
- This algorithm only requires performing function evaluations and it reduced significantly the computational cost associated to projection-based algorithms.

4.4.1 Illustrating the algorithm with a two dimensional problem

In order to facilitate the exposition, let us consider a two dimensional problem. The objective is to approximate the multivariate function by interpolation using a separated interpolation basis:

$$f(x,y) \approx f^{(M)}(x,y) := \sum_{m=1}^{M} \alpha_m \, \psi^m(x,y) \quad \text{with} \quad \psi^m(x,y) = \psi_x^m(x) \, \psi_y^m(y),$$

where the coefficients α_m are computed solving the following linear system —assuming that the interpolation points (x_j, y_j) are already known—:

$$f(x_j,y_j) = \sum_{m=1}^{M} \alpha_m \, \psi_x^m(x_j) \, \psi_y^m(y_j), \quad 1 \le j \le M.$$

Let us define the residual of the approximation associated to a rank-M separated representation of the multivariate function as follows:

$$r_M(x,y) := f(x,y) - f^{(M)}(x,y). \tag{4.13}$$

The residual will be zero at the interpolation points, but we do not have much information about the error elsewhere since we cannot afford exploring the full space in the general multidimensional case. In practice, the true error depends on the *goodness* of the basis functions ψ_x^j and ψ_y^j. To illustrate this fact, observe that if the interpolation functions are taken as a Kronecker delta centered on the interpolation points,

$$\psi_x^j = \delta(x - x_j) \quad \text{and} \quad \psi_y^j = \delta(y - y_j), \quad 1 \leq j \leq M,$$

the residual will be satisfied at the interpolation points but the approximation will be completely wrong elsewhere. On the other extreme, we could compute well-suited functions ψ_x and ψ_y using the POD or solving an approximation problem with the PGD —that is, using projection-based techniques—, but these options are computationally unaffordable, as it has been shown in §4.3.1. In the middle of these two extreme options, there are plenty of possibilities. The question is therefore how to choose well-adapted basis functions avoiding the *curse of dimensionality* associated to projection-based techniques.

Suppose that $M = 0$, i.e. no term has been computed yet. For the time being, let us assume that the first interpolation point (x_1, y_1) is known. We want to find ψ_x^1 and ψ_y^1 such that the residual is minimized. In such sense, this constitutes a greedy approach. A good compromise between accuracy and computational cost is to take ψ_x^1 as the restriction of the multivariate function on $(\,\cdot\,, y_1)$. Analogously, ψ_y^1 is taken as the restriction of the multivariate function on $(x^1, \,\cdot\,)$. In other words, the first basis function is restriction of the multivariate function on the cross centered on (x_1, y_1):

$$\psi^1 := \psi_x^1 \, \psi_y^1 \quad \text{with} \quad \begin{cases} \psi_x^1 = r_0(x, y_1) = f(x, y_1), \\ \psi_y^1 = r_0(x_1, y) = f(x_1, y), \end{cases}$$

where r_0 is the residual defined according to Eq. (4.13). Observe that the basis has separated structure by construction. Up to this point, we have a manner to compute the basis functions but they depend on the election of the interpolation point. That is, a different interpolation point leads to different basis functions. Therefore here we propose to perform an alternating directions algorithm whose outputs are simultaneoulsy both the interpolation point and the basis function. It proceeds as follows:

1. We initialize the algorithm by choosing y_1 randomly. This allows computing ψ_x^1 as explained before. Then inspired from the Magic Points algorithm, we seek the point where the ψ_x^1 is maximized and we normalize:

$$x_1 = \arg\max_{x \in I_x} |\psi_x^1|, \quad \text{then} \quad \psi_x^1 = \frac{\psi_x^1}{\|\psi_x^1\|_\infty}.$$

Of course, $\|\psi_x^1\|_\infty = \psi_x^1(x_1)$. We use the same notation for both normalized and non-normalized versions of the functions in order to keep the exposition as clear as possible.

2. From x_1 just updated, we are able to compute ψ_y^1 as explained before. Again, following the Magic Points spirit, y_1 initially assumed random can be updated and then normalized as follows:

$$y_1 = \arg\max_{y \in I_y}(\psi_y^1), \quad \text{then} \quad \psi_y^1 = \frac{\psi_y^1}{\|\psi_y^1\|_\infty}.$$

Observe that:

$$\|\psi_y^1\|_\infty = \|\psi_x^1\|_\infty = \|\psi^1\|_\infty.$$

The algorithm continues until the interpolation point (and so, the basis function) does not change anymore. In our experiences, this algorithm converges very quickly, normally 3 to 5 iterations, although a dependence on the dimension of the problem has been observed, meaning that more alternating iterations are needed to converge in higher dimensional problems.

Once the alternating directions algorithm converges, the rank-one approximation is written as:

$$f \approx f^{(1)} = \alpha_1 \, \psi_x^1 \, \psi_y^1.$$

The coefficient α_1 is computed so as to make the approximation $f^{(1)}$ pass through $f(x_1, y_1)$. Of course, the rank-one approximation is not expected to be good enough and therefore a second term may be added. The alternating directions algorithm is again applied. The function $\psi^2 = \psi_x^2 \psi_y^2$ is built from the restriction of the residual r_1 on the cross centered on (x_2, y_2):

$$\begin{cases} \psi_x^2 = r_1(x, y_2) = f(x, y_2) - f^{(1)}(x, y_2), \\ \psi_y^2 = r_1(x_2, y) = f(x_2, y) - f^{(1)}(x_2, y). \end{cases}$$

where r_1 denotes the residual as defined in Eq. (4.13). The process continues exactly in the same way for the subsequent terms. In order to stop the greedy algorithm the following error indicator can be used:

$$e_M := \|f(x_{M+1}, y_{M+1}) - f^{(M)}(x_{M+1}, y_{M+1})\|_\infty. \tag{4.14}$$

This criterion requires convergence on the $(M+1)$-th interpolation point.

4.4.2 Numerical example: two-dimensional Rosenbrock function

In order to illustrate the performance of the algorithm, we choose the Rosenbrock function whose separated representation is assumed to be ignored. For the general multidimensional

case, it is defined as follows:

$$f(x_1, \dots, x_D) = \sum_{i=1}^{D-1} (1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2, \tag{4.15}$$

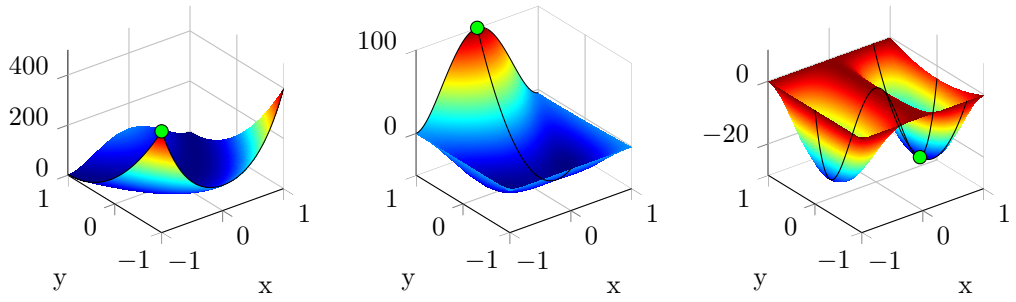
in a domain $I = [-1, 1]^D$. A uniform node distribution made of $N_d = 51$ nodes in every direction is used. The objective is to find a separated representation of the two dimensional version of this function starting with no a priori inspection of it. We initialize the alternating directions algorithm with a randomly chosen point. After two iterations, a first interpolation point is found:

$$(x_1, y_1) = (-1, -1).$$

Figure 4.1a shows the initial residual, which of course coincides with the function itself. The interpolation point found by the algorithm is depicted in green. It is to be highlighted that the algorithm succeeds in finding the maximum of the residual. In black, we highlight the cross that serves to compute the basis function. A new interpolation point is initialized randomly and after four iterations, a second interpolation point is found:

$$(x_2, y_2) = (0, 1).$$

Figure 4.1b shows the residual after the first mode has been computed, and we see again that the algorithm succeeds in placing the interpolation point at the maximum of the residual. Figure 4.1c provides exactly the same information but for the third residual —after a rank-two separated representations has been computed—.



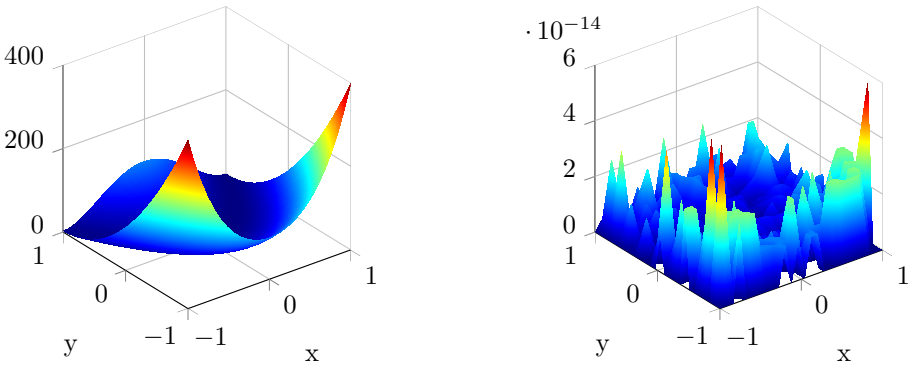
(a) $r_0(x, y)$, after 0 modes (b) $r_1(x, y)$, after 1 mode (c) $r_2(x, y)$, after 2 modes

Figure 4.1: Two-dimensional Rosenbrock function. Residuals, modes and interpolation points.

In this simple case, the algorithm performed optimally in the sense that a rank-three separated representation is found —see Eq. (4.15)—. However, after computing the third mode we do not know if the separation has converged or not, as the residual cannot be explicitly evaluated. Therefore, we compute a fourth interpolation point in order to be able

to evaluate Eq. (4.14). If it is below a fixed tolerance, the algorithm is stopped. In this example, $\varepsilon = 10^{-8}$. In all our tests, this procedure appears to be very robust. Therefore, in general, we always need to compute an extra term to estimate the error.

Figure 4.2a and Figure 4.2b show the exact Rosenbrock function and the absolute error of the computed approximation, respectively.

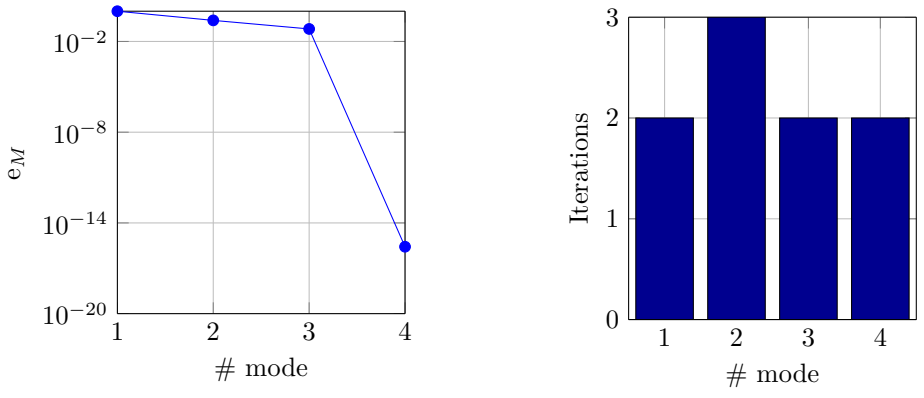


(a) 2D Rosenbrock function

(b) Approximation error

Figure 4.2: Two-dimensional Rosenbrock function. Error of the computed approximation.

Figure 4.3a shows the evolution of the stop criterion. We see that after computing the third mode the error drops to a negligible level, and therefore the function can be written in three terms. Figure 4.3b shows the number of iterations to convergence in the alternating directions loop.



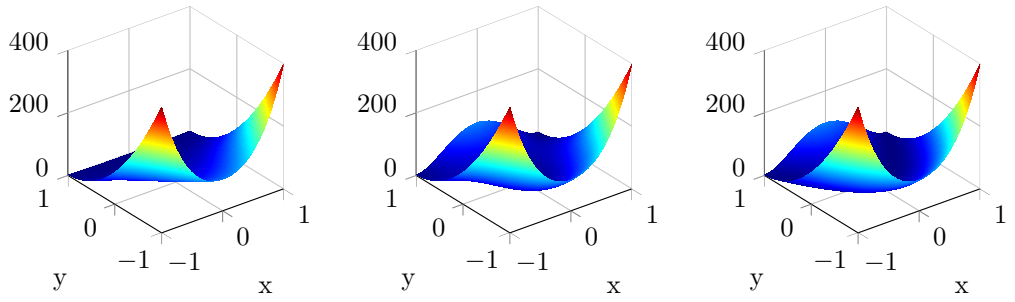
(a) Convergence

(b) Alternating iterations

Figure 4.3: Two-dimensional Rosenbrock function. Convergence analysis.

To illustrate the greedy algorithm used to compute the separated representation, we show

in Figure 4.4 how the separation is progressively enriched with, one, two and finally three modes.



(a) Rank-one $f^{(1)}$ (b) Rank-two $f^{(2)}$ (c) Rank-three $f^{(3)}$

Figure 4.4: Two-dimensional Rosenbrock function. Greedy enrichment of the solution.

Finally, Figure 4.5 shows the normalized modes. Observe that they are normalized with respect to their maximum absolute value.

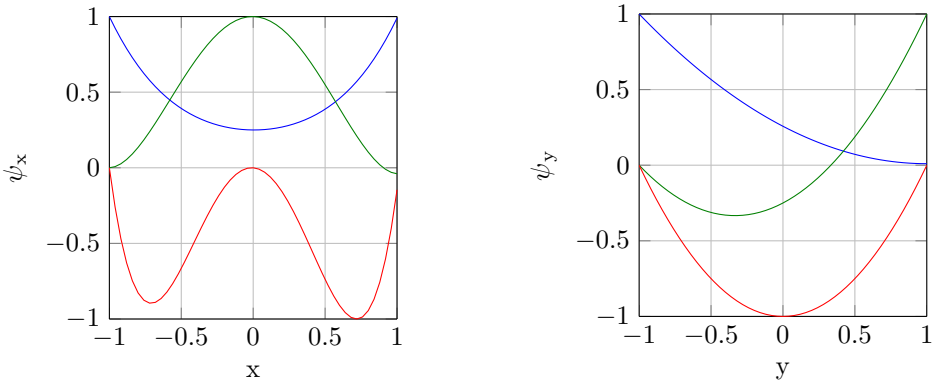


Figure 4.5: Two-dimensional Rosenbrock function. Computed modes ψ_x (left) and ψ_y (right).

4.4.3 Generalization to the multidimensional case

In the general multidimensional case the a priori Empirical Interpolation algorithm is formulated as follows. The objective is to compute a rank-M separated approximation:

$$f(\boldsymbol{x}) \approx f^{(M)}(\boldsymbol{x}) := \sum_{m=1}^{M} \alpha_m \, \psi^m(\boldsymbol{x}) \quad \text{with} \quad \psi^m(\boldsymbol{x}) = \prod_{d=1}^{D} \psi_d^m(x_d),$$

where the coefficients α_m are computed solving the following linear system —assuming that the interpolation points $\{\boldsymbol{x}_{1\leq m\leq M}\}$ are already known—:

$$f(\boldsymbol{x}_j) = \sum_{m=1}^{M} \alpha_m \, \psi^m(\boldsymbol{x}_j), \quad 1 \leq j \leq M. \tag{4.16}$$

The residual of the rank-M separated representation of the multivariate function is defined as follows:

$$r_M(\boldsymbol{x}) := f(\boldsymbol{x}) - f^{(M)}(\boldsymbol{x}). \tag{4.17}$$

With these ingredients, the Algorithm 5 generalizes the method already introduced in §4.4.1 to the multidimensional case.

Algorithm 5 A priori Empirical Interpolation

Require: Multivariate function $f(\boldsymbol{x})$

while $(m \leq M_{\max}$ & $e_M > \varepsilon)$ **do** ▷ Greedy enrichment loop
  Initialize int. point $\hat{\boldsymbol{x}}$
  for $i = 1$ to $I_{\max}$ **do** ▷ Alternating directions loop
    Set $\hat{\boldsymbol{x}}_{\text{old}} = \hat{\boldsymbol{x}}$
    for $d = 1$ to D **do** ▷ Loop for every dimension
      Update $\psi_d = r_M(\hat{x}_1, \ldots, \cdot, \ldots, \hat{x}_D)$ ▷ Separated basis function
      Update $\hat{x}_d = \arg\max_{x_d \in I_d} |\psi_d|$ ▷ Interpolation point
      Normalize ψ_d
    end for
    Check if $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}_{\text{old}}$ ▷ Stagnation criterion
  end for
  Set $\psi^m = \prod_{d=1}^{D} \psi_d$
  Compute coefficients α_m from Eq. (4.16)
  Compute error estimate e_M with Eq. (4.17)
end while
return $f^{(M)}$ ▷ Rank-M approximation

4.5 Numerical examples

In this section we provide separability results on the three and four-dimensional versions of the Rosenbrock functions, defined in Eq. (4.15). These examples are of academic interest as they allow illustrating the performance of the algorithm. More challenging results shall be considered in future research.

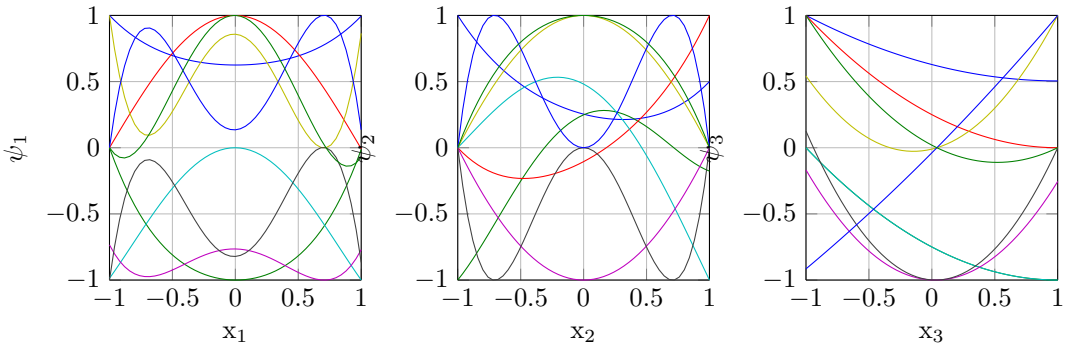


Figure 4.6: Three-dimensional Rosenbrock function. Computed modes ψ_1 (left), ψ_2 (center) and ψ_3 (right).

4.5.1 Three-dimensional Rosenbrock function

One of the principal observations when switching from two to three dimension is that, very much like PGD in higher dimensions, the algorithm here proposed is no longer optimal in the sense that it does not provide necessarily the most compact expansion. This is of course inherited from the alternating directions algorithm proposed to find the interpolation point. Although this is an unwanted behaviour of the algorithm, computing modes is very cheap and the computational time is not significantly affected. Besides, it is possible to combine this algorithm with a PGD projection —which is now very cheap because it is applied on a separated representation— to recompress the expansion. This is considered in §4.5.2

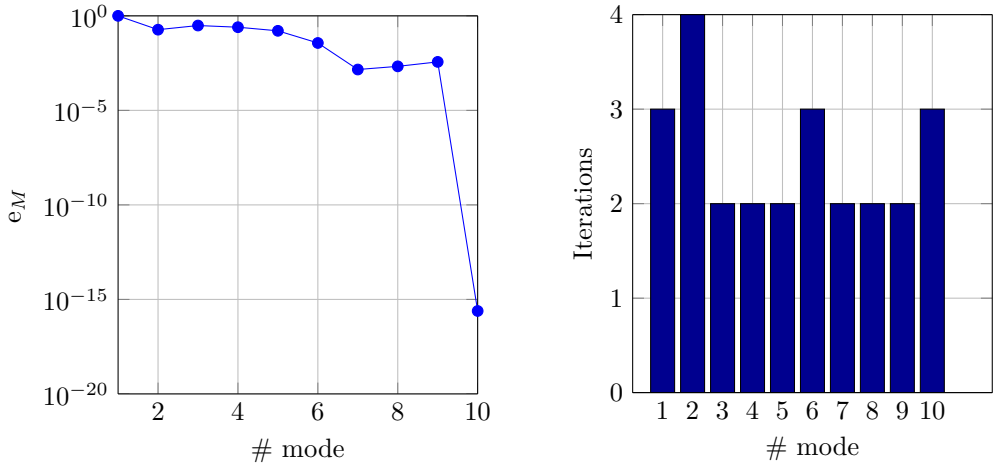
The separated representation of the three-dimensional Rosenbrock function converged with 9 modes (the tenth was negligible). Fig. 4.6 shows the computed modes in each direction.

Fig. 4.7a shows the convergence of the algorithm while Fig. 4.7b shows that the number of iterations inside the alternating directions algorithm remains very low.

We may define an *efficiency* ratio of the algorithm as follows:

$$\eta = 1 - \frac{N_{it} \times \sum_{d=1}^{D} N_d}{\prod_{d=1}^{D} N_d}, \tag{4.18}$$

being N_{it} the number of fixed-point iterations and N_d the number of nodes in the dimension d. This ratio measures the number of function evaluations made with the proposed algorithm versus a full evaluation. Of course, the closer to 1, the better performs the algorithm. In this example, a ratio $\eta = 0.9712$ is obtained which means that the function has been separated by performing less than 3% evaluations of the total data.



(a) Convergence

(b) Fixed-point iterations

Figure 4.7: Three-dimensional Rosenbrock function. Convergence analysis.

4.5.2 Four-dimensional Rosenbrock function

Finally, we consider the four-dimensional Rosenbrock function. Fig. 4.8 shows the computed modes in each direction.

Fig. 4.9a and Fig. 4.9b illustrate the convergence process and the number of iterations inside the alternating directions algorithm, respectively. The latter remains very low as in the previous examples. The convergence is still exponential in this problem.

As suggested in §4.5.1, a PGD projection step can be used in order to reduce the rank of the separated representation. Fig. 4.10 shows the effect of applying a final compression step for different error levels required to the algorithm. For instance, if the required error level is set to 10^{-6}, a total of 108 terms are computed while only 69 are kept after performing a PGD projection.

It is to be remarked the particular behaviour of compression, that produces more compact expansions for higher precision levels. This is explained by the need of the PGD algorithm of knowing a *true* approximation of the function to be able to find its separated representation. Of course, it could also be possible to apply intermediate PGD projection steps. For an error level of 10^{-8}, the efficiency parameter, defined in Eq. (4.18), is $\eta = 0.9856$, which means that the separated representation is computed by evaluating less than 2% of the total data. The computational time to compute 144 terms using a MATLAB® implementation is in the order of 10sec.

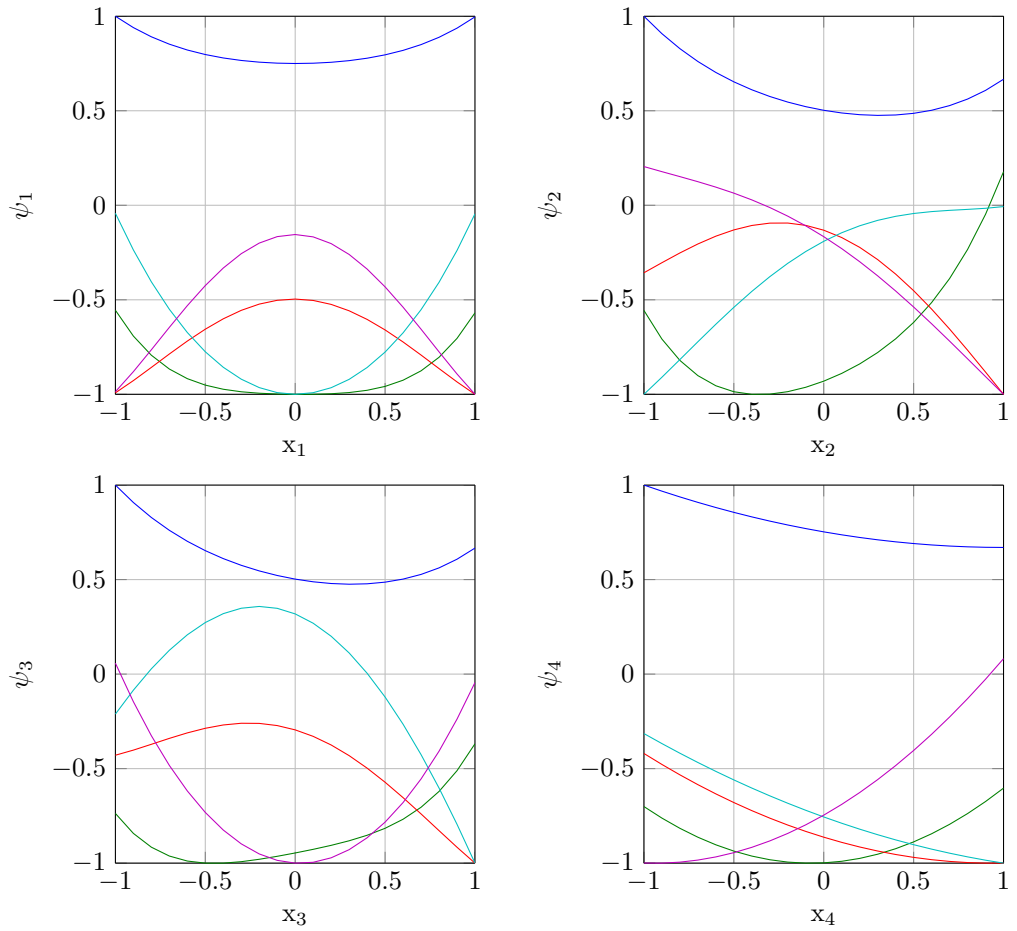
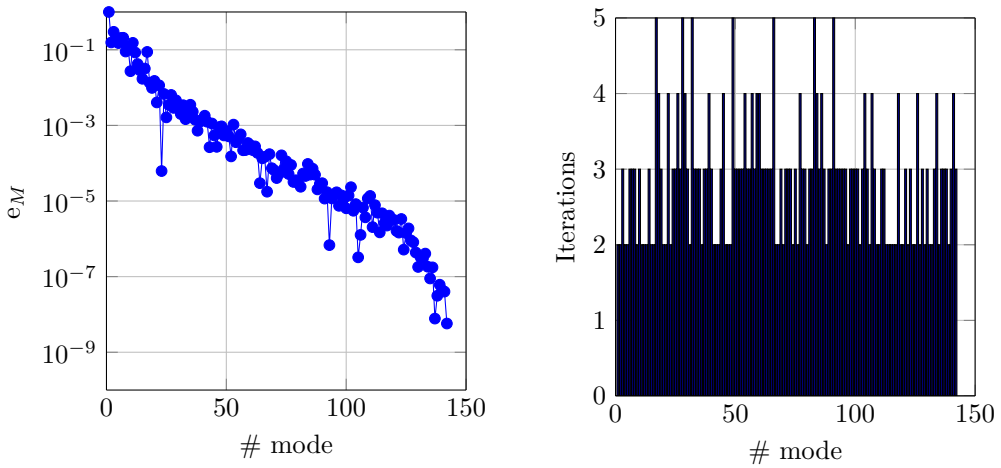


Figure 4.8: Four-dimensional Rosenbrock function. First five computed modes ψ_d, $d = 1, \ldots, 4$.



(a) Convergence

(b) Fixed-point iterations

Figure 4.9: Four-dimensional Rosenbrock function. Convergence analysis.

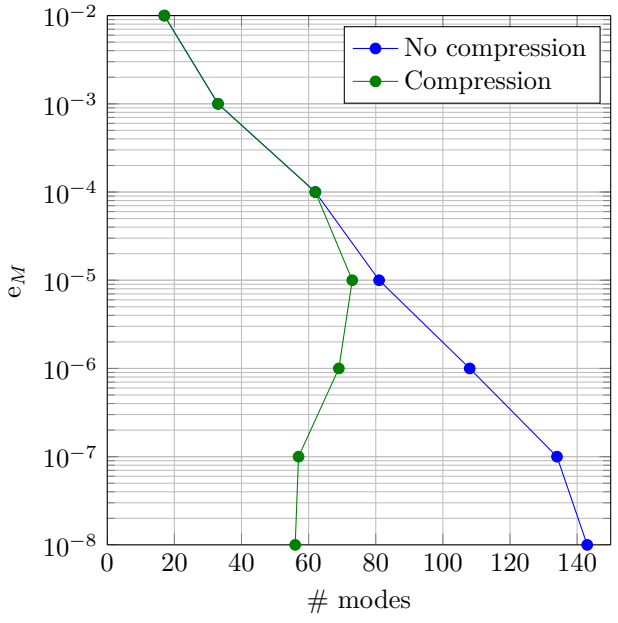


Figure 4.10: Four-dimensional Rosenbrock function. Effect of a final PGD projection.

Conclusion

In Chapter 1 we made en exhaustive review on the use of separated representations that comprised not only the Proper Generalized Decomposition (PGD) framework but other Model Order Reduction methods and Low-rank Tensor Approximation techniques. To the author's belief, the following conclusions hold:

- Both Model Order Reduction and Low-rank Tensor Approximations have reached in last years a significant maturity in their respective branches of science. Many fundamental advances have been achieved in last fifteen years: high-order and hierarchical singular value decompositions of tensors in 2000 [46] and 2010 [61], respectively, or the foundation of both Reduced Basis Methods in 2001 [111] and Proper Generalized Decomposition in 2007 [7], just to cite some examples. At the same time, a huge number of applications in many fields of science have demonstrated the interest of these techniques.

- Today, separated representations are probably the best alternative to push forward the current computability limits with regard to multidimensional problems. Disregarding how big the computational power will be in the future, it seems that there is no definitive solution to the curse of dimensionality. In consequence, we will need of either model reduction or low-rank methods as long as engineering needs of solving multidimensional models.

- In spite of the previous paragraphs, many challenges are of course still open. In the particular case of PGD, they can be summarized as follows. On one hand, there is a need of appropriate formulations and algorithms to ensure the separability and compactness of the solution of non-self adjoint and nonlinear problems. On the other hand, there is a need of formulating PGD problems —specially transient and nonlinear problems— in separated representation as a basic requirement for the computational performance of the PGD.

In Chapter 2, we introduced frequency-based formulations for the solution of parametrized structural dynamics equations. The following conclusions can be drawn:

- The need of solving parametrized eigenvalue problems, as required by Modal Analysis, or expensive direct time integrations, are completely avoided.
- An offline/online approach allowed recovering the space-time solution from a precomputed Generalized Transfer Function (GTF) and the applied excitation by performing a simple Fourier inverse transform. In addition, real-time applications can be readily envisaged thanks to the application of the convolution theorem.
- The GTF is valid not only for one particular forcing term but for any forcing term that can be written as a combination of the considered frequencies (linearity). This fact, along with the parametric nature of the approach, makes of the GTF a strong tool for the solution of parametrized structural dynamics equations.
- Two limitations of this approach are identified. The first concerns the transient response due to initial conditions, which is in principle lost using a frequency approach. A remedy could be to use the space modes of the GTF to implement a Reduced Basis approach. However, this approach should be further investigated to assess its performance. The second limitation concerns nonlinear problems, which are in general precluded. However, nonlinearities that only apply on the harmonic amplitude, can be efficiently assessed with this method, as shown by C. Germoso in [56].

The frequency approach presented in Chapter 2 is further developed in Chapter 3 taking advantage of some additional properties not exploited there. This following conclusions can be drawn:

- The frequency formulation allows overcoming the separability issues associated to moving excitations, which can be thought as waves and therefore they are intrinsically non-separable in space and time.
- The *reciprocity principle* is proven thanks to the symmetrization introduced by the frequency formulation. This property yields a direct application for the real-time monitoring of thermal processes.
- An offline/online approach allowed recovering the space-time solution of a thermal problem from a precomputed Generalized Transfer Function (GTF) and the applied excitation by performing a simple Fourier inverse transform. Real-time applications are demonstrated in several examples making use of the convolution theorem.
- The potentiality and the accuracy of the approach are demonstrated with applications in defect detection, multi-parametric evaluation and inverse calibration of processes.
- The same two limitations identified for the frequency-approach presented in Chapter 2, as well as the described remedies, apply in this case.

Chapter 4 provides a more general regard on the separated problem formulation. A general-purpose method to compute separated representations of multivariate functions is presented. The following conclusions and perspectives apply:

- The interest of interpolation-based techniques as a tool to encompass the computational complexity associated to projection-based techniques is demonstrated.
- An innovative extension of the Empirical Interpolation Method adapted to the a priori model reduction framework is proposed. Its interest and potentiality in the separated formulation of nonlinear problems in the PGD framework are justified.
- Promising results and convergence analysis are presented in moderate dimension problems, in which the technique seems both computationally efficient and robust.
- These performances should be confirmed by solving more challenging problems, and specially, nonlinear problems. This constitutes a work in progress.

Bibliography

[1] Aghighi, S., Ammar, A., Metivier, C., Normandin, M., and Chinesta, F. (2013). Non incremental transient solution of the Rayleigh-Bénard convection model using the PGD. *J. Non-Newtonian Fluid Mech.*, 200:65–78. *cited on page 39*

[2] Aguado, J., Abisset-Chavanne, E., Cueto, E., Chinesta, F., and Keunings, R. (2015a). Fractional modelling of functionalized CNT suspensions. *Rheol. Acta*, 54(2):109–119. *cited on page 56*

[3] Aguado, J. V., Huerta, A., Chinesta, F., and Cueto, E. (2015b). Real-time monitoring of thermal processes by reduced order modeling. *Int. J. Numer. Meth. Engng.*, 102(5):991–1017. *cited on page 86*

[4] Ammar, A., Chinesta, F., Díez, P., and Huerta, A. (2010a). An error estimator for separated representations of highly multidimensional models. *Comput. Meth. Appl. Mech. Engrg.*, 199(25-28):1872–1880. *cited on page 33, 61 and 95*

[5] Ammar, A., Cueto, E., and Chinesta, F. (2012). Reduction of the Chemical Master Equation for gene regulatory networks using Proper Generalized Decompositions. *Int. J. Numer. Methods Biomed. Eng.*, 28(9):960–973. *cited on page 17*

[6] Ammar, A., Huerta, A., Chinesta, F., Cueto, E., and Leygue, A. (2014). Parametric solutions involving geometry: a step towards efficient shape optimization. *Comput. Meth. Appl. Mech. Engrg.*, 268:178–193. *cited on page 40 and 97*

[7] Ammar, A., Mokdad, B., Chinesta, F., and Keunings, R. (2007). A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids. Part II: transient simulation using space-time separated representations. *J. Non-Newtonian Fluid Mech.*, 144(2-3):98–121. *cited on page 6, 12, 32, 39 and 139*

[8] Ammar, A., Normandin, M., and Chinesta, F. (2010b). Solving parametric complex fluids models in rheometric flows. *J. Non-Newtonian Fluid Mech.*, 165(23):1588–1601. *cited on page 39*

[9] Ammar, A., Normandin, M., Daim, F., Gonzalez, D., Cueto, E., and Chinesta, F. (2010c). Non-incremental strategies based on separated representations: applications in computational rheology. *Commun. Math. Sci.*, 8(3):671–695. *cited on page 39*

[10] Ammar, A., Prulière, E., Chinesta, F., and Laso, M. (2009). Reduced numerical modeling of flows involving liquid-crystalline polymers. *J. Non-Newtonian Fluid Mech.*, 160(2):140–156. *cited on page 8*

[11] Ammar, A., Zghal, A., Morel, F., and Chinesta, F. (2015). On the space-time separated representation of integral linear viscoelastic models. *Comput. Methods Appl. Mech. Eng.*, 343(4):247–263. *cited on page 40*

[12] Amsallem, D. and Farhat, C. (2011). An online method for interpolating linear parametric reduced-order models. *SIAM J. Sci. Comput.*, 33(5):2169–2198. *cited on page 8*

[13] Antoulas, A., Sorensen, D., and Gugercin, S. (2001). A survey of model reduction methods for large-scale systems. *Contemp. Math.*, 280:193–220. *cited on page 5*

[14] Arias, I. and Achenbach, J. (2004). Rayleigh wave correction for the BEM analysis of two-dimensional elastodynamic problems in a half-space. *Int. J. Numer. Meth. Engng.*, 60(13):2131–2146. *cited on page 88*

[15] Barbarulo, A. (2012). *On a PGD model order reduction technique for mid-frequency acoustic.* PhD thesis, Ecole Normale Supérieure de Cachan, France. *cited on page 37, 42 and 109*

[16] Barbarulo, A., Ladevèze, P., Riou, H., and Graveleau, M. (2012). A model order reduction technique to consider uncertainties over mid and high broad frequency bands. In Sas, P., Moens, D., and Jonckheere, S., editors, *Proceedings of International Conference on Noise and Vibration Engineering (ISMA2012) / International Conference on Uncertainty in Structural Dynamics (USD2012), Leuven, Belgium*, pages 1651–1663. *cited on page 42*

[17] Barbarulo, A., Ladevèze, P., Riou, H., and Kovalevsky, L. (2014). Proper Generalized Decomposition applied to linear acoustic: A new tool for broad band calculation. *J. Sound Vibr.*, 333(11):2422–2431. *cited on page 47*

[18] Barrault, M., Maday, Y., Nguyen, N., and Patera, A. (2004). An "empirical interpolation method": application to efficient reduced-basis discretization of partial differential equations. *C.R. Acad. Sci. I-Math.*, 339(9):667–672. *cited on page 115 and 126*

[19] Bathe, K.-J. (2006). *Finite Element Procedures.* Civil engineering series. Prentice-Hall, Englewood Cliffs, NJ. *cited on page 10 and 47*

[20] Bellman, R. E. (2003). *Dynamic Programming.* Courier Dover Publications, New York, republished edition. *cited on page 1 and 20*

[21] Bellomo, N. (2008). *Modeling complex living systems. A Kinetic Theory and Stochastic Game Approach.* Modeling and Simulation in Science, Engineering and Technology. Birkhauser, Basel, 1st edition. *cited on page 17*

[22] Benner, P., Gugercin, S., and Willcox, K. (2013). A survey of Model Reduction methods for parametric systems. In Preprints, M. P. I. M., editor, *Preprint MPIMD/13-14.* *cited on page 5*

[23] Białecki, R., Kassab, A., and Fic, A. (2005). Proper Orthogonal Decomposition and modal analysis for acceleration of transient FEM thermal analysis. *Int. J. Numer. Meth. Engng.*, 62(6):774–797. *cited on page 85*

[24] Bialecki, R., Kassab, A., and Fic, A. (2005). Proper Orthogonal Decomposition and Modal Analysis for acceleration of transient FEM thermal analysis. *Int. J. Numer. Meth. Engng.*, 62(6):774–797. *cited on page 8*

[25] Billaud-Friess, M., Nouy, A., and Zahm, O. (2014). A tensor approximation method based on ideal minimal residual formulations for the solution of high-dimensional problems. *arXiv:1304.6126.* *cited on page 31*

[26] Bird, R., Curtiss, C., Armstrong, R., and Hassager, O. (1987). *Dynamics of polymeric liquids: Kinetic Theory*, volume 2. John Wiley & Sons Ltd., Chichester, 2nd edition. *cited on page 17*

[27] Bognet, B., Leygue, A., and Chinesta, F. (2014). Separated representations of 3D elastic solutions in shell geometries. 1(1):1–34. *cited on page 40*

[28] Bognet, B., Leygue, A., Chinesta, F., Poitou, A., and Bordeu, F. (2011). Advanced simulation of models defined in plate geometries: 3D solutions with 2D computational complexity. *Comput. Meth. Appl. Mech. Engrg.*, 201:1–12. *cited on page 40*

[29] Boucinha, L., Ammar, A., Gravouil, A., and Nouy, A. (2014). Ideal minimal residual-based Proper Generalized Decomposition for non-symmetric multi-field models. Application to transient elastodynamics in space-time domain. *Comput. Meth. Appl. Mech. Engrg.*, 273:56–76. *cited on page 37*

[30] Bozorgnia, Y. and Bertero, V. V. (2004). *Earthquake Engineering: from Engineering Seismology to performance-based Engineering.* CRC Press, Taylor & Francis Group, Boca Raton, FL, 1st edition. *cited on page 81*

[31] Bungartz, H. and Griebel, M. (2004). Sparse grids. *Acta Numer.*, 13:147–269. *cited on page 6 and 122*

[32] Burkardt, J., Gunzburger, M., and Lee, H.-C. (2006). POD and CVT-based reduced-order modeling of Navier-Stokes flows. *Comput. Methods Appl. Mech. Engrg.*, 196(1-3):337–355. *cited on page 8*

[33] Carroll, J. and Chang, J. (1970). Analysis of individual differences in multidimensional scaling via an N-way generalization of Eckart-Young decomposition. *Psychometrika*, 35(3):283–319. *cited on page 23 and 24*

[34] Chancellor, C., Ammar, A., Chinesta, F., Magnin, M., and Roux, O. (2013). Linking discrete and stochastic models: the Chemical Master Equation as a bridge between process hitting and Proper Generalized Decomposition. *Lect. Notes Comput. Sc.*, 8130:50–63. *cited on page 40*

[35] Chaturantabut, S. and Sorensen, D. (2010). Nonlinear model order reduction via Discrete Empirical Interpolation. *SIAM J. Sci. Comput.*, 32(5):2737–2764. *cited on page 115 and 126*

[36] Chinesta, F., Ammar, A., and Cueto, E. (2010). Proper Generalized Decomposition of multiscale models. *Int. J. Numer. Meth. Engng.*, 83(8):1114–1132. *cited on page 39*

[37] Chinesta, F., Ammar, A., Lemarchand, F., Beauchene, P., and Boust, F. (2008). Alleviating mesh constraints: model reduction, parallel time integration and high resolution homogenization. *Comput. Meth. Appl. Mech. Engrg.*, 197(5):400–413. *cited on page 39*

[38] Chinesta, F., Ladevèze, P., and Cueto, E. (2011). A short review on model order reduction based on Proper Generalized Decomposition. *Arch. Comput. Methods Eng.*, 18(4):395–404. *cited on page 5 and 95*

[39] Chinesta, F., Leygue, A., Bognet, B., Ghnatios, C., Poulhaon, F., Bordeu, F., Barasinski, A., Poitou, A., Chatel, S., and Maison-Le-Poec, S. (2014). First steps towards an advanced simulation of composites manufacturing by automated tape placement. *Int. J. Mater. Form.*, 7(1):81–92. *cited on page 39 and 85*

[40] Chinesta, F., Leygue, A., Bordeu, F., Aguado, J., Cueto, E., Gonzalez, D., Alfaro, I., Ammar, A., and Huerta, A. (2013). PGD-based Computational Vademecum for efficient design, optimization and control. *Arch. Comput. Methods Eng.*, 20(1):31–59. *cited on page 5, 17, 20, 94, 95, 97 and 128*

[41] Clough, R. W. and Penzien, J. (1993). *Dynamics of Structures*. Civil engineering series. McGraw-Hill, New York, NY, 2nd edition. *cited on page 47 and 51*

[42] Cochelin, B., Damil, N., and Potier-Ferry, M. (1994a). The Asymptotic Numerical Method: an efficient perturbation technique for nonlinear structural mechanics. *Rev. Européenne Elém. Finis*, 3(2):281–297. *cited on page 117*

[43] Cochelin, B., Damil, N., and Potier-Ferry, M. (1994b). Asymptotic-numerical methods and Padé approximants for non-linear elastic structures. *Int. J. Numer. Meth. Engng.*, 37(7):1187–1213. *cited on page 117*

[44] Cooley, J. and Tukey, J. (1965). An algorithm for the machine computation of the complex Fourier series. *Math. Comp.*, 19(90):297–301. *cited on page 62*

[45] Crandall, S. (1970). The role of damping in Vibration Theory. *Numer. Algor.*, 11(1):3–18. *cited on page 56*

[46] De Lathauwer, L., De Moor, B., and Vanderwalle, J. (2000). A multilinear Singular Value Decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278. *cited on page 25, 28 and 139*

[47] De Silva, V. and Lim, L.-H. (2008). Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM J. Matrix Anal. Appl.*, 30(3):1084–1127. *cited on page 13 and 27*

[48] Dorival, O., Rouch, P., and Allix, O. (2006). A substructured version of the Variational Theory of Complex Rays dedicated to the calculation of assemblies with dissipative joints in the medium-frequency range. *Eng. Comput.*, 23(7-8):729–748. *cited on page 47*

[49] Duffy, D. G. (2001). *Green's functions with applications*. Studies in Advanced Mathematics. Chapman & Hall/CRC, Boca Raton, FL. *cited on page 93*

[50] Eftang, J. L. (2011). *Reduced basis methods for parametrized partial differential equations*. PhD thesis, Norwegian University of Science and Technology, Trondheim, Norway. *cited on page 6*

[51] Everson, R. and Sirovich, L. (1995). The Karhunen-Loève procedure for gappy data. *J. Opt. Soc. Am.*, 12(8):1657–1664. *cited on page 125*

[52] Falcó, A. and Nouy, A. (2011). A Proper Generalized Decomposition for the solution of elliptic problems in abstract form by using a functional Eckart-Young approach. *J. Math. Anal. Appl.*, 376:469–480. *cited on page 13 and 31*

[53] Falcó, A. and Nouy, A. (2012). Proper Generalized Decomposition for nonlinear convex problems in tensor Banach spaces. *Numer. Math.*, 121(3):503–530. *cited on page 31 and 32*

[54] Feng, Z., Chen, J., and Zhang, Y. (2010). Real-time solution of heat conduction in a finite slab for inverse analysis. *Int. J. Therm. Sci.*, 49(5):762–768. *cited on page 85*

[55] Georgiou, I. and Schwartz, I. (2002). Dynamics of large scale coupled structural/mechanical systems: a singular perturbation/Proper Orthogonal Decomposition approach. 59(4):1178–1207. *cited on page 49*

[56] Germoso, C., Aguado, J., Fraile, A., Alarcón, E., and Chinesta, F. (2015). Efficient PGD-based dynamic calculation of non-linear soil behavior. *Preprint.* *cited on page 81 and 140*

[57] Ghnatios, C., Masson, F., Huerta, A., Leygue, A., Cueto, E., and Chinesta, F. (2012). Proper Generalized Decomposition based dynamic data-driven control of thermal processes. *Comput. Meth. Appl. Mech. Engrg.*, 213-216:29–41. *cited on page 40 and 85*

[58] Girault, M., Videcoq, E., and Petit, D. (2010). Estimation of time-varying heat sources through inversion of a low order model built with the modal identification method from in-situ temperature measurements. *Int. J. Heat Mass Transfer*, 53(1-3):206–219. *cited on page 85*

[59] Gonzalez, D., Alfaro, I., Quesada, C., Cueto, E., and Chinesta, F. (2015). Computational Vademecums for the real-time simulation of haptic collision between nonlinear solids. *Comput. Meth. Appl. Mech. Engrg.*, 283:210–223. *cited on page 40*

[60] Gonzalez, D., Cueto, E., and Chinesta, F. (2014). Real-time direct integration of reduced solid dynamics equations. *Int. J. Numer. Meth. Engng.*, 99(9):633–653. *cited on page 40*

[61] Grasedyck, L. (2010). Hierarchical Singular Value Decomposition of tensors. *SIAM J. Matrix Anal. Appl.*, 31(4):2029–2054. *cited on page 26, 28 and 139*

[62] Grasedyck, L., Kressner, D., and Tobler, C. (2013). A literature survey of low-rank tensor approximation techniques. *arXiv:1302.7121.* *cited on page 5, 21, 23, 30 and 31*

[63] Gunzburger, M., Peterson, J., and Shadid, J. (2007). Reduced-order modeling of time-dependent pdes with multiple parameters in the boundary data. *Comput. Methods Appl. Mech. Engrg.*, 196(4-6):1030–1047. *cited on page 17*

[64] Hackbusch, W. (2012). *Tensor spaces and numerical tensor calculus.* Springer Series in Computational Mathematics. Springer-Verlag, Berlin-Heidelberg, 1st edition. *cited on page 7 and 22*

[65] Hackbusch, W. and Kühn, S. (2009). A new scheme for the tensor representation. *J. Fourier Anal. Appl.*, 15(5):706–722. *cited on page 23 and 26*

[66] Harshman, R. (1970). Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis. 16:1–84. *cited on page 23 and 24*

[67] Hastad, J. (1990). Tensor rank is NP-complete. *J. Algorithms*, 11:644–654. *cited on page 27*

[68] Hetmaniuk, U., Tezaur, R., and Farhat, C. (2012). Review and assessment of interpolatory model order reduction methods for frequency response structural dynamics and acoustics problems. *Int. J. Numer. Meth. Engng.*, 90(13):1636–1662. *cited on page 8 and 47*

[69] Hinze, M. and Volkwein, S. (2005). Proper Orthogonal Decomposition surrogate models for nonlinear dynamical systems: error estimates and suboptimal control. 45:261–306. *cited on page 49*

[70] Hughes, T. and Hulbert, G. (1988). Space-time finite element methods for elastodynamics: Formulations and error estimates. *Comput. Meth. Appl. Mech. Engrg.*, 66(3):339–363. *cited on page 64*

[71] Hughes, T. J. R. (2000). *The finite element method: linear static and dynamic finite element analysis*. Dover Publications, New York. Corrected reprint of the 1987 original, Prentice Hall, Englewood Cliffs, NJ. *cited on page 7, 10 and 47*

[72] Hulbert, G. and Hughes, T. (1990). Space-time finite element methods for second-order hyperbolic equations. *Comput. Meth. Appl. Mech. Engrg.*, 84(3):327–348. *cited on page 64*

[73] Ishihara, K., Yoshida, N., and Tsujino, S. (1985). Modeling of stress-strain relations of soils in cyclic loading. In Kawamoto, T. and Itikawa, Y., editors, *Fifth International Conference on Numerical Methods in Geomechanics, Nagoya, Japan*, volume 1, pages 373–380. *cited on page 81*

[74] Jaishankar, A. and McKinley, G. (2012). Power-law rheology in the bulk and at the interface: quasi-properties and fractional constitutive equations. *Proc. R. Soc. A*, 469(2149). *cited on page 56*

[75] Jolliffe, I. (2002). *Principal Component Analysis*. Springer Series in Statistics. Springer-Verlag, New York, 2nd edition. *cited on page 6*

[76] Kammler, D. W. (2007). *A first course in Fourier analysis*. Cambridge University Press, Cambridge, 2nd edition. *cited on page 93*

[77] Kiers, H. (2000). Towards a standardized notation and terminology in multiway analysis. *J. Chemometrics*, 14:105–122. *cited on page 22*

[78] Kilbas, A., Srivastava, H., and Trujillo, J. (2006). *Theory and applications of fractional differential equations*. North-Holland Mathematics Studies. Elsevier, 1st edition. *cited on page 56*

[79] Kim, T. (1998). Frequency-domain Karhunen-Loève method and its application to linear dynamic systems. 36(11):2117–2123. *cited on page 49*

[80] Kolda, T. and Bader, B. (2009). Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500. *cited on page 5, 22, 23 and 27*

[81] Kovalevsky, L., Ladevèze, P., and Riou, H. (2012a). The Fourier version of the Variational Theory of Complex Rays for medium-frequency acoustics. *Comput. Meth. Appl. Mech. Engrg.*, 225:142–153. *cited on page 47*

[82] Kovalevsky, L., Ladevèze, P., Riou, H., and Bonnet, M. (2012b). The Variational Theory of Complex Rays for three-dimensional Helmholtz problems. *J. Comput. Acoust.*, 20(4). *cited on page 47*

[83] Ladevèze, P. (1999). *Nonlinear computational structural mechanics*. Mechanical Engineering Series. Springer-Verlag, New York, 1st edition. *cited on page 117 and 122*

[84] Ladevèze, P., Arnaud, L., Rouch, P., and Blanzé, C. (2001). The Variational Theory of Complex Rays for the calculation of medium-frequency vibrations. *Eng. Comput.*, 18(1-2):193–214. *cited on page 47*

[85] Ladevèze, P., Passieux, J.-C., and Neron, D. (2010). The LATIN multiscale computational method and the Proper Generalized Decomposition. *Comput. Meth. Appl. Mech. Engrg.*, 199(21-22):1287–1296. *cited on page 117 and 122*

[86] Ladevèze, P. and Riou, H. (2005). Calculation of medium-frequency vibrations over a wide frequency range. *Comput. Methods Appl. Mech. Engrg.*, 194(27-29):3167–3191. *cited on page 47*

[87] Lavedèze, P. and Chamoin, L. (2011). On the verification of model reduction methods based on the Proper Generalized Decomposition. *Comput. Meth. Appl. Mech. Engrg.*, 200(23-24):2032–2047. *cited on page 33, 61 and 95*

[88] Lee, J. (2004). *A first course in combinatorial optimization*. Cambridge Texts in Applied Mathematics. Cambridge University Press, Cambridge. *cited on page 95*

[89] Leygue, A., Chinesta, F., Beringhier, M., Nguyen, T., Grandidier, J., Pesavento, F., and Schrefler, B. (2013). Towards a framework for non-linear thermal models in shell domains. *Int. J. Numer. Method H.*, 23(1):55–73. *cited on page 122*

[90] Loève, M. (1965). *Fonctions aléatoires du second ordre, in: Processus stochastiques et mouvement Brownien*. Gauthier-Villars, Paris, 2nd edition. *cited on page 6*

[91] Love, A. E. (1944). *A treatise on the Mathematical Theory of Elasticity*. Dover Publications, New York, NY, 4th edition. *cited on page 86 and 88*

[92] Maday, Y., Nguyen, N., Patera, A., and Pau, S. (2009). A general multipurpose interpolation procedure: the Magic Points. *CPPA*, 8(1):383–404. *cited on page 126*

[93] Maday, Y., Patera, A., and Turinici, G. (2002). A priori convergence theory for reduced-basis approximations of single-parametric elliptic partial differential equations. *J. Sci. Comput.*, 17(1-4):437–446. *cited on page 9*

[94] Maday, Y. and Ronquist, E. (2002). A reduced-basis element method. *C. R. Math.*, 335(2):195–200. *cited on page 6*

[95] Maday, Y. and Ronquist, E. (2004). The reduced basis element method: application to a thermal fin problem. *SIAM J. Sci. Comput.*, 26(1):240–258. *cited on page 85*

[96] Masson, F., Gonzalez, D., Cueto, E., and Chinesta, F. (2013). Proper Generalized Decomposition based Dynamic Data-Driven Applications Systems. *RIMNI*, 29(2):104–113. *cited on page 40*

[97] Modesto, D., Zlotnik, S., and Huerta, A. (2015). Proper Generalized Decomposition for parameterized Helmholtz problems in heterogeneous and unbounded domains: application to harbor agitation. *Comput. Methods Appl. Mech. Eng.* *cited on page 40 and 43*

[98] Moitinho de Almeida, J. P. (2013). A basis for bounding the errors of Proper Generalized Decomposition solutions in solid mechanics. *Int. J. Numer. Meth. Engng.*, 94(10):961–984. *cited on page 95*

[99] Monteyne, G., Javed, S., and Vandersteen, G. (2014). Heat transfer in a borehole heat exchanger: Frequency domain modeling. *Int. J. Heat Mass Transfer*, 69:129–139. *cited on page 85*

[100] Murono, Y. and Nogami, Y. (2006). Shear stress-shear strain relationship taking into account S-shape hysteresis loop. In *Symposium of 12th Japan Earthquake Engineering*, volume 12, pages 494–497. *cited on page 81*

[101] Newmark, N. (1959). A method of computation for structural dynamics. *J. Eng. Mech.*, 85:67–94. *cited on page 64*

[102] Niroomandi, S., Alfaro, I., Cueto, E., and Chinesta, F. (2008). Real-time deformable models of non-linear tissues by model reduction techniques. *Comput. Meth. Prog. Bio.*, 91(3):223–231. *cited on page 8*

[103] Niroomandi, S., Alfaro, I., Cueto, E., and Chinesta, F. (2010). Model order reduction for hyperelastic materials. *Int. J. Numer. Meth. Engng.*, 81(9):1180–1206. *cited on page 8*

[104] Nobile, F., Tempone, R., and Webster, C. (2008). A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM J. Numer. Anal.*, 46(5):2309–2345. *cited on page 6*

[105] North, G., Bell, T., Cahalan, R., and Moeng, F. (1982). Sampling errors in the estimation of empirical orthogonal functions. *Mon. Weather Rev.*, 110:699–706. *cited on page 8*

[106] Nouy, A. (2010). A priori model reduction through Proper Generalized Decomposition for solving time-dependent partial differential equations. *Comput. Meth. Appl. Mech. Engrg.*, 199:1603–1626. *cited on page 13, 14, 31, 33, 34, 35, 37 and 95*

[107] Parés, N., Díez, P., and Huerta, A. (2008). Bounds of functional outputs for parabolic problems. I. Exact bounds of the discontinuous Galerkin time discretization. *Comput. Meth. Appl. Mech. Engrg.*, 197(19-20):1641–1660. *cited on page 88*

[108] Park, H. and Cho, D. (1996a). The use of the Karhunen-Loève decomposition for the modeling of distributed parameter systems. *Chem. Eng. Sci.*, 51(1):81–98. *cited on page 85*

[109] Park, H. and Cho, D. (1996b). The use of the Karhunen-Loève decomposition for the modelling of distributed parameter systems. *Chem. Engineer. Science*, 51(1):81–98. *cited on page 8*

[110] Pinnau, R. and Schulze, A. (2007). Model reduction techniques for frequency averaging in radiative heat transfer. *J. Comput. Phys.*, 226(1):712–731. *cited on page 85*

[111] Prud'homme, C., Rovas, D., Veroy, K., Machiels, L., Maday, Y., Patera, A., and Turinici, G. (2001). Reliable real-time solution of parametrized partial differential equations: Reduced-Basis Output Bound Methods. *J. Fluids Eng.*, 124(1):70–80. *cited on page 9 and 139*

[112] Prulière, E., Chinesta, F., and Ammar, A. (2010). On the deterministic solution of multidimensional parametric models using the Proper Generalized Decomposition. *Math. Comput. Simul.*, 81(4):791–810. *cited on page 6, 17 and 97*

[113] Prulière, E., Chinesta, F., Ammar, A., Leygue, A., and Poitou, A. (2013). On the solution of the heat equation in very thin tapes. *Int. J. Therm. Sci.*, 65(0):148–157. *cited on page 39 and 85*

[114] Quesada, C., Alfaro, I., Gonzalez, D., Cueto, E., and Chinesta, F. (2014). PGD-based model reduction for surgery simulation: solid dynamics and contact detection. *Lect. Notes Comput. Sc.*, 8789:193–202. *cited on page 40*

[115] Quraishi, S., C., S., and V., M. (2014). Solution of large scale parametric eigenvalue problems arising from brake squeal modeling. In Wiley-VCH Verlag GmbH & Co. KGaA, W., editor, *Proceedings in Applied Mathematics and Mechanics (PAMM)*, volume 14, pages 891–892. *cited on page 49 and 54*

[116] Riesz, F. and Sz.-Nagy, B. (1990). *Functional analysis*. Dover Books on Mathematics. Dover Publications, New York, reprint edition. *cited on page 11*

[117] Riou, H., Ladevèze, P., and Sourcis, B. (2008). The multiscale VTCR approach applied to acoustics problems. *J. Comput. Acoust.*, 16(4):487–505. *cited on page 47*

[118] Rouch, P. and Blanzé, C. (2014). Vibrational analysis of structures with stochastic interfaces in the medium-frequency range: Experimental validation on a touch screen. *J. Sound Vibr.*, 333(6):1612–1628. *cited on page 47*

[119] Rouch, P. and Ladevèze, P. (2003). The Variational Theory of Complex Rays: A predictive tool for medium-frequency vibrations. *Comput. Meth. Appl. Mech. Engrg.*, 192(28-30):3301–3315. *cited on page 47*

[120] Rozza, G., Huynh, D., and Patera, A. (2008). Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations – application to transport and continuum mechanics. *Arch. Comput. Methods Eng.*, 15(3):229–275. *cited on page 18*

[121] Rylatt, D. and O'Donovan, T. (2012). Time and frequency domain investigation of the heat transfer to a synthetic air jet. *JPCS*, 395(1). *cited on page 85*

[122] Shannon, C. (1949). Communication in the presence of noise. *Proc. Institute of Radio Engineers*, 18(1):10–21. Reprint as classic paper in: Proceedings of the IEEE, Vol. 86, No. 2, 447–457, February 1998. *cited on page 57, 100 and 105*

[123] Sierociuk, D., Dzieliński, A., Sarwas, G., Petras, I., Podlubny, I., and Skovranek, T. (2013). Modelling heat transfer in heterogeneous media using fractional calculus. *Philos. Trans. R. Soc. A-Math. Phys. Eng. Sci.*, 371(1990). *cited on page 85*

[124] Temam, R. (1979). *Navier-Stokes equations. Theory and numerical analysis*, volume 2 of *Studies in Mathematics and its Applications*. North-Holland Publishing Co., Amsterdam-New York, revised edition. *cited on page 88*

[125] Temam, R. (1997). *Infinite-dimensional dynamical systems in mechanics and physics*, volume 68 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 2nd edition. *cited on page 17*

[126] Tisseur, F. and Meerbergen, K. (2001). The Quadratic Eigenvalue Problem. *SIAM Rev.*, 43(2):235–286. *cited on page 48, 51 and 52*

[127] Tucker, L. (1964). The extension of factor analysis to three-dimensional matrices. In Holt, R. and Winston, N. Y., editors, *Contributions to Mathematical Psychology*, pages 110–127. *cited on page 23 and 25*

[128] Vega, J., Fraile, A., Alarcon, E., and Hermanns, L. (2012). Dynamic response of underpasses for high-speed train lines. *J. Sound Vibr.*, 331(23):5125–5140. *cited on page 88*

[129] Verdugo, F. and Díez, P. (2012). Computable bounds of functional outputs in linear visco-elastodynamics. *Comput. Methods Appl. Mech. Eng.*, 245-246:313–330. *cited on page 64*

[130] Veroy, K. and Patera, A. (2005). Certified real-time solution of the parametrized steady incompressible navier-stokes equations: Rigorous reduced-basis a posteriori error bounds. *Int. J. Numer. Meth. Fluids*, 47(8-9):773–788. *cited on page 18*

[131] Videcoq, E., Quemener, O., Lazard, M., and Neveu, A. (2008). Heat source identification and on-line temperature control by a branch eigenmodes reduced model. *Int. J. Heat Mass Transfer*, 51(19-20):4743–4752. *cited on page 85*

[132] Willcox, K. (2006). Unsteady flow sensing and estimation via the gappy Proper Orthogonal Decomposition. *Comput. Fluids*, 35(2):208–226. *cited on page 125*

[133] Willcox, K. and Peraire, J. (2002). Balanced model reduction via the Proper Orthogonal Decomposition. *AIAA Journal*, 40(11):2323–2330. *cited on page 49*

[134] Wirtz, D. and Haasdonk, B. (2012). Efficient a posteriori error estimation for nonlinear kernel-based reduced systems. *Syst. Contr. Lett.*, 61(1):203–2011. *cited on page 6*

[135] Wirtz, D. and Haasdonk, B. (2013). An improved vectorial Kernel Orthogonal Greedy algorithm. *Comput. Methods Appl. Mech. Eng.*, 6:83–100. *cited on page 6*

[136] Zienkiewicz, O. C. and Taylor, R. L. (2000a). *The finite element method. Vol. 1 The basis*. Butterworth Heinemann, Oxford, 5th edition. *cited on page 51*

[137] Zienkiewicz, O. C. and Taylor, R. L. (2000b). *The finite element method. Vol. 2 Solid mechanics*. Butterworth Heinemann, Oxford, 5th edition. *cited on page 47, 48 and 121*

Appendices

Appendix A

Algebraic form of the structural dynamics (x, ω, μ)-operator

In order to obtain the tensor operator used to solve the tensor problem Eq. (2.28), let us consider the left-hand side of Eq. (2.27). Inserting into that equation both $\hat{\boldsymbol{u}}$ and $\hat{\boldsymbol{w}}$, defined in Eq. (2.26) (Galerkin approach), we get:

$$\int_{I_\omega}\int_{I_\mu}\sum_{i_s,j_s}^{N}\sum_{i_\omega,j_\omega}^{N_\omega}\sum_{i_\mu,j_\mu}^{N_\mu}\bar{w}_{i_s,i_\omega,i_\mu}v_\omega^{i_\omega}\otimes v_\mu^{i_\mu}\otimes\left[(\boldsymbol{v}_s^{i_s})^H\,\boldsymbol{A}\,\boldsymbol{v}_s^{j_s}\right]\otimes v_\omega^{j_\omega}\otimes v_\mu^{j_\mu}\,u_{j_s,j_\omega,j_\mu}\,d\omega\,d\mu.$$

Substituting $\boldsymbol{A}$ and grouping the terms conveniently:

$$\begin{aligned}\sum_{i_s,j_s}^{N}\sum_{i_\omega,j_\omega}^{N_\omega}\sum_{i_\mu,j_\mu}^{N_\mu}\bar{w}_{i_s,i_\omega,i_\mu}\Bigg(&-(\boldsymbol{v}_s^{i_s})^H\boldsymbol{M}\boldsymbol{v}_s^{j_s}\otimes\int_{I_\omega}v_\omega^{i_\omega}v_\omega^{j_\omega}\omega^2\,d\omega\otimes\int_{I_\mu}v_\mu^{i_\mu}v_\mu^{j_\mu}\,d\mu+\ldots\\&\ldots+\mathrm{i}(\boldsymbol{v}_s^{i_s})^H\boldsymbol{C}\boldsymbol{v}_s^{j_s}\otimes\int_{I_\omega}v_\omega^{i_\omega}v_\omega^{j_\omega}\omega\,d\omega\otimes\int_{I_\mu}v_\mu^{i_\mu}v_\mu^{j_\mu}\mu\,d\mu+\ldots\\&\ldots+(\boldsymbol{v}_s^{i_s})^H\boldsymbol{K}\boldsymbol{v}_s^{j_s}\otimes\int_{I_\omega}v_\omega^{i_\omega}v_\omega^{j_\omega}\,d\omega\otimes\int_{I_\mu}v_\mu^{i_\mu}v_\mu^{j_\mu}\,d\mu\Bigg)u_{j_s,j_\omega,j_\mu}.\end{aligned}\tag{A.1}$$

Since vectors $\boldsymbol{v}_s^{1\le i\le N}$ have been defined in Section 1.1.3 as the canonical basis in $\mathbb{C}^N$, we have:

$$(\boldsymbol{v}_s^{i_s})^H\boldsymbol{M}\boldsymbol{v}_s^{j_s}=m_{i_sj_s},\quad(\boldsymbol{v}_s^{i_s})^H\boldsymbol{C}\boldsymbol{v}_s^{j_s}=c_{i_sj_s}\quad\text{and}\quad(\boldsymbol{v}_s^{i_s})^H\boldsymbol{K}\boldsymbol{v}_s^{j_s}=k_{i_sj_s},$$

being $m_{i_sj_s}, c_{i_sj_s}$ and $k_{i_sj_s}$ the (i_s, j_s)-th entries of the mass, damping and stiffness matrices, respectively. Eq. (A.1) allows defining the operator:

$$\boldsymbol{A}=-\boldsymbol{M}\otimes\boldsymbol{B}_\omega\otimes\boldsymbol{M}_\mu+\mathrm{i}\boldsymbol{C}\otimes\boldsymbol{D}_\omega\otimes\boldsymbol{D}_\mu+\boldsymbol{K}\otimes\boldsymbol{M}_\omega\otimes\boldsymbol{M}_\mu,$$

where matrices $\boldsymbol{M}_\omega$, $\boldsymbol{B}_\omega$, $\boldsymbol{D}_\omega \in \mathbb{R}^{N_\omega \times N_\omega}$ and $\boldsymbol{M}_\mu$,$\boldsymbol{D}_\mu \in \mathbb{R}^{N_\mu \times N_\mu}$ are defined respectively by their entries as follows:

$$
\begin{aligned}
(m_\omega)_{i_\omega j_\omega} &= \int_{I_\omega} v_\omega^{i_\omega}\, v_\omega^{j_\omega}\, d\omega, \\
(b_\omega)_{i_\omega j_\omega} &= \int_{I_\omega} \omega^2\, v_\omega^{i_\omega}\, v_\omega^{j_\omega}\, d\omega, \\
(d_\omega)_{i_\omega j_\omega} &= \int_{I_\omega} \omega\, v_\omega^{i_\omega}\, v_\omega^{j_\omega}\, d\omega \quad \text{with} \quad 1 \le i_\omega, j_\omega \le N_\omega,
\end{aligned}
$$

and

$$
\begin{aligned}
(m_\mu)_{i_\mu j_\mu} &= \int_{I_\mu} v_\mu^{i_\mu}\, v_\mu^{j_\mu}\, d\mu, \\
(d_\mu)_{i_\mu j_\mu} &= \int_{I_\mu} \mu\, v_\mu^{i_\mu}\, v_\mu^{j_\mu}\, d\mu, \quad \text{with} \quad 1 \le i_\mu, j_\mu \le N_\mu.
\end{aligned}
$$

Appendix B

Green's function problem for a parabolic operator

Although what follows is well known it is not standard to write the Green's function problem associated to the homogenous heat equation with homogeneous Dirichlet and non-homogeneous Neumann boundary conditions. The adjoint Green's function problem associated to (3.2) is detailed in (3.3) where its last equation, namely $G = 0$ on $\Omega \times [t_0, T[$, corresponds to the *causality* condition. Green's identity can be written as

$$\int_\Omega \int_0^{t_0} \Big[G(\nabla^2 u - \partial_t u) - u(\nabla^2 G + \partial_t G) \Big] dt d\Omega \\ = \int_{\partial\Omega} \int_0^{t_0} \Big[G(\boldsymbol{n} \cdot \nabla) u - u(\boldsymbol{n} \cdot \nabla) G \Big] dt d\Gamma + \int_\Omega \Big[uG|_{t=0} - uG|_{t=t_0} \Big] d\Omega. \quad \text{(B.1)}$$

Consequently, using (3.2) and (3.3) in the previous identity (B.1), a representation for $u(\boldsymbol{x}_0, t_0)$ with $(\boldsymbol{x}_0, t_0) \in \Gamma_N \times]0, T[$ is obtained, namely

$$u(\boldsymbol{x}_0, t_0) = \int_{\Gamma_N} \int_0^{t_0} G(\boldsymbol{x}, t; \boldsymbol{x}_0, t_0)\, q(\boldsymbol{x}, t) dt d\Gamma + \int_\Omega u_0(\boldsymbol{x})\, G(\boldsymbol{x}, 0; \boldsymbol{x}_0, t_0) d\Omega, \quad \text{(B.2)}$$

where now it can be clearly identified that the *causality* condition implies that the solution at time t_0 cannot depend on any of its values at later times. Recall T can be arbitrarily large.

Appendix C

Reciprocity proof in the frequency domain for even and odd real excitations

This appendix is aimed to prove the reciprocity principle, recall (3.13)

$$\langle \hat{q}_1, \hat{u}_2 \rangle = \langle \hat{q}_2, \hat{u}_1 \rangle, \tag{C.1}$$

when both $\hat{q}_1$ and $\hat{q}_2$ are real. As discussed in Section 3.2.5, reciprocity, holds if the two conditions stated in equation (3.14) are verified, namely

$$\big(\nabla \hat{u}_1, \nabla \hat{u}_2\big) = \big(\nabla \hat{u}_2, \nabla \hat{u}_1\big) \text{ and } \big(\hat{u}_1, \hat{u}_2\big) = \big(\hat{u}_2, \hat{u}_1\big).$$

Recalling that $\big(\,\cdot\,,\,\cdot\,\big)$ is the $\mathcal{L}^2$ scalar product of *complex* functions in Ω, see (3.9), these conditions are equivalent to

$$\Im\Big[\big(\nabla \hat{u}_1, \nabla \hat{u}_2\big)\Big] = 0 \text{ and } \Im\Big[\big(\hat{u}_1, \hat{u}_2\big)\Big] = 0.$$

Since $\hat{u}_1$ and $\hat{u}_2$ give values in $\mathbb{C}$, one can define the real functions a_1, b_1, a_2 and b_2 such that $\hat{u}_1 = a_1 + ib_1$ and $\hat{u}_2 = a_2 + ib_2$, and the previous expressions are equivalent to

$$\big(\nabla a_1, \nabla b_2\big) = \big(\nabla b_1, \nabla a_2\big) \tag{C.2a}$$

$$\big(a_1, b_2\big) = \big(b_1, a_2\big). \tag{C.2b}$$

Recall also that $\hat{u}_1$ and $\hat{u}_2$ are the corresponding solutions of the weak problem (3.8) for the two excitations $\hat{q}_1$ and $\hat{q}_2$ and for any imposed frequency ω. By definition, a_i and b_i for $i = 1, 2$ also belong to space of trial and test functions, namely $\mathcal{H}^1_{\Gamma_D}$. Thus, equation (3.8)

for the pair $\{\hat{u}_1, \hat{q}_1\}$ can be particularized as follows:

$$\begin{aligned}\big(\nabla\hat{u}_1, \nabla a_2\big) + i\omega\big(\hat{u}_1, a_2\big) &= \big\langle \hat{q}_1, a_2\big\rangle,\\ \big(\nabla\hat{u}_1, \nabla b_2\big) + i\omega\big(\hat{u}_1, b_2\big) &= \big\langle \hat{q}_1, b_2\big\rangle,\end{aligned} \tag{C.3}$$

and likewise for $\{\hat{u}_2, \hat{q}_2\}$

$$\begin{aligned}\big(\nabla\hat{u}_2, \nabla a_1\big) + i\omega\big(\hat{u}_2, a_1\big) &= \big\langle \hat{q}_2, a_1\big\rangle,\\ \big(\nabla\hat{u}_2, \nabla b_1\big) + i\omega\big(\hat{u}_2, b_1\big) &= \big\langle \hat{q}_2, b_1\big\rangle.\end{aligned} \tag{C.4}$$

Recall the splitting of $\hat{u}_1$ into real and imaginary parts, $\hat{u}_1 = a_1 + ib_1$, to also split equations (C.3) into real and imaginary parts as

$$\big(\nabla a_1, \nabla a_2\big) - \omega\big(b_1, a_2\big) = \big\langle \hat{q}_1, a_2\big\rangle, \tag{C.5a}$$
$$\big(\nabla b_1, \nabla a_2\big) + \omega\big(a_1, a_2\big) = 0, \tag{C.5b}$$
$$\big(\nabla a_1, \nabla b_2\big) - \omega\big(b_1, b_2\big) = \big\langle \hat{q}_1, b_2\big\rangle, \tag{C.5c}$$
$$\big(\nabla b_1, \nabla b_2\big) + \omega\big(a_1, b_2\big) = 0, \tag{C.5d}$$

where the hypothesis that both $\hat{q}_1(\boldsymbol{x}, \omega)$ and $\hat{q}_2(\boldsymbol{x}, \omega)$ belong to $\mathbb{R}$ is used.

Likewise, for $\hat{u}_2 = a_2 + ib_2$ each equation in (C.4) is split into real and imaginary parts as

$$\big(\nabla a_2, \nabla a_1\big) - \omega\big(b_2, a_1\big) = \big\langle \hat{q}_2, a_1\big\rangle \tag{C.6a}$$
$$\big(\nabla b_2, \nabla a_1\big) + \omega\big(a_2, a_1\big) = 0 \tag{C.6b}$$
$$\big(\nabla a_2, \nabla b_1\big) - \omega\big(b_2, b_1\big) = \big\langle \hat{q}_2, b_1\big\rangle \tag{C.6c}$$
$$\big(\nabla b_2, \nabla b_1\big) + \omega\big(a_2, b_1\big) = 0 \tag{C.6d}$$

Recall now that the $\mathcal{L}^2$ scalar product is symmetric for any pair of real functions, more specifically $\big(u, v\big) = \big(v, u\big)$ for all u and $v \in \mathbb{R}$. Then subtract (C.6b) from (C.5b), and (C.6d) from (C.5d) to obtain the desired conditions (C.2). Thus if (C.2) are verified, then (3.14) also holds and reciprocity is demonstrated. Note that these results hold for any ω.

To further close these appendix, note that the other equations not used up to now also produce the same results. After subtracting (C.6a) from (C.5a) and (C.6c) from (C.5c), the following equations are obtained:

$$\begin{aligned}\omega\Big[\big(b_2, a_1\big) - \big(b_1, a_2\big)\Big] &= \big\langle \hat{q}_1, a_2\big\rangle - \big\langle \hat{q}_2, a_1\big\rangle,\\ \big(\nabla a_1, \nabla b_2\big) - \big(\nabla a_2, \nabla b_1\big) &= \big\langle \hat{q}_1, b_2\big\rangle - \big\langle \hat{q}_2, b_1\big\rangle.\end{aligned}$$

However, both left-hand-sides in the previous equations are zero because they correspond to (C.2), which was just proven, and these equations become

$$\big\langle \hat{q}_1, a_2\big\rangle = \big\langle \hat{q}_2, a_1\big\rangle \text{ and } \big\langle \hat{q}_1, b_2\big\rangle = \big\langle \hat{q}_2, b_1\big\rangle,$$

which corresponds to split (C.1) into real and imaginary parts using the definitions splitting of $\hat{u}_1 = a_1 + ib_1$ and $\hat{u}_2 = a_2 + ib_2$.

l'université nantes angers le mans

PÔLE DE RECHERCHE ET D'ENSEIGNEMENT SUPÉRIEUR

Centrale Nantes

Thèse de Doctorat

José V. AGUADO

Stratégies avancées pour la formulation séparée de problèmes dans le cadre de la méthode Proper Generalized Decomposition

Advanced strategies for the separated formulation of problems in the Proper Generalized Decomposition framework

Résumé

Les travaux présentés s'inscrivent dans le cadre de la méthode « Proper Generalized Decomposition » (PGD) pour la résolution numérique d'équations différentielles aux dérivées partielles. La performance numérique de la PGD est basée sur l'hypothèse d'existence d'une représentation séparée de rang faible de la solution, mais aussi des données du problème. Cette thèse s'intéresse à la formulation séparée de ces dernières. La démarche mise en œuvre examine deux approches différentes. La première consiste à chercher une reformulation permettant de s'affranchir du calcul de la représentation séparée. Le recours à une approche fréquentielle permet de passer outre la non séparabilité dans le domaine spatio-temporel des excitations de type « propagation d'ondes ». L'efficacité de cette approche est démontrée au travers de problèmes linéaires transitoires de dynamique des structures ainsi que de transfert thermique. De plus, le passage dans le domaine fréquentiel donnant un caractère symétrique aux équations nous permet de prouver le principe de réciprocité. Ce principe est mis à profit pour développer un outil temps-réel de surveillance de procédés. La seconde approche examinée consiste à développer une méthode numérique innovante pour calculer la représentation séparée de fonctions à plusieurs variables. La stratégie choisie est basée sur une technique d'interpolation qui permet de réduire la complexité numérique associée aux techniques de projection traditionnelles. La performance numérique de cette méthode est démontrée au travers de son application à des coefficients non séparés et non linéaires.

Mots clés
Représentations séparées, PGD, réduction de modèles, approximation tensorielle de rang faible, approche fréquentielle, méthode d'interpolation empirique, principe de réciprocité.

Abstract

This work fits within the Proper Generalized Decomposition (PGD) framework for the numerical solution of multidimensional partial differential equations. The computational performance of the PGD relies on the assumption that not only the solution but also the problem data admit a low-rank separated representation. This work is concerned with the separated formulation of the problem data. The solution to these separability issues is sought in two ways. The first strategy consists in looking for an efficient formulation such as to encompass the need of computing the separated representation. This is achieved by means of a frequency-based approach which allows overcoming the non-separability in the space-time domain of wave-like excitations. The efficiency of the approach is demonstrated in both transient linear structural dynamics and heat transfer problems. Besides, the reciprocity principle can be proven thanks to the symmetrization introduced by the frequency-based formulation, yielding a direct application for the real-time monitoring of processes. The second strategy devises an innovative method to compute the separated representation of multivariate functions. An a priori interpolation-based approach is designed in order to effectively reduce the computational complexity associated to traditional projection-based methods. The performance of this method is demonstrated with application to non-separated and nonlinear coefficients.

Key Words
Separated representations, PGD, model order reduction, low-rank tensor approximations, frequency approaches, empirical interpolation method, reciprocity principle