École Centrale de Nantes

Master 2 in Computational Mechanics

Model order reduction for parametric PDEs: hyper-reduction through FOCUSS algorithm

Advisors:

Dr. Domenico Borzacchiello

Prof. Stefano Berrone

Candidate:

Mattia Manucci

Academic Year

2017-2018

Contents

List of Figures

List of Tables

Introduction

This master thesis work is collocated in the general field of model order reduction for numerical simulations, that, explained in few words, is a field concerning the development of numerical methods aiming to speed up the computation, paying the price of the introduction of controllable error in the simulation. More precisely, we work with the reduced basis (RB) techniques, recognised to be efficient and consolidated methods in the model order reduction field. Hence, they are very frequently employed and there is a rich literature about them very easy to access for those interested. Nevertheless, in order to have a self-consistent work that should be understood by anyone with knowledges in numerical analysis and finite element method (FEM), we start this thesis with a chapter describing from a mathematical point of view the reduced basis methods. In writing the chapter we consulted the book of A. Quarteroni [19], since it contains a self-consistent and compact summary over the reduced basis that perfectly adapts to this thesis matters.

Actually there are cases in which reduced basis techniques lose their efficiency, that is trying to apply them in the contest of non-linear[1] parametrized partial differential equations (PDEs). Non-linear PDEs are often necessary in the simulation of complex physical and industrial process.

To recover the efficiency of the reduced basis methods several approaches have been proposed during the time but, until know, for what we know, all of them show drawbacks which compromise their use in some applications. Chapter 2 explores the state of art of this non-linear reduction techniques, for which was coined the term *hyper-reduction* methods, highlighting the issues in which they stumble. We refer to part of the work presented in [10] for this chapter.

It is exactly in the topic of hyper-reduction where the core our work lives. In-

---

[1] It would be better to say "non-affine", continuing the read will be clear the reason of this statement.

deed we readapt a method, called FOCUSS, proposed in the contest of sparse signal reconstruction to the contest of Model Order Reduction for non-linear parametrized PDEs; this new technique is such that overcomes the issues of the other hyper-reduction approaches. The power and efficiency of the readapted FOCUSS is accurately tested on three problems, numerically implemented through Matlab. The first problem is built "ad hoc" and we call it the *"polynomial test case"*, whose reason would be clear after presenting the problem. The second and the third are two typical FEM problems involving non-linear PDEs, one concerning the thermal diffusion, the other the poro-elasticity. On all these three problems, we measure not only the performances but also we compare them with some of the other hyper-reduction techniques results mentioned in Chapter 2.
Chapter 3 is written, at first, to describe the modifications done on the FOCUSS' version of [8] and then to test it on the polynomial problem. Instead in Chapter 4 we report the results of applying FOCUSS on the two problems coming from a FEM discretization. In both these last two chapters we also dedicate some space to topics directly related to our works, such as the use of the ℓ^1-norm in the contest of vector sparsification or the construction of snapshots matrix for the second reduction stage.

Chapter 1

Model Order Reduction: the Reduced Basis method

The constant increase of available computational power, accompanied by the progressive improvement of algorithms for solving large linear systems, make nowadays possible the numerical simulation of complex, multiscale and multiphysics phenomena by means of high-fidelity (or full-order) approximation techniques such as the finite element method, finite volumes, finite differences or spectral methods. However, this might be quite demanding, because it involves up to $O(10^6 - 10^9)$ degrees of freedom and several hours (or even days) of CPU time, also on powerful hardware parallel architectures.

High-fidelity approximation techniques can become prohibitive when we expect them to deal quickly and efficiently with the repetitive solution of partial differential equations (PDEs). This is, e.g., the case of PDEs depending on parameters, the so called parametrized PDEs. The nature of the input parameters depend on which kind of physical model the PDEs refers, e.g., (i) in nonlinear viscous flows governed by Navier-Stokes equations the Reynolds number can be varied to study the flow, (ii) the conductivity in the nonlinear thermal diffusion or (iii) the permeability in a poroelasticity model (which is, indeed, the benchmark utilized in Chapter 4). In these three cases, evaluating the behaviour of the system by means of a high-fidelity technique, such as the finite element (FE) method, is computationally expensive because it entails the solution of very large (nonlinear) algebraic systems, arising from the discretization of the underpinning PDE.

Concerning this type of problems, *reduced-order modelling* – alternatively named *model order reduction* in the literature – is a generic expression used

to identify any approach aimed to replace the high-fidelity problem by one featuring a much lower numerical complexity with the introduction of an error in relation to the high-fidelity solution; this error can be kept below a prescribed tolerance. *Reduced-order model* (ROM), given an instance of the parameter, is able to evaluate the solution at a cost that is independent from the dimension of the original high-fidelity problem. The ROM techniques basic idea is the assumption, often verified by reality, that the behaviour of a complex system can be described by a small number of dominant modes. In particular, among the reduced-order modelling techniques, a remarkable instance is represented by *reduced basis* (RB) methods. The strategy is to resolve the high-fidelity problem only for few instances of the input parameters during an *Offline* phase computationally onerous, with the aim of building a set of base solutions (i.e. a *reduced base*), of a much smaller dimension compared to the number of degree of freedom (Dof) of the high-fidelity problem. These base functions will correspond to the numerical solutions of the high-fidelity problem for specific values, appropriately selected, of the parameters. After that, for every new vector of the input parameters, the corresponding solution will be searched through an opportune linear combination of the functions of the reduced base. The unknown coefficients of this combination will be obtained during the *Online* phase thanks to the solution of a reduced problem generated through a Galerkin projection on the reduced space; this Online stage will only require the solution of a linear system with an associated small dimension matrix.
It is important to underline that the RB methods do not replay the high-fidelity methods, but they are built over these. Therefore the reduced solution does not approximate directly the exact solution of the problem but rather its high-fidelity approximation.

1.1 Parametrized PDEs

Before introducing the main features of the reduced basis methods, it is useful to set up the theory for the general problem. Let's indicate with $\mathcal{D} \subset \mathbb{R}^p$, $p \geq 1$, a set of input parameters that may describe the physical properties of the system, boundary condition, source terms, or the geometry of the computational domain. The problems treated by ROM methods can be described theoretically in the following way:
given $\boldsymbol{\mu} \in \mathcal{D}$, evaluate the variable of interest $s(\boldsymbol{\mu}) = J(u(\boldsymbol{\mu}))$ where $u(\boldsymbol{\mu}) \in$

$V = V(\Omega)$ is the solution of the following parametrized PDE

$$L(\boldsymbol{\mu})u(\boldsymbol{\mu}) = F(\boldsymbol{\mu}); \tag{1.1}$$

where $\Omega \subset \mathbb{R}^d,\ d = 1, 2, 3$ stays for a regular domain, V is an opportune Hilbert space, V' its dual, $L(\boldsymbol{\mu}) : V \to V'$ a differential operator of the second order and $F(\boldsymbol{\mu}) \in V'$. The weak formulation of the problem (1.1) is given by: find $u(\boldsymbol{\mu}) \in V = V(\Omega)$ such that

$$a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f(v; \boldsymbol{\mu}) \quad \forall v \in V, \tag{1.2}$$

where the bilinear form is obtained from $L(\boldsymbol{\mu})$,

$$a(u, v; \boldsymbol{\mu}) = {}_{V'}\langle L(\boldsymbol{\mu})u, v\rangle_V \quad \forall u, v \in V, \tag{1.3}$$

while

$$f(v; \boldsymbol{\mu}) = {}_{V'}\langle F(\boldsymbol{\mu})u, v\rangle_V \tag{1.4}$$

is a linear and continuum form. It is assumed that, for every $\boldsymbol{\mu} \in D,\ a(\cdot, \cdot; \boldsymbol{\mu})$ is continuum and coercive, that is $\exists\ \bar{\gamma}, \alpha_0 > 0$:

$$\gamma(\boldsymbol{\mu}) = \sup_{u \in V} \sup_{v \in V} \frac{a(u, v; \boldsymbol{\mu})}{\|u\|_V \|v\|_V} < \bar{\gamma} < +\infty, \quad \alpha(\boldsymbol{\mu}) = \inf_{u \in V} \frac{a(u, u; \boldsymbol{\mu})}{\|u\|_V^2} \geq \alpha_0. \tag{1.5}$$

The functional J, which associate the unknown quantities of the equations with the variable of interest, is a linear and continuum form over V. Under these standard assumptions on a and f, (1.2) admits an unique solution, thanks to the Lax-Milgram theorem.

Finally another hypothesis is introduced, that is fundamental to ensure the computational efficiency of a reduced basis method: it is required that the parametric dependence of the bilinear form a and of the linear form f is affine in relation to $\boldsymbol{\mu}$, which means that the two forms can be expressed as:

$$a(w, v; \boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \Theta_a^q(\boldsymbol{\mu}) a^q(w, v) \quad \forall v, w \in V,\ \boldsymbol{\mu} \in \mathcal{D}, \tag{1.6}$$

$$f(v; \boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \Theta_f^q(\boldsymbol{\mu}) f^q(w) \quad \forall w \in V,\ \boldsymbol{\mu} \in \mathcal{D}, \tag{1.7}$$

where $\Theta_a^q : \mathcal{D} \to \mathbb{R},\ q = 1, ..., Q_a$ and $\Theta_f^q : \mathcal{D} \to \mathbb{R},\ q = 1, ..., Q_f$, are only functions of $\boldsymbol{\mu}$, while $a^q : V \times V \to \mathbb{R},\ f^q : V \to \mathbb{R}$ are respectively bilinear

and linear forms independent from $\boldsymbol{\mu}$. All of these quantities independent from $\boldsymbol{\mu}$ will be evaluated Offline, making the Online computation strongly less expensive.
Although the evaluation of the variable of interest had been one of the reasons for the development of the RB methods, for the purpose of this thesis it is enough to focus on the evaluation of the solution $u(\boldsymbol{\mu})$; for more details on the evaluation of the variable of interest it is remained to [18, 23].

1.1.1 A preliminary example

It is useful to introduce a simple example of parametrized problem which belongs to the case of physical parameters; more complex problems, which include both physical and geometrical parameters, require a deeper treatment, whose references can be found in [20]. In this example will be shown how bilinear and linear operator of a parametrized problem can be written according (1.6) and (1.7).
Let's consider a process of diffusion, advection and reaction of a substance inside a domain $\Omega \in \mathbb{R}^2$, on its boundary Dirichlet homogeneous conditions are imposed for simplicity; the concentration u of this substance satisfies the following problem

$$\begin{cases} -\nabla \cdot (\mathbb{K}\nabla u) + \mathbf{b} \cdot \nabla u + au = f & \text{in } \Omega, \\ u = 0 & \text{in } \partial\Omega, \end{cases} \tag{1.8}$$

where:

- $\mathbb{K} \in \mathbb{R}^{2\times 2}$ is a symmetric and positive definite matrix, which characterizes the diffusion property of the substance;
- $\mathbf{b}$ is a given advection field such that $\nabla \cdot \mathbf{b} = 0$;
- $a > 0$ is a positive coefficient of reaction.

For the analysis of this kind of problem it is reminded to Chapter 12 of [19].
In this case we are interested to solve the problem (1.8) for different values of the diffusion coefficients, of the advection field and of the reaction coefficient.
An example of parametrized coefficients is given by:

$$\mathbb{K} = \begin{pmatrix} \mu_1 & 0 \\ 0 & \mu_2 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} \cos\mu_3 \\ \sin\mu_3 \end{pmatrix}, \qquad a = \mu_4, \qquad f = 1 + \mu_5$$

which describes a variable diffusion (anisotropic if $\mu_1 \neq \mu_2$), an advection field with constant modulus but different direction (inclined with an angle of μ_3 respect the horizontal) and, more generally, different regimes where, depending on the values of the parameters μ_1, μ_2 e μ_4, the transport and/or the reaction can be dominant compared to the diffusion. At the same time, the variation of the parameter μ_5 represents a different contribute of the source term. Taking for example

$$\mu_1, \mu_2 \in (0.05, 1), \qquad \mu_3 \in (0, 2\pi), \qquad \mu_4 \in (0, 10), \qquad \mu_5 \in (0, 10)$$

problem (1.8) is well defined for every choice of $\boldsymbol{\mu} \in \mathcal{D} = (0.01, 1)^2 \times (0, 2\pi) \times (0, 10)^2$. A variable of interest can be the average of the concentration over the domain, given by

$$s(\boldsymbol{\mu}) = \int_\Omega u(\boldsymbol{\mu}) d\Omega.$$

Now the problem (1.8) can be re-written according the weak formulation (1.2) taking $V = H_0^1(\Omega)$,

$$\begin{aligned} a(w, v; \boldsymbol{\mu}) =& \mu_1 \int_\Omega \frac{\partial w}{\partial x}\frac{\partial v}{\partial x} d\Omega + \mu_2 \int_\Omega \frac{\partial w}{\partial y}\frac{\partial v}{\partial y} d\Omega \\ &+ \cos(\mu_3) \int_\Omega \frac{\partial w}{\partial x} v d\Omega + \sin(\mu_3) \int_\Omega \frac{\partial w}{\partial y} v d\Omega + \mu_4 \int_\Omega wv d\Omega \end{aligned} \tag{1.9}$$

and

$$f(v; \boldsymbol{\mu}) = (1 + \mu_5) \int_\Omega v d\Omega. \tag{1.10}$$

It can be easily observed that this problem is coercive for every choice of $\boldsymbol{\mu} \in \mathcal{D}$, since $-1/2\nabla \cdot \mathbf{b} + a = \mu_4 > 0$. In the examined case, a vector of $p = 5$ parameters describes the physical properties of interest; both a and f are affine in relation to $\boldsymbol{\mu}$, which means that they satisfy the property of affine parametric dependence (1.6)-(1.7): indeed in this case $Q_a = 5$, $Q_f = 1$,

$$\Theta_a^1(\boldsymbol{\mu}) = \mu_1, \;\; \Theta_a^2(\boldsymbol{\mu}) = \mu_2, \;\; \Theta_a^3(\boldsymbol{\mu}) = \cos(\mu_3), \;\; \Theta_a^4(\boldsymbol{\mu}) = \sin(\mu_3), \;\; \Theta_a^5(\boldsymbol{\mu}) = \mu_4,$$
$$\Theta_f^1(\boldsymbol{\mu}) = 1 + \mu_5,$$

and

$$a^1(w,v) = \int_\Omega \frac{\partial w}{\partial x}\frac{\partial v}{\partial x} d\Omega, \quad a^2(w,v) = \int_\Omega \frac{\partial w}{\partial y}\frac{\partial v}{\partial y} d\Omega,$$

$$a^3(w,v) = \int_\Omega \frac{\partial w}{\partial x} v d\Omega, \quad a^4(w,v) = \int_\Omega \frac{\partial w}{\partial y} v d\Omega, \quad a^5(w,v) = \int_\Omega w v d\Omega,$$

$$f^1(v) = \int_\Omega v d\Omega.$$

1.2 Main features of a reduced basis method

As previously observed, the term *reduced order model* for a parametrized PDE (as (1.2)) stays for any technique that, in function of the examined problem, aims to reduce the dimension of the algebraic system resulting from the discretization of the PDE. This can be achieved in two ways: (*i*) simplifying the physical model which is expressed by the set of parametrized PDEs, and (*ii*) trying to reduce the degrees of freedom of the discrete problem associated with the equations.

The reduced basis methods (RB) are a particular case of ROM methods, in which the solution is obtained through a projection of the high-fidelity problem on a subspace of small dimension; that subspace is generated by a set of base functions which are typically global and strongly dependent from the examined problem, rather than in a space generated by a much larger number of base functions (which can be local, when the problem is treated with finite element, and global, when treated with spectral methods). According to this, it is clear that RB methods belong to the second category of reduced order model techniques since they reduce the degrees of freedom related to the high-fidelity problem leaving unchanged the physical model.

In this section it will be used the strong form (1.1) of the differential problem in order to underline the essential components of an RB method; it is important to remark that the RB method shown here can be built starting from any numerical discretization technique, and not necessarily from the ones based on the weak form of the problem.

The aim of a RB method for PDE is to evaluate, in a very efficiently computational way, an approximation of small dimension of the solution of the PDE. The most common techniques to build a reduced bases space in the case of parametrized PDE, as the *proper orthogonal decomposition* (POD) or the *greedy* algorithm, allow therefore to determine the reduced solution through a *projection* on an opportune subspace of small dimension. The main features of a reduced model can be summarized as follows:

- *high-fidelity discretization technique*: as previously observed, the aim is not to replace an high-fidelity discretization technique with a reduced model.
  In the case of problem (1.1), the approximation high-fidelity can be expressed in the following compact way: given $\boldsymbol{\mu}$, evaluate $s_h(\boldsymbol{\mu}) = f(u_h(\boldsymbol{\mu}))$ where $u_h(\boldsymbol{\mu}) \in V^{N_h}$ is such that
  $$L_h(\boldsymbol{\mu})u_h(\boldsymbol{\mu}) = F_h(\boldsymbol{\mu}); \tag{1.11}$$
  $V^{N_h} \subset V$ stays for a finite dimensional space with a big dimension N_h, $L_h(\boldsymbol{\mu})$ an opportune discrete operator and $F_h(\boldsymbol{\mu})$ a given known term. Instead, the weak formulation of the problem is based on the Galerkin approximation of (1.2): find $u_h(\boldsymbol{\mu}) \in V^{N_h}$ such that
  $$a(u_h(\boldsymbol{\mu}), v_h; \boldsymbol{\mu}) = f(v_h; \boldsymbol{\mu}) \quad \forall v_h \in V^{N_h}. \tag{1.12}$$

- *Galerkin Projection*: an RB method is based on the selection of a reduced basis, obtained from a set of solutions of the high-fidelity problem $\{u_h(\boldsymbol{\mu}^{\boldsymbol{i}})\}_{i=1}^{N}$, the so called *snapshots*, and on the evaluation of a reduced approximation $u_N(\boldsymbol{\mu})$ expressed through a linear combination of such base functions, whose coefficients are calculated thanks to a projection on the space RB
  $$V_N = \mathrm{span}\{u_h(\boldsymbol{\mu}^i), i = 1, ..., N\}, \tag{1.13}$$
  where $N = \dim(V_N) \ll N_h$. Therefore the reduced problem can be expressed as follow: given $\boldsymbol{\mu} \in \mathcal{D}$, evaluate $s_N(\boldsymbol{\mu}) = f(u_N(\boldsymbol{\mu}))$, where $u_N(\boldsymbol{\mu}) \in V_N$ solve
  $$a(u_N(\boldsymbol{\mu}), v_N; \boldsymbol{\mu}) = f(v_N; \boldsymbol{\mu}) \quad \forall v_N \in V_N. \tag{1.14}$$
  Lower is the value of the dimension N, more efficient, in term of computational speed, is the solution of the reduced problem. Note that the RB solution and the RB variable of interest are only an approximation, for a given N_h, of the high fidelity solution $u_h(\boldsymbol{\mu})$ and of the output $s_h(\boldsymbol{\mu})$ (indeed, only indirectly, of $u(\boldsymbol{\mu})$ and $s(\mu)$).
  Naturally, problem (1.14) can be also interpreted in an operatorial form as
  $$L_N(\boldsymbol{\mu})u_N(\boldsymbol{\mu}) = F_N(\boldsymbol{\mu}). \tag{1.15}$$

- *Offline/Online procedure*: under opportune assumption, the generation of the *database* of snapshots can be done Offline only once, and can be completely separated from every new request of input-output evaluation for a new instance of $\boldsymbol{\mu}$, which is done in the Online phase. Obviously the aim of the Online phase is to solve the reduced problem for values of $\boldsymbol{\mu} \in \mathcal{D}$ not selected during the Offline phase. Note that the computational effort of the Offline phase has to be such that it is well compensated by the reduction of the problem to evaluate, so that the entire procedure is efficient from a computational point of view. The level of efficiency reached strongly depends from the examined problem.

- *Error estimation*: it is possible to relate to a RB method with a posteriori error estimation $\Delta_N(\boldsymbol{\mu})$, accurate and not expensive to evaluate, in such way that

  $$\|u_h(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})\|_V \leq \Delta_N(\boldsymbol{\mu}) \quad \forall \boldsymbol{\mu} \in \mathcal{D}, \ \ , N = 1, ..., N_{max}; \tag{1.16}$$

  and in the same way it is possible to derive expressions for the estimator $\Delta_N^s(\boldsymbol{\mu})$ of the error on the variable od interest, such that

  $$|s_h(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})| \leq \Delta_N^s(\boldsymbol{\mu}).$$

  These estimators can be used not only to verify the accuracy of the approximation RB, but also to sample the parametric space in an opportune way during the phase of the construction of the reduced base.

1.3 The reduced basis method

In this section there are more details about the generation of the reduced problem, while, in Section 1.5 are described the main strategies used for the construction of the reduced basis space.
A reduced basis (RB) approximation is typically obtained through a Galerkin projection (or Petrov-Galerkin if the solutions space and the test functions space are different) on an N-dimensional space V_N which has to approximate the manifold

$$\mathcal{M}_h = u_h(\boldsymbol{\mu}) \in V^{N_h} : \boldsymbol{\mu} \in \mathcal{D} \tag{1.17}$$

made by the set of the high-fidelity solutions for every value of the input parameters in the parametric domain $\mathcal{D}$. If the manifold has a small dimension and is enough regular, it is reasonable to assume that each of its points, which correspond to a solution $u_h(\boldsymbol{\mu})$ for a value of $\boldsymbol{\mu} \in \mathcal{D}$, is well approximated by a linear combination of a relative small number of snapshots. Given an optimal way to select the snapshots, which will be discuss later, it has to be ensured that is possible:

1. to find a good combination of the selected snapshots to obtain the RB solution;
2. to represent the RB space through an opportune base;
3. to evaluate the coefficients of the development on the reduced basis in a extremely efficient way.

The way to overcome each of these problems is shown in the following sections.

1.3.1 Reduced bases spaces

The most popular way to build a reduced space is the one concerning the use of solutions "snapshots" of the high-fidelity problem, the spaces generated in this way are sometimes called *RB Lagrangian spaces*. References for different kind of approaches can be find in [17] and [13].
Given a positive integer number N_{max}, it is possible to define a succession of RB spaces V_N^{RB}, generic or one contained into the other, with $1 \leq N \leq N_{max}$, such that each V_N^{RB} is an N-dimensional subspace of V^{N_h}, which means

$$V_1^{RB} \subset V_2^{RB} \subset ...V_{N_{max}}^{RB} \subset V^{N_h}. \tag{1.18}$$

To define such succession it is introduced, for a given $N \in 1, ..., N_{max}$, a sample

$$S_N = \{\boldsymbol{\mu}^{\mathbf{1}}, ..., \boldsymbol{\mu}^N\} \tag{1.19}$$

of elements $\boldsymbol{\mu}^n \in \mathcal{D}$, with $1 \leq n \leq N$, which can be selected opportunely (see Section 1.5). For each of these elements is associated the *snapshot* $u_h(\boldsymbol{\mu}^n) \in V^{N_h}$. Therefore the corresponding RB spaces are given by

$$V_N^{RB} = \text{span}\{u_h(\boldsymbol{\mu}^{\boldsymbol{n}}),\ 1 \leq n \leq N\}. \tag{1.20}$$

Note that, for construction, the spaces V_N^{RB} satisfy (1.18) and the sets of samples (1.19) are one inside the other $S_1 = \{\boldsymbol{\mu}^1\} \subset S_2 = \{\boldsymbol{\mu}^1, \boldsymbol{\mu}^2\} \subset \cdots \subset S_{N_{max}}$.

1.3.2 Galerkin projection

The reduced problem is now built thanks to a Galerkin projection: given $\boldsymbol{\mu} \in \mathcal{D}$, find $u_N(\boldsymbol{\mu}) \in V_N^{RB} \subset V^{N_h}$ such that

$$a(u_N(\boldsymbol{\mu}), v_N; \boldsymbol{\mu}) = f(v_N; \boldsymbol{\mu}) \quad \forall v_N \in V_N^{RB} \tag{1.21}$$

and, eventually, evaluate the variable of interest $s_N(\mu) = J(u_N(\boldsymbol{\mu}))$. The problem (1.21) is the Galerkin-reduced basis (G-RB) approximation of problem (1.2). Taking (1.12) and subtracting (1.21), one get the property

$$a(u_h(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu}), v_N; \boldsymbol{\mu}) = 0 \quad \forall v_N \in V_N^{RB}, \tag{1.22}$$

which is the Galerkin orthogonality for the reduced problem (see Chapter 4 of [19]).

Let's consider now the discrete equations associated to the Galerkin approximation (1.21). First of all it is important to choose carefully the base of the reduced space; indeed a bad choice of the base can lead, even for small dimension N, to system strongly ill-conditioned, since the snapshots of (1.20) become more and more collinear with the increasing of N in the case in which the space V_N^{RB} rises errors which goes rapidly to zero. To avoid an ill-conditioned reduced system it is generally applied the orthonormalization technique of Gram-Schmidt to the set of snapshots $u_h(\boldsymbol{\mu}^n)$, $1 \leq n \leq N_{max}$ and respect the scalar product $(\cdot,\cdot)_V$, in such way to obtain orthonormal base functions $\zeta_n, 1 \leq n \leq N_{max}$, such that $(\zeta_n, \zeta_m)_V = \delta_{nm}$, $1 \leq n, m \leq N_{max}$, where δ_{nm} is the Kronecker delta.

The set $\{\zeta_n\}_{n=1,...,N}$ is chosen as the base of the space V_N^{RB}, for each $1 \leq N \leq N_{max}$. The base functions chosen in this way are not snapshots of the high-fidelity problem, but they generate the same space:

$$V_N = \text{span}\{\zeta_1, ..., \zeta_n\} = \text{span}\{u_h(\boldsymbol{\mu}^1), ..., u_h(\boldsymbol{\mu}^N)\}. \tag{1.23}$$

Substituting the expression

$$u_N(\boldsymbol{\mu}) = \sum_{m=1}^{N} u_N^{(m)}(\boldsymbol{\mu})\zeta_m \tag{1.24}$$

in equation (1.21) and choosing as test function $v_N = \zeta_n$, the following algebraic system is obtained

$$\sum_{m=1}^{N} a(\zeta_m, \zeta_n; \boldsymbol{\mu}) u_N^{(m)}(\boldsymbol{\mu}) = f(\zeta_n; \boldsymbol{\mu}), \quad 1 \leq n \leq N, \tag{1.25}$$

whose unknows are the RB coefficients $u_N^{(m)}(\boldsymbol{\mu}),\ 1 \leq m, n \leq N$.

1.3.3 Offline-Online procedure

The system (1.25) is usually a very low dimension one, however it involves quantities related to the high-fidelity N_h-dimensional space, as the base functions ζ_n, $1 \leq b \leq N$. Using these quantities to assemble the stiffness RB matrix for each value of $\boldsymbol{\mu}$ leaves the cost of the single evaluation input-output $\boldsymbol{\mu} \to s_N(\boldsymbol{\mu})$ too high. The problem is overcome thanks to the previous assumption of *parametric affine dependence*. For the sake of simplicity, f is considered not to depend on the parameter $\boldsymbol{\mu}$.
Thanks to (1.6), the system (1.25) can be expressed as follow,

$$\left(\sum_{q=1}^{Q_a} \Theta_a^q(\boldsymbol{\mu}) \mathbb{A}_N^q\right) \mathbf{u}_N(\boldsymbol{\mu}) = \mathbf{f}_N, \tag{1.26}$$

where $(\mathbf{u}_N(\boldsymbol{\mu}))_m = u_N^{(m)}(\boldsymbol{\mu})$, $(\mathbb{A}_N^q)_{mn} = a^q(\zeta_n, \zeta_m)$, $(\mathbf{f}_N)_n = f(\zeta_n)$, for $1 \leq m, n \leq N$. Therefore the computation requires an expensive Offline phase, $\boldsymbol{\mu}$-independent, to execute only once, and an Online phase extremely efficient, to run for each selected value of $\boldsymbol{\mu} \in \mathcal{D}$:

- in the Offline phase, first, the snapshots $u_h(\boldsymbol{\mu}^n)$ are computed, then the base functions ζ_n through Gram-Schmidt orthonormalization, $1 \leq n \leq N_{max}$; after this there is the assembly and memorization of the structures

  $$f(\zeta_n), \quad 1 \leq n \leq N_{max}, \tag{1.27}$$

  $$a^q(\zeta_n, \zeta_m), \quad 1 \leq n, m \leq N_{max},\ 1 \leq q \leq Q_a. \tag{1.28}$$

  Therefore the cost of the Offline operations depends on N_{max}, Q_a, and N_h;

- in the Online phase, the structure defined in (1.28) are used to form

  $$\sum_{q=1}^{Q_a} \Theta_a^q(\boldsymbol{\mu}) a^q(\zeta_n, \zeta_m), \quad 1 \leq n, m \leq N; \tag{1.29}$$

  and the resulting linear system $N \times N$ (1.26) is solved to compute the weights $u_N^{(m)}(\boldsymbol{\mu})$, $1 \leq m \leq N$. Therefore, the cost of the Online operations is $O(Q_a N^2)$ concerning the evaluation of the sum (1.29); $O(N^3)$ for the solution of the system (1.26), note that the RB matrix obtained are full. The *storage* cost of the structures, which are necessary during the Online phase is $O(Q_a N_{max}^2)+O(N_{max})$ operations, thanks to condition (1.18): for every given value of of N, it is possible to extract the RB matrix of dimensions $N \times N$ as the main under-matrix of the correspondent matrix of dimensions $N_{max} \times N_{max}$.

The Online cost to evaluate $u_N(\boldsymbol{\mu})$ (and, it can be easily seen, also to evaluate the variable of interest $s_N(\boldsymbol{\mu})$) results independent form N_h, implying two consequences: first, if the dimension N is small, the output will be very fast; second, N_h can be chosen relatively big to make the error $\|u(\boldsymbol{\mu}) - u_h(\boldsymbol{\mu})\|_V$ enough small, without influencing the Online cost.

1.4 Algebraic and geometric interpretation of the RB problem

To better understand the RB method is useful to clarify which relationships exist between the Galerkin-reduced basis (G-RB) approximation and the Galerkin-high-fidelity (1.12) approximation from both an algebraic and geometric point of view. More information can be found in [20].
Let's indicate with $\mathbf{u}_h(\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$ and $\mathbf{u}_N(\boldsymbol{\mu}) \in \mathbb{R}^{N}$ the vectors of the degrees of freedom associated to the functions $u_h(\boldsymbol{\mu}) \in V^{N_h}$ and $u_N(\boldsymbol{\mu}) \in V_N^{RB}$, respectively, which are given by

$$\mathbf{u}_h(\boldsymbol{\mu}) = (u_h^{(1)}(\boldsymbol{\mu}), ..., u_h^{N_h}(\boldsymbol{\mu}))^T, \qquad \mathbf{u}_N(\boldsymbol{\mu}) = (u_N^{(1)}(\boldsymbol{\mu}), ..., u_N^{N_h}(\boldsymbol{\mu}))^T.$$

With $\{\tilde{\varphi}^r\}_{r=1}^{N_h}$ the Lagrangian nodal base of V^{N_h} is indicated, therefore $\tilde{\varphi}^r(\mathbf{x}_s) = \delta_{rs}$, and $\{w_r\}_{r=1}^{N_h}$ indicates a set of weights such that $\sum_{r=1}^{N_h} w_r = |\Omega|$ and $\{\mathbf{x}_r\}_{r=1}^{N_h}$ represents the set of discretization nodes, $r, s = 1, ..., N_h$[1]. Such base results to be orthogonal in relation to the discrete scalar product

$$(u_h, v_h)_h = \sum_{r=1}^{N_h} w_r u_h(\mathbf{x}_r) v_h(\mathbf{x}_r).$$

---

[1] Note that the set of discretization nodes is usually not equal but proportional to N_h since it depends from how many quadrature nodes are considered in each single element of the mesh.

It is useful to normalize the base functions defining

$$\varphi^r = \frac{1}{\sqrt{w_r}}\tilde{\varphi}^r, \quad (\varphi^r, \varphi^s)_h = \delta_{rs}, \qquad r, s = 1, ..., N_h. \tag{1.30}$$

Thanks to the orthonormality of the base functions, the following relations $v_h^{(r)} = (v_h, \varphi^r)_h$, for $r = 1, ..., N_h$, hold.

1.4.1 Algebraic interpretation of the problem

First, it is highlighted the algebraic connection between the problem (G-RB) (1.21) and Galerkin-high-fidelity (1.12), which has important consequences on the computational aspects connected with a RB method.
In matrix form, the problem (G-RB) (1.25) can be written as

$$\mathbb{A}_N(\boldsymbol{\mu})\mathbf{u}_N(\boldsymbol{\mu}) = \mathbf{f}_N, \tag{1.31}$$

where $\mathbf{f}_N = (f_N^{(1)}, ..., f_N^{(N)})^T$, $f_N^{(k)} = f(\zeta_k)$, $(\mathbb{A}_N(\boldsymbol{\mu}))_{km} = a(\zeta_m, \zeta_k; \boldsymbol{\mu})$, with $k, m = 1, ..., N$. On the other side, the Galerkin-high-fidelity problem (1.12) in matrix form is given by

$$\mathbb{A}_h(\boldsymbol{\mu})\mathbf{u}_h(\boldsymbol{\mu}) = \mathbf{f}_h, \tag{1.32}$$

with $\mathbf{f}_h = (f_h^{(1)}, ..., f_h^{(N)})^T$, being $f_h^{(r)} = f(\varphi_r)$ if the integral is exactly evaluated, or $f_h^{(r)} = (f, \varphi^r)_h$ if the integral is computed through a quadrature formula, while $(\mathbb{A}_h(\boldsymbol{\mu}))_{rs} = a(\varphi^s, \varphi^r; \boldsymbol{\mu})$, for $r, s = 1, ..., N_h$. For simplicity, the dependence on $\boldsymbol{\mu}$ can be avoid by now.
Let's define $\mathbb{V} \in \mathbb{R}^{N_h \times N}$ the *transformation matrix*, whose components are given by

$$(\mathbb{V})_{rk} = (\zeta_k, \varphi^r)_h, \quad r = 1, ..., N_h \ , \ k = 1, ..., N. \tag{1.33}$$

Thanks to this definition, it is possible to prove these identities:

$$\mathbf{f}_N = \mathbb{V}^T\mathbf{f}_h, \qquad \mathbb{A}_N = \mathbb{V}^T\mathbb{A}_h\mathbb{V}, \tag{1.34}$$

indeed

$$\begin{aligned}(\mathbb{V}^T\mathbb{A}_h\mathbb{V})_{km} =& \sum_{r,s=1}^{N_h}(\mathbb{V})_{kr}^T(\mathbb{A}_h)_{rs}(\mathbb{V})_{sm} = \sum_{r,s=1}^{N_h}(\zeta_k, \varphi^r)_h a(\varphi^s, \varphi^r)(\zeta_m, \varphi^s)_h \\ =& a\left(\sum_{s=1}^{N_h}(\zeta_m, \varphi^s)_h\varphi^s, \sum_{r=1}^{N_h}(\zeta_k, \varphi^r)_h\varphi^r\right) = a(\zeta_m, \zeta_k) = (\mathbb{A}_N)_{km}\end{aligned}$$

and, in the same way,

$$\begin{aligned}
(\mathbb{V}^T\mathbf{f}_h)^{(k)} =& \sum_{r=1}^{N_h}(\mathbb{V})_{kr}^T(\mathbf{f}_h)^{(r)} = \sum_{r=1}^{N_h}(\zeta_k,\varphi^r)_h f(\varphi^r) \\
=& f\left(\sum_{r=1}^{N_h}(\zeta_k,\varphi^r)_h\varphi^r\right) = f(\zeta_k) = (\mathbf{f}_N)_k. \qquad (1.35)
\end{aligned}$$

Using (1.34), every matrix coming from the discretization and affine decomposition of the problem, which is independent from $\boldsymbol{\mu}$ given by $\mathbb{A}_N^q$, can be assembled only once in the Offline phase after the high-fidelity matrix $\mathbb{A}_h^q$ has been evaluated.
The vectorial representation of the error between the solution of the problem (G-RB) and the Galerkin-high-fidelity approximation is

$$\mathbf{e}_h = \mathbf{u}_h - \mathbb{V}\mathbf{u}_N. \qquad (1.36)$$

Likewise, the vectorial representation of the residual of the high-fidelity problem, evaluated on the (G-RB) solution, is

$$\mathbf{r}_h(\mathbf{u}_N) = \mathbf{f}_h - \mathbb{A}_h\mathbb{V}\mathbf{u}_N. \qquad (1.37)$$

The following lemma gives the main algebraic connection between the (G-RB) problem and the Galerkin-high-fidelity approximation:

Lemma 1.1 *The following algebraic relations hold:*

$$\mathbb{A}_h\mathbf{e}_h = \mathbf{r}_h(\mathbf{u}_N), \qquad (1.38)$$

$$\mathbb{V}^T\mathbb{A}_h\mathbf{u}_h = \mathbf{f}_N, \qquad (1.39)$$

$$\mathbb{V}^T\mathbf{r}_h(\mathbf{u}_N) = \mathbf{0}, \qquad (1.40)$$

where $\mathbf{e}_h$ *and* $\mathbf{r}_h(\mathbf{u}_N)$ *are defined by* (1.36) *and* (1.37), *respectively.*

Proof. Equation (1.38) comes directly from (1.36) and (1.32).
Multiplying from the left (1.32) for $\mathbb{V}^T$, (1.39) is obtained, thanks to (1.34). Finally, (1.40) comes from (1.37) using the identities in (1.34) and the problem (1.31). □

1.4.2 Geometric interpretation of the problem (G-RB)

To characterize geometrically the solution $\mathbf{u}_N$ of the problem (G-RB) as well as the error (1.36), it can be used the fact that the base matrix $\mathbb{V}$ defined by (1.33) identifies an orthogonal projection on the subspace $\mathbf{V}_N = \mathrm{span}\{\mathbf{v}_1, ..., \mathbf{v}_N\}$ of $\mathbb{R}^{N_h}$, generated by the columns of $\mathbb{V}$. Note that $\dim(\mathbf{V}_N) = N$ since the columns of $\mathbf{V}$ are linearly independent. With the assumption that the base functions $\{\zeta_k\}_{k=1,...,N}$ are orthonormal in relation to the scalar product $(\cdot,\cdot)_h$, which means

$$(\zeta_k, \zeta_m) = \sum_{j=1}^{N_h} w_j \zeta_k(x_j)\zeta_m(x_j) = \delta_{km}, \tag{1.41}$$

follows that

$$\mathbb{V}^T\mathbb{V} \in \mathbb{R}^{N\times N}, \qquad \mathbb{V}^T\mathbb{V} = \mathbb{I}_N \tag{1.42}$$

where $\mathbb{I}_N$ stays for the identity matrix of dimension N.

Lemma 1.2 *The following statements hold:*

1. *the matrix $\mathbf{\Pi} = \mathbb{V}\mathbb{V}^T \in \mathbb{R}^{N_h\times N_h}$ is a projection matrix from $\mathbb{R}^{N_h}$ on the subspace $\mathbf{V}_N$;*
2. *the matrix $\mathbb{I}_{N_h} - \mathbf{\Pi} = \mathbb{I}_{N_h} - \mathbb{V}\mathbb{V}^T \in \mathbb{R}^{N_h\times N_h}$ is a projection matrix from $\mathbb{R}^{N_h}$ on the subspace $\mathbf{V}_N^{\perp}$, being this last one the subspace of $\mathbb{R}^{N_h}$ orthogonal to $\mathbf{V}_N$;*
3. *the residual $\mathbf{r}_h(\mathbf{u}_N)$ satisfies*

$$\mathbf{\Pi}\mathbf{r}_h(\mathbf{u}_N) = \mathbf{0}, \tag{1.43}$$

which means that $\mathbf{r}_h(\mathbf{u}_N)$ belongs to the orthogonal space $(\mathbf{V}_N)^{\perp}$.

Proof. The first property is a direct consequence of the orthonormal property (1.42): indeed

$$\forall \mathbf{w}_N \in \mathbf{V}_N \;\; \exists\, \mathbf{v}_N \in \mathbb{R}^N \mid \mathbf{w}_N = \mathbb{V}\mathbf{v}_N.$$

Therefore, $\forall \mathbf{v}_h \in \mathbb{R}^{N_h}$, $\forall \mathbf{w}_N \in \mathbf{V}_N$,

$$(\mathbf{\Pi}\mathbf{v}_h, \mathbf{w}_N)_2 = (\mathbf{\Pi}\mathbf{v}_h, \mathbb{V}\mathbf{v}_N)_2 = (\mathbb{V}^T\mathbf{v}_h, \mathbb{V}^T\mathbb{V}\mathbf{v}_N)_2 = (\mathbf{v}_h, \mathbb{V}\mathbf{v}_N)_2 = (\mathbf{v}_h, \mathbf{w}_N)_2.$$

Second statement directly comes from the first, while (1.43) from (1.40). □

Thanks to the supposition of an orthonormal base, the error $\mathbf{e}_h = \mathbf{u}_h - \mathbb{V}\mathbf{u}_N$ can be decomposed in the sum of two orthogonal terms:

$$\begin{aligned} \mathbf{e}_h &= \mathbf{u}_h - \mathbb{V}\mathbf{u}_N = (\mathbf{u}_h - \mathbf{\Pi}\mathbf{u}_h) + (\mathbf{\Pi}\mathbf{u}_h - \mathbb{V}\mathbf{u}_N) \\ &= (\mathbb{I}_{N_h} - \mathbf{\Pi})\mathbf{u}_h + \mathbb{V}(\mathbb{V}^T\mathbf{u}_h - \mathbf{u}_N) = \mathbf{e}_{\mathbf{V}_N^\perp} + \mathbf{e}_{\mathbf{V}_N}. \end{aligned} \tag{1.44}$$

The first term, orthogonal to $\mathbf{V}_N$, takes into account that the high-fidelity solution does not necessary belong to the subspace $\mathbf{V}_N$, while the second one, which stays in $\mathbf{V}_N$, relays to the fact that a different problem from the original one is resolved. Indeed the following result holds

Proposition 1.1 *The high-fidelity representation of* $\mathbf{u}_N \in \mathbb{R}^N$*, defined by* $\tilde{\mathbf{u}}_h(\boldsymbol{\mu}) = \mathbb{V}\mathbf{u}_N(\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$*, solves the "equivalent" high-fidelity problem*

$$\mathbb{V}\mathbb{V}^T\mathbb{A}_h(\boldsymbol{\mu})\mathbb{V}\mathbb{V}^T\tilde{\mathbf{u}}_h = \mathbb{V}\mathbb{V}^T\mathbf{f}_h(\boldsymbol{\mu}). \tag{1.45}$$

The matrix $\mathbb{V}\mathbb{V}^T\mathbb{A}_h(\boldsymbol{\mu})\mathbb{V}\mathbb{V}^T$ *has rank equal to* N *and its pseudo-inverse of Moore-Penrose is given by*

$$(\mathbb{V}\mathbb{V}^T\mathbb{A}_h(\boldsymbol{\mu})\mathbb{V}\mathbb{V}^T)^\dagger = \mathbb{V}\mathbb{A}_N^{-1}(\boldsymbol{\mu})\mathbb{V}^T. \tag{1.46}$$

1.5 Construction of the reduced spaces

In this section, two techniques to sample the parametric space and to evaluate the snapshots necessary to build the reduced base are presented. Both techniques are later used in the problems treated in this master thesis work. The first technique presented is the *greedy* algorithm origninally introduced in [18, 22]. It is based on the idea to select, at each step, the element representing a local optimal in relation to an opportune indicator of the error. The second method is the so called *proper orthogonalized decomposition* (POD). This technique was created with the aim to speed up the solution of time-dependent problems and only later has been extended to parametric problems (in the first applications the time was considered as the only parameter).

1.5.1 Greedy algorithm

From an abstract point of view, a greedy algorithm is a general procedure which allows to approximate each element of a compact set K of an Hilbert

space V through a subspace of K with a given dimension $N \geq 1$, this subspace has to be built choosing opportunely some elements of K. Basically the idea is to search elements $\{x_1, ..., x_N\}$ in K such that every $x \in K$ is well approximated by elements of the subspace $K_N = \text{span}\{x_1, ..., x_N\}$.
In the case of construction of the reduced base $K = \mathcal{M}_h$, where $\mathcal{M}_h$ is the manifold of the solutions defined in (1.17) and the greedy algorithm has the following form:

$$\begin{aligned}
&\boldsymbol{\mu}_1 = \arg\max_{\boldsymbol{\mu}\in\mathcal{D}} \|u_h(\boldsymbol{\mu})\|_V \,; \\
&\text{given } \boldsymbol{\mu}^1, ..., \boldsymbol{\mu}^{N-1}, \\
&\qquad \text{set } V_{N-1} = \text{span}\{u_h(\boldsymbol{\mu}^1), ..., u_h(\boldsymbol{\mu}^{N-1})\}; \\
&\qquad \text{evaluate } \boldsymbol{\mu}^N = \arg\max_{\boldsymbol{\mu}\in\mathcal{D}} d(u_h(\boldsymbol{\mu}), V_{N-1}); \\
&\text{iterate until } \arg\max_{\boldsymbol{\mu}\in\mathcal{D}} d(u_h(\boldsymbol{\mu}), V_{N_{max}}) < \varepsilon^*_{\text{tol}}.
\end{aligned} \tag{1.47}$$

$\varepsilon^*_{\text{tol}}$ is a fixed tolerance, while $d(u_h(\boldsymbol{\mu}), V_{N-1})$ indicates the distance between $u_h(\boldsymbol{\mu})$ and the subspace V_{N-1}; it is given by

$$d(u_h(\boldsymbol{\mu}), V_{N-1}) = \left\| u_h(\boldsymbol{\mu}) - \Pi_{V_{N-1}} u_h(\boldsymbol{\mu}) \right\|_V , \tag{1.48}$$

which means that in the step Nth the selected snapshot $u_h(\boldsymbol{\mu}^N)$ is the element of the manifold which is worst approximated by its orthogonal projection on V_{N-1}. The elements of the set $\{u_h(\boldsymbol{\mu}^1), ..., u_h(\boldsymbol{\mu}^N)\}$, generated thanks to the greedy algorithm, are then orthonormalized in relation to the scalar product $(\cdot, \cdot)_V$, giving therefore an orthonormal base $\{\zeta_1, ..., \zeta_N\}$ of V_N.
However, if it is performed with this descriptions, the greedy algorithm (1.47) would still be impracticable from a computational point of view: indeed, in each step the selection of the optimal snapshot would require the solution of an optimization problem, in which the evaluation of $\arg\max_{\boldsymbol{\mu}\in\mathcal{D}} d(u_h(\boldsymbol{\mu}), V_{N-1})$ requires the evaluation of the high-fidelity solution $u_h(\boldsymbol{\mu})$ for every $\boldsymbol{\mu} \in \mathcal{D}$, leading to an excessive high cost.
In the practice, this cost is strongly reduced substituting the search of a maximum over $\mathcal{D}$ with the maximum over a very large sample $\Xi_{\text{train}} \subset \mathcal{D}$, of cardinality $|\Xi_{\text{train}}| = n_{\text{train}}$, which is necessary to select the reduced space or to *train* the RB approximation. Nevertheless, the solution of many high-fidelity problems (n_{train} problems) is still required.
Another simplification consists on substituting the approximation error with

a posteriori estimator of the error $\Delta_{N-1}(\boldsymbol{\mu})$ such that,

$$\|u_h(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})\|_V \leq \Delta_N(\boldsymbol{\mu}) \quad \forall \boldsymbol{\mu} \in \mathcal{D} \tag{1.49}$$

and which is easy to evaluate. The RB-greedy algorithm has then the following pseudo-code:

```
$S_1 = \boldsymbol{\mu}^1$;
evaluate $u_h(\boldsymbol{\mu}^1)$;
$V_1 = \text{span}\{u_h(\boldsymbol{\mu}^1)\}$;
for $N = 2, ...$
    $\boldsymbol{\mu}^N = \arg\max_{\boldsymbol{\mu} \in \Xi_{\text{train}}} \Delta_{N-1}(\boldsymbol{\mu})$;
    $\varepsilon_{N-1} = \Delta_{N-1}(\boldsymbol{\mu}^N)$;
    if $\varepsilon_{N-1} \leq \varepsilon^*_{\text{tol}}$
        $N_{max} = N - 1$;
    end;
    evaluate $u_h(\boldsymbol{\mu}^N)$;
    $S_N = S_{N-1} \cup \{\boldsymbol{\mu}^N\}$;
    $V_N = V_{N-1} \cup \text{span}\{u_h(\boldsymbol{\mu}^N)\}$;
end.
```

Basically at the Nth iteration of the algorithm, among all the possible candidates $u_h(\boldsymbol{\mu})$, $\boldsymbol{\mu} \in \Xi_{\text{train}}$, the element, whose posteriori estimation (1.49) indicates to be the worst approximated by the solution of the RB problem associated to the space V_{N-1}, is added to the set of the already selected snapshots.

1.5.2 Proper Orthogonal Decomposition (POD)

An alternative technique to the greedy algorithm for the construction of the reduced spaces in the case of parametrized problems is the *proper orthogonal decomposition* (POD). This technique reduces the dimensionality of a system transforming the starting variables into a new set of variables unrelated between them (called *modes* POD, or principal component), such that the first modes describe well a good portion of the energy carried by the original variables.

From an algebraical point of view, the POD technique is based on the use of the singular values decomposition (SVD), whose modality of application is going to be clarified now.
Considering a set of n_{train} vectors *snapshots* $\{\mathbf{u}_1, ..., \mathbf{u}_{n_{\text{train}}}\}$ belonging to $\mathbb{R}^{N_h}$, the snapshots matrix $\mathbb{U} \in \mathbb{R}^{N_h \times n_{\text{train}}}$

$$\mathbb{U} = [\mathbf{u}_1 \;\; \mathbf{u}_2 \;\; ... \;\; \mathbf{u}_{n_{\text{train}}}],$$

with $n_{\text{train}} = |\Xi_{\text{train}}| \ll N_h$, can be formed. Clearly it is true that

$$\mathbf{u}_j = (u_j^{(1)}, ..., u_j^{(N_h)}) \in \mathbb{R}^{N_h}, \;\; u_j^{(r)} = u_h(x_r; \boldsymbol{\mu}^j) \Leftrightarrow u_h(x; \boldsymbol{\mu}^j) = \sum_{r=1}^{N_h} u_j^{(r)} \varphi^r(\mathbf{x}). \tag{1.50}$$

The decomposition in singular values of $\mathbb{U}$ is given by

$$\mathbb{V}^T \mathbb{U} \mathbb{Z} = \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix},$$

where $\mathbb{V} = [\boldsymbol{\zeta}_1 \;\; \boldsymbol{\zeta}_2 \;\; ... \;\; \boldsymbol{\zeta}_{N_h}] \in \mathbb{R}^{N_h \times N_h}$ and $\mathbb{Z} = [\boldsymbol{\psi}_1 \;\; \boldsymbol{\psi}_2 \;\; ... \;\; \boldsymbol{\psi}_{n_{\text{train}}}] \in \mathbb{R}^{n_{\text{train}} \times n_{\text{train}}}$ are orthogonal matrices and $\Sigma = \text{diag}(\sigma_1, ..., \sigma_r)$, being $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r$. The integer $r \leq n_{\text{train}}$ indicates the rank of $\mathbb{U}$, which results strictly less than n_{train} if the snapshots are linearly dependent. It can be written

$$\mathbb{U}\boldsymbol{\psi}_i = \sigma_i \boldsymbol{\zeta}_i \;\text{ and }\; \mathbb{U}^T \boldsymbol{\zeta}_i = \sigma_i \boldsymbol{\psi}_i, \quad i = 1, ..., r$$

or, equivalently,

$$\mathbb{U}^T \mathbb{U} \boldsymbol{\psi}_i = \sigma_i^2 \boldsymbol{\psi}_i \;\text{ and }\; \mathbb{U}\mathbb{U}^T \boldsymbol{\zeta}_i = \sigma_1^2 \boldsymbol{\zeta}_i, \quad i = 1, ..., r; \tag{1.51}$$

therefore $\sigma_i^2, \; i = 1, ..., r$ are the not zero eigenvalues of the *correlation matrix* $\mathbb{C} = \mathbb{U}^T \mathbb{U}$,

$$\mathbb{C}_{ij} = \mathbf{u}_i^T \mathbf{u}_j, \quad 1 \leq i, j \leq n_{\text{train}},$$

listed in ascending order. For every $N \leq n_{\text{train}}$, the POD base of dimension N is defined as the set of the first N left singular vector $\boldsymbol{\zeta}_1, ..., \boldsymbol{\zeta}_N$ of $\mathbb{U}$ or, alternatively, as the set of vectors

$$\boldsymbol{\zeta}_j = \frac{1}{\sigma_j} \mathbb{U} \boldsymbol{\psi}_j, \quad 1 \leq j \leq N \tag{1.52}$$

obtained from the first N eigenvectors $\boldsymbol{\psi}_1, ..., \boldsymbol{\psi}_N$ of the correlation matrix $\mathbb{C}$.

For construction, the POD base is orthonormal. Furthermore, if $\{\mathbf{z}_1, ..., \mathbf{z}_N\}$ indicates an arbitrary set of N orthonormal vectors in $\mathbb{R}^{N_h}$ and $\Pi_{Z_N}\mathbf{w}$ is the projection of the vector $\mathbf{w} \in \mathbb{R}^{N_h}$ on $Z_N = \text{span}\{\mathbf{z}_1, ..., \mathbf{z}_N\}$, which means

$$\Pi_{Z_N}\mathbf{u} = \sum_{n=1}^{N} \pi^n_{Z_N}(\mathbf{u})\mathbf{z}_n, \quad \text{with} \quad \pi^n_{Z_N}(\mathbf{u}) = \mathbf{u}^T\mathbf{z}_n,$$

the POD base (1.52) generated by the set of snapshots $\mathbf{u}_1, ..., \mathbf{u}_{n_{\text{train}}}$ can be characterized as the solution of the following minimum problem:

$$\begin{cases} \min\{E(\mathbf{z}_1, ..., \mathbf{z}_N), \; \mathbf{z}_i \in \mathbb{R}^{N_h}, \; \mathbf{z}_i^T\mathbf{z}_j = \delta_{ij} \; \forall 1 \le i, j \le N\} \\ \\ \qquad \text{with } E(\mathbf{z}_1, ..., \mathbf{z}_N) = \sum_{i=1}^{n_{\text{train}}} \|\mathbf{u}_i - \Pi_{Z_N\mathbf{u}_i}\|_2^2 \,. \end{cases} \tag{1.53}$$

Basically the POD base minimize, among all the possible sets of N orthonormal vectors $\{\mathbf{z}_1, ..., \mathbf{z}_N\}$ in $\mathbb{R}^{N_h}$, the sum of the square of the errors $E(\mathbf{z}_1, ..., \mathbf{z}_N)$ between every snapshots $\mathbf{u}_i$ and its projection $\Pi_{Z_N}\mathbf{u}_i$ on the subspace Z_N. The quantity $E(\mathbf{z}_1, ..., \mathbf{z}_N)$ is often called as POD *energy*.

The POD technique construction just presented is based on the so called *snapshots method*, introduced by Sirovich [27]. It is also possible to prove that

$$E(\boldsymbol{\zeta}_1, ..., \boldsymbol{\zeta}_N) = \sum_{i=N+1}^{r} \sigma_i^2; \tag{1.54}$$

which expresses the fact that the error made by the POD base of dimension N in the approximation of the set of snapshots is equal to the sum of the square of the singular values corresponding at the $r - N$ modes not selected for the construction of the base. Therefore, it is possible to define N_{max} in such a way that $E(\boldsymbol{\zeta}_1, ..., \boldsymbol{\zeta}_N) \le \epsilon^*_{\text{tol}}$, where ϵ^*_{tol} is a given tolerance. To do this, is enough to choose N_{max} as the smallest value of N such that

$$I(N) = \sum_{i=1}^{N} \sigma_i^2 \Big/ \sum_{i=1}^{r} \sigma_i^2 \ge 1 - \delta, \tag{1.55}$$

which means that the energy carried by the last $r - N_{\text{max}}$ modes is equal to $\delta > 0$, where δ can be chosen as small as wanted. The key feature of this procedure is that, even if δ is usually a very small value (typically $\delta = 10^{-\beta}$ with $\beta = 3, 4, ...,$), in many cases N_{max} is relatively small (very lower than

r).

To conclude note that, in a lot of applications, the POD approach for the construction of the reduced space can result more expensive, from a computational point of view, than the greedy algorithm. Indeed, this last one requires only to evaluate N, typically few, solutions of the high-fidelity problem, while the POD requires the evaluation of n_{train}, possibly many, solutions high-fidelity to determine the snapshots matrix, in addition to the solution of an eigenvalue problem for the correlation matrix $\mathbb{C} \in \mathbb{R}^{N_h \times N_h}$. Nevertheless, the POD results a more general technique, it is applicable also to those problems where is not possible to derive a posteriori estimation of the error that is essential for the effectiveness of a greedy algorithm. Moreover, (1.55) provides an useful information about the content of energy neglected by the selected POD modes to build the reduced base, which is an indication, in L^2-norm rather than V-norm, of the approximation error.

Chapter 2

Non-affine model order reduction via hyper-reduction

In the previous chapter the focus has been pointed on the reduced basis method, which is the most popular approach for Galerkin projection based Model Order Reduction. In order to take advantage of the speeding in computational time due to the projection of our problem on a subspace of dimension N strongly lower compared the one N_h of the high-fidelity problem, a fundamental hypothesis has been made: the parametric dependence of the bilinear form and of the linear form, coming from the weak formulation, has to be affine in relation to $\boldsymbol{\mu}$, allowing to write them as in (1.6) and (1.7). This is a very restrictive hypothesis. Indeed, if we want to build models that are truly representative of the physical phenomena, it is necessary, most of times, to introduce a dependence of the parameters of our models from some quantities related to the variables of the problem.
For example, in (1.8) we could have that the diffusivity coefficients μ_1 and μ_2 are directly dependent from u and in this case it is clear that the affine decomposition (1.6) is not possible since we cannot extract any more the dependence from the parameters of the bilinear form $a(w, v; \boldsymbol{\mu})$ and put it in the scalar functions $\Theta_a^q(\boldsymbol{\mu})$.
What we loose without the assumption of affine dependence of the equations from the parameters is the possibility to pre-compute Offline the reduced operators $\mathbb{A}_N^q$, and therefore to write our problem in the form (1.26) for which the solution has a computational cost which depends on N instead of N_h as the original discretized equations.
To fix this problem, the most popular techniques involve the use of a second

reduction stage. As the first reduction stage works directly on the high-fidelity approximation simplifying its computation, the second reduction stage works on the first reduction stage approximation, speeding its evaluation always trough an opportune "simplification" of the reduced problem. Since the second reduction stage is a simplification of an already simplified discretization, the error it introduces in relation to the high fidelity approximation, directly sums with the one of the first reduction stage.
Compared to the first reduction stage, for which there are methods (e.g. reduced basis) whose use is considered standard since they works well for large part of the problems, the second stage of dimensionality reduction, for which [24] coined the term *hyper-reduction*, is far more challenging and is nowadays a topic of discussion in the model reduction community. In the following, is reported the classification of hyper-reduction methods made in [10], we prefer to write directly here the classification instead to only remind to the quotation, because having in mind which are the different techniques developed to treat the second stage of reduction is a crucial passage to fully understand the contribution of this work.

2.1 Classification of "hyper-reduction" methods

Let $\mathbf{F}^h \in \mathbb{R}^{N_h}$ denote the full-order term carrying a general, nonaffine relationship with both the input variable and the state variable, e.g. something coming from a finite element discretization like

$$\mathbf{F}^h = \mathbf{A}^h(\boldsymbol{\mu}(\mathbf{u}))\mathbf{u},$$

with $\mathbf{A}^h(\boldsymbol{\mu}(\mathbf{u})) \in \mathbb{R}^{N_h \times N_h}$ and $\mathbf{u} \in \mathbb{R}^{N_h}$. The corresponding projection onto the reduced order space will be represented by $\mathbf{F} \in \mathbb{R}^m$ $(m \ll N_h)$, the connection between these two variables being the matrix of basis vector $\mathbf{\Phi} \in \mathbb{R}^{N_h \times m}$ $(\mathbf{F} = \mathbf{\Phi}^T \mathbf{F}^h)$. The approaches to treat with the approximation of $\mathbf{F}^h$ can be broadly classified as *nodal vector approaches* and *integral approaches.*

2.1.1 Nodal vector approximation approaches

In this type of approaches, the approximation is given by replacing the finite element vector $\mathbf{F}^h$ by a low-dimensional interpolant $\mathbf{F}^h \approx \mathbf{R}_\mathbf{F} \mathbf{F}^h_\mathbf{z}$, with $\mathbf{R}_\mathbf{F} \in$

$\mathbb{R}^{N_h \times m}$ interpolation matrix, and $\mathbf{F}^h_{\mathbf{z}}$ the entries of $\mathbf{F}^h$ corresponding to the degrees of freedom ($\mathbf{z} \subset \{1, 2, ..., N_h\}$) at which the interpolation takes place. The interpolation matrix is obtained following the procedure of computing a basis matrix for $\mathbf{F}^h$, and then determining, through an offline phase, a set of indices so that the error is minimized over a set of representative snapshots of $\mathbf{F}^h$. Some of the most famous methods which belong to this category are: the Empirical Interpolation Method (EIM) [4], the Discrete Empirical Interpolation Method (DEIM) [7], the Best Point Interpolation Method (BPIM) [15] and the Missing Point Estimation Method [3].

2.1.2 Integral approximation approaches

In a finite element context, $\mathbf{F}$ can be regarded, not only as a projection of a large vector into a reduced-order space ($\mathbf{F} = \mathbf{\Phi}^T \mathbf{F}^h$), but also as the result of integrating over the concerned domain $\Omega \in \mathbb{R}^d$, with $d = 2$ or 3, the corresponding reduced-order variable $\mathbf{f} = \mathbf{\Phi}^T \mathbf{f}^h$ ($\mathbf{f}^h : \Omega \to \mathbb{R}^{N_h}$), i.e.:

$$\mathbf{F} = \mathbf{\Phi}^T \int_\Omega \mathbf{f}^h \, \mathrm{d}\Omega = \int_\Omega \mathbf{f} \, \mathrm{d}\Omega. \tag{2.1}$$

Knowing this, the problem can be now viewed as the approximation of an integral, rather than the approximation of a vector. There are two possible ways to treat integral approximation: (1) looking for a low-dimensional approximation of the integrand or (2) approximating the integral itself as a weighted sum of the integrand evaluated at optimal sampling points.

1. *Interpolation of the integrand.* These methods follow, basically, the same procedure described for vector approximation approaches with the important difference that now the interpolant is not built for the integral but directly for the *integrand*; which means that if we make $\mathbf{f}(\mathbf{x}) \approx \sum_{g \in \mathbf{z}} \mathbf{R}_g(\mathbf{x}) \mathbf{f}(\mathbf{x}_g)$, where $\mathbf{R}_g$ ($g \in \mathbf{z}$) stands for the interpolation functions, then we can write

$$\mathbf{F} = \int_\Omega \mathbf{f} \mathrm{d}\Omega \approx \sum_{g \in \mathbf{z}} \overbrace{\left(\int_\Omega \mathbf{R}_g \mathrm{d}\Omega \right)}^{\mathbf{Q}_g} \mathbf{f}(\mathbf{x}_g) = \sum_{g \in \mathbf{z}} \mathbf{Q}_g \mathbf{f}(\mathbf{x}_g). \tag{2.2}$$

   Hence, the integral can be approximated as a sum of the product of matrix weights $\mathbf{Q}_g$ (which can be computed Offline once the interpolation base has been decided) and the integrand evaluated at the interpolating points $\mathbf{x}_g \in \mathbf{z}$ which have to be chosen among the Gauss points of the underlying finite element mesh. Example of this approach is presented in [9] or [11].

2. *Cubature methods.* The last approach to be discussed relays to the *cubature methods.* The integral is approximated as a finite sum of *positive scalar* weights $\{w_g\}_{g=1}^{m}$ times the integrand evaluated at appropriately chosen sampling points:

$$\mathbf{F} \approx \sum_{g=1}^{m} w_g \mathbf{f}(\bar{\mathbf{x}}_g). \tag{2.3}$$

The first scheme of this type was proposed by An. et al. (2009) [26]; this strategy consists on determining, among the integration points of the FE mesh, a reduced set of points and associated *positive* weights so that the integration error is minimized over a set of representative samples of the integrand. The motivation behind constraining the weights to be positive scalars, is that, in doing so, the contribution to the Jacobian matrix due to the nonlinear term, inherits the spectral properties of its full-order counterpart. For example, in a structural problem, if the FE stiffness matrix is symmetric and positive definite, one would wish to have these properties also in reduced-order counterpart. Note that this desirable attribute is not enjoyed by the other two approaches presented previously. Indeed, interpolatory schemes ruin the symmetry and, depending on the location of the sampling points, may also destroy the positive definiteness of finite element stiffness matrices [11, 2]. As a consequence, such schemes tend to be less robust than the finite element models they intend to approximate.

All the cubature methods search for a sparse representation of the vector containing the integration points of the FE mesh and of the associated weights has to be find. The word sparse means that if the number of integration points of the FE mesh is $O(N_h)$, then the number of points m for the reduced quadrature rule has to be such that $m \ll N_h$. This sparsity should be introduced explicitly, for example through a heuristic sequential point selection process [10] or with an approximate ℓ^0 optimization [6]; in both cases, the main problem is that they remind to the solution of non-negative least-square problems which can be computationally expensive. Another way to obtain sparse vectors was proposed by Ryu and Boyd in [25] and it consists in replacing the ℓ^0-norm with the use of the ℓ^1-norm; this norm naturally yields quadrature rules that are sparse and furthermore the offline problem can be cast as a linear program (LP) efficiently treated by the SIMPLEX algorithm [25, 16, 30] or it can be cast with the LASSO algorithm [29]. We believe that the sparsification thorough ℓ^1-norm, although demonstrated to work for some test problems in the papers above mentioned, has an intrinsic problem with the objective it wants to reach which limits its efficiency.

In Chapter 3 we clarify the motivation of the precedent statement as well as the reasons for why ℓ^1-norm provides naturally sparse vector solutions in many other contests of application are explained.

Chapter 3

Hyper-reduction through the FOCUSS algorithm

This chapter is the core of our work, indeed here we present FOCUSS, which is an algorithm originally introduced by Irina F. Gorodnitsky and Bhaskar D. Rao [8] for the sparse signal reconstruction and that we had the intuition to adapt as a cubature method for the hyper-reduction. We present some modifications made to their version of FOCUSS algorithm in order to ensure the positiveness of the weights and to overcome some numerical problems.
Through the simple example of the polynomial problem, we show how to use this algorithm to get sparse quadrature rules and its efficiency respect the ℓ^1-norm minimization algorithm LASSO. We also compare this method with the heuristic approach of [10], the nodal vector interpolation method EIM [4] and the dual SIMPLEX algorithm used in [25, 16, 30]; the result of this comparison evidences the fact that the FOCUSS algorithm is the only one able to recover the *Gauss-Legendre quadrature rule*, which, for a given family of polynomial functions of degree n, is the more sparse quadrature rule able to exact integrate the family of functions.
Before proceeding with the introduction of FOCUSS and the explanation of how it works, the problem of finding a sparse quadrature rule is introduced from a generic point of view. Furthermore, we dedicate the second section to highlight why the ℓ^1-norm minimization is usually involved in the contest of vector sparsification.

3.1 From the evaluation of integrals to the solution of undetermined linear system

Let's take a family of scalar functions $\mathcal{F}$ with finite cardinality $|\mathcal{F}| = m$ defined over $\Omega \subset \mathbb{R}^N$ with Ω open, limited and connected; we demand that each element f_i of $\mathcal{F}$ is at least integrable over Ω, therefore we require $f_i \in L^\infty(\Omega)$ for $i = 1, .., m$ since we deal with a bounded domain.
Suppose we want evaluate

$$\int_\Omega f_i \,\mathrm{d}\Omega \qquad \forall i = 1, .., m; \tag{3.1}$$

as a first idea we can use an high accuracy quadrature formula involving n points with $n >> m$,

$$\int_\Omega f_i d\Omega \approx \sum_{j=1}^{n} f_i(\mathbf{x}_j) w_j \qquad \forall i = 1, .., m. \tag{3.2}$$

For our propose we can consider the values gotten with (3.2) as the exact values of (3.1) and we can collect them in a vector $\mathbf{b} \in \mathbb{R}^m$, while the values $f_i(\mathbf{x}_j)$ and w_j are collected respectively in a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and in a vector $\mathbf{w} \in \mathbb{R}^n$, therefore we can rewrite expression (3.2) as the algebraical equation

$$\mathbf{A}\mathbf{w} = \mathbf{b}. \tag{3.3}$$

If we imagine not having the vector of weights $\mathbf{w}$, we can look at (3.3) as a linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{3.4}$$

where $\mathbf{A}$ is the coefficients matrix of the linear system, $\mathbf{x}$ the vector containing the unknowns and $\mathbf{b}$ the vector of known terms. The system (3.4) is underdetermined since $m << n$ and therefore it admits ∞^{n-q} solutions where $q \leq m$ is the rank of the matrix $\mathbf{A}$. Obviously $\mathbf{w}$ is a solution of (3.4) but now we can look for solutions of the system which are "sparse", that is, solutions where the number of non zero entries is less or equal m. We also wish that each non zero entry of $\mathbf{x}$ is positive, the need of this constrain has been clarified in the previous chapter when we have spoken about the cubature methods.
In order to get a sparse solution of the system with positive non zero entries, we propose the use of the algorithm FOCUSS, the general idea under this algorithm and all the improvements made to adapt it to our necessities are reported in Section 3.3.

3.2 Sparse solution through ℓ^1-norm minimization

The most natural way to require that a vector $\mathbf{x} \in \mathbb{R}^n$ is sparse, which means that the number of non-zero entries m has to be such that $m \ll n$, is to ask that its ℓ^0-norm is minimum. With ℓ^0-norm we mean the norm that counts the number of non-zero elements of a vector; note that we should avoid to state ℓ^0 as a norm since it does not satisfy the property $\|\alpha\mathbf{x}\|_{\ell^0} = |\alpha| \cdot \|\mathbf{x}\|_{\ell^0}$. Unfortunately, it is quite complicated and there are not efficient algorithms to express the minimization of an objective vectorial function through the ℓ^0-norm.

The most popular way to overcome the algorithmic implementation issues of the ℓ^0-norm is to replace it with the ℓ^1-norm which has already been mentioned in Chapter 2 and allows to cast efficiently the problem as a linear programming one. It is now necessary to give the idea of why ℓ^1-norm succeeds also in providing sparse vectors.

If we limit ourselves to only look at the minimization of the ℓ^1-norm of a vector, it is absolutely not clear the reason why this should give us a sparse representation of the vector. What we should do is to look to an undetermined system as (3.4) and the ℓ^1-norm penalty as a whole. To clarify the meaning of this expression, let's take a concrete example.

Suppose we want to find a line that matches a set of points in 2D space. We

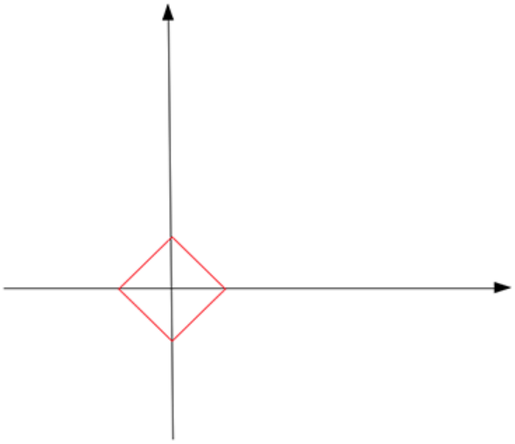

Figure 3.1: Shape of the ℓ^1 ball.

know that at least 2 points are needed to fix a line. In the case the training data has only one point, we have infinite solutions: every line that passes through the point is a solution. Suppose the point is at $[10,\ 5]$; a line is defined as a function $y = ax + b$. Then the problem is to find a solution to this equation:

$$[10\ \ 1] \times \begin{pmatrix} a \\ b \end{pmatrix} = (5),$$

therefore all the points of the line $b = 5 - 10a$ are possible solutions.
Now if we draw all points that have a ℓ^1-norm equals to a constant c, they form the shape of Figure 3.1. The shape looks like a tilted square, in high dimension space it would be an octahedron. Notice that on this red shape not all points are sparse, only on the tips the points are sparse. Now the way to find a sparse solution is enlarging this red shape from the origin by giving an ever-growing c to "touch" the line. The key intuition is that the touch point is most likely at a tip of the shape. Since the tip is a sparse point, the solution defined by the touch point is also a sparse solution. As an example, in the graph of Figure 3.2, the red shape grows 3 times until it touches the blue line $b = 5 - 10a$. The touch point $[0.5,\ 0]$, as it can be seen, is at a tip of the red shape and is a sparse vector. Therefore we say, by finding the solution point with the smallest ℓ^1-norm, (0.5), out of all possible solutions (points on the blue line), we find a sparse solution $[0.5,\ 0]$ for our problem. At the

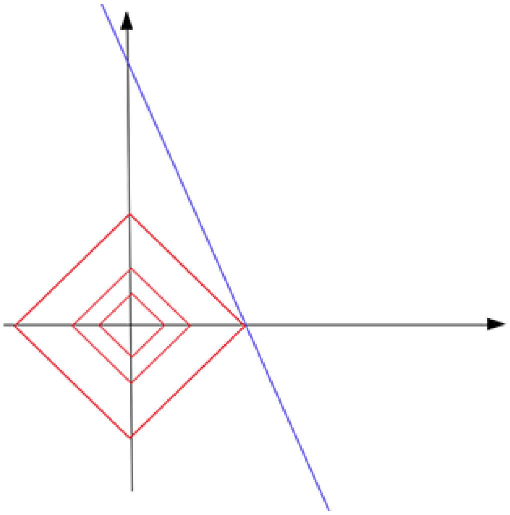

Figure 3.2: Touched point.

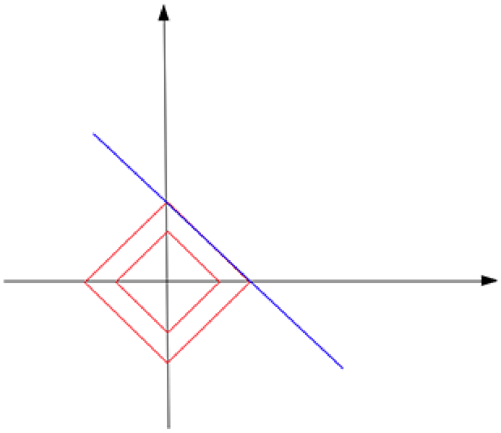

Figure 3.3: No sparse recovery case.

touch point, the constant c is the smallest ℓ^1-norm you could find within all possible solutions.

The intuition of using ℓ^1-norm is that the shape formed by all points whose ℓ^1-norm equals to a constant c has many tips (spikes) that happen to be sparse (lays on one of the axes of the coordinate system). Now if we grow this shape to touch the solutions we find for our problem (usually a surface or a cross-section in high dimension), the probability that the touch point of the 2 shapes is at one of the "tips" or "spikes" of the ℓ^1-norm shape is very high. This is why ℓ^1-norm minimization of an objective function, under linear constrain such us (3.4), works in recovering sparse vectors.

Unfortunately, ℓ^1-norm does not always touch the solution at a tip. Indeed, suppose we still want to find a line out of 2D points, but this time, the only training data is a point $[1,\ 1000]$. In this case, the solution line $b = 1000 - a$ is in parallel to one of the edges of the ℓ^1-norm shape as shown by Figure 3.3. Eventually, they touch on an edge, not by a tip. Not only we cannot have a unique solution this time, but most of the regularized solutions are still not sparse.

But again, the probability of touching a tip is very high. And this is even more correct for high dimension, real-world problems. Indeed, when the coordinate system has more axes, the ℓ^1-norm shape has more tips.

Now the question becomes: is the ℓ^1-norm the best kind of norm to find a sparse solution? We have already said that the best option would be to use directly the ℓ^0-norm but then we should face other problems. By the way,

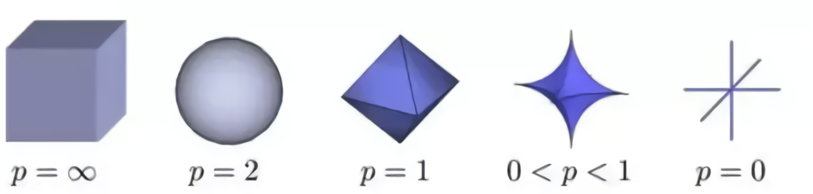

Figure 3.4: Shape of the 3D ball for different norms.

there is still a class of norms which can be considered; it turns out that the ℓ^p-norm with $0 < p < 1$ gives the best result. This can be explained by looking at the shapes of different norms in Figure 3.4. When $p < 1$, the shape is more "scary", with more sharpen, outbreaking spikes. Whereas when $p = 2$, the shape becomes a smooth, non-threatening ball. Unfortunately, the direct use of the ℓ^p-norm to minimize the objective function has the same implementation complications of ℓ^0. Nevertheless, something can still be done with an "indirect" ℓ^p-norm minimization approach and indeed, this is the content of the next section.

3.3 FOCUSS

When we are in front of a system like (3.4) we know that the solution is not unique, but if we ask for the minimum norm solution $\mathbf{x}_{mn}$, that is the one minimizing the ℓ^2-norm, the following proposition provides a way to express it uniquely.

Proposition 3.1 *One particular solution of the undetermined system* (3.4)*, assuming* $\mathbf{A}$ *full rank so that* $\mathbf{A}\mathbf{A}^T$ *is invertible, is*

$$\mathbf{x}_{mn} = \mathbf{A}^{+}\mathbf{b}, \tag{3.5}$$

where $\mathbf{A}^{+} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}$ *denotes the Moore-Penrose inverse*[1] [5]*. This solution is the one that minimizes* $\|\mathbf{x}\|_{\ell^2}$*, i.e.* $\mathbf{x}_{mn}$ *is the solution of the optimization problem:*

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|\mathbf{x}\|_{\ell^2}$$
$$\text{subject to } \mathbf{A}\mathbf{x} = \mathbf{b}. \tag{3.6}$$

---

[1] Note that the Moore-Penrose inverse is more generally defined as $\mathbf{A}^H(\mathbf{A}\mathbf{A}^H)^{-1}$. In this contest, since there are no complex quantities involved, it is enough to consider the transport matrix.

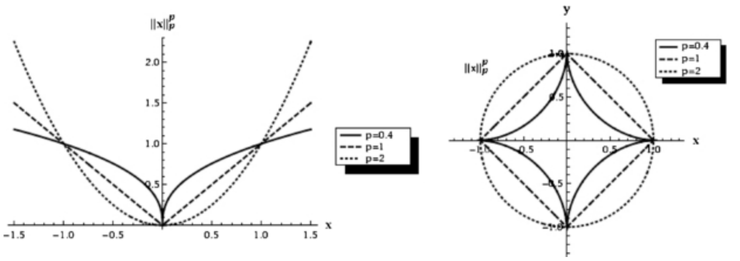

Figure 3.5: Shapes of the 1D polynomial functions x^p and of the unit balls when making use of the $\|\cdot\|_{\ell^p}$-norms.

Proof. Substituting (3.5) in (3.4) it is clear that $\mathbf{x}_{mn}$ is a solution of the system. Now suppose $\mathbf{A}\mathbf{x} = \mathbf{b}$, therefore $\mathbf{A}(\mathbf{x} - \mathbf{x}_{mn}) = \mathbf{0}$ and

$$
\begin{aligned}
(\mathbf{x} - \mathbf{x}_{mn})^T \mathbf{x}_{mn} &= (\mathbf{x} - \mathbf{x}_{mn})^T \mathbf{A}^T (\mathbf{A}\mathbf{A}^T)^{-1} \mathbf{b} \\
&= (\mathbf{A}(\mathbf{x} - \mathbf{x}_{mn}))^T (\mathbf{A}\mathbf{A}^T)^{-1} \mathbf{b} \\
&= \mathbf{0}
\end{aligned}
$$

i.e., $(\mathbf{x} - \mathbf{x}_{mn}) \perp \mathbf{x}_{mn}$, so

$$
\|\mathbf{x}\|_{\ell^2}^2 = \|\mathbf{x} + \mathbf{x}_{lm} - \mathbf{x}_{lm}\|_{\ell^2}^2 = \|\mathbf{x}_{lm}\|_{\ell^2}^2 + \|\mathbf{x} - \mathbf{x}_{mn}\|_{\ell^2}^2 \geq \|\mathbf{x}_{mn}\|_{\ell^2}^2 ,
$$

which means that $\mathbf{x}_{mn}$ has the smallest ℓ^2-norm among all the solutions of (3.4). □

Unfortunately, the minimum norm solution does not provide a sparse solution. Rather, it has the tendency to spread the energy among a large number of entries of $\mathbf{x}$ instead of putting all the energy into just a few entries; nevertheless, it is the starting point to understand the idea under FOCUSS. Suppose that instead the ℓ^2-norm we search for a solution which minimize the ℓ^p-norm with $0 < p < 1$, for this kind of solution the entries x_j which are very small ($x_j \ll 1$) are strongly penalized since the elevation to a power less than 1 gives to them a greater contribution to the norm that has to be minimized compared the elevation to power 2 as in the ℓ^2-norm, see Figure 3.5. Unfortunately, there is not a simple expression as (3.5) to find the minimum ℓ^p-norm solution since ℓ^p is not an Hilbert spaces if $p \neq 2$; nevertheless, something can be done replacing the minimization of $||\mathbf{x}||_{\ell^2}$ with the minimization of the weighted norm $||\mathbf{W}^{-1}\mathbf{x}||_{\ell^2}$, where $\mathbf{W}$ is a square invertible matrix. The solution minimizing this norm is given by

$$
\mathbf{x} = \mathbf{W}((\mathbf{A}\mathbf{W})^{+})\mathbf{b}. \tag{3.7}
$$

In FOCUSS $\mathbf{W}$ is chosen properly in order to obtain something approximating, implicitly, the minimization of an ℓ^p-norm. It is an iterative algorithm

Algorithm 1 `FOCUSS`

```
    Input: $A$, $b$, $x_0$, $p$, maximum iteration I, tolerance for convergence tol
    Output: $x$
for $k = 1$ to $I$ do
    $W = \mathrm{diag}(x_{k-1}^p)$
    $AWWA^T y = b \Rightarrow y$ (Solve)
    $x_k = WWA^T y$
    $e = \|x_k - x_{k-1}\| \,/\, \|x_k\|$
    if $e <$ tol then
        break
    end if
end for
```

which starts from a given initial solution $\mathbf{x}_0$ and at each step k the matrix $\mathbf{W}$ is defined, using the Matlab syntax for functions, as $\mathbf{W} = \mathrm{diag}(\mathbf{x}_{k-1}^p)$; then the solution $\mathbf{x}_k$ is computed as in (3.7). After some iterations of the algorithm we have $\mathbf{x}_{k-1} \approx \mathbf{x}_k$ and consequentially the objective minimized at each step becomes

$$||\mathbf{W}^{+}\mathbf{x}_k||_{\ell^2}^2 = \sum_{i=1,(x_i)_{k-1}\neq 0}^{n} \left(\frac{(x_i)_k}{(x_i)_{k-1}^p} \right)^2 \approx \sum_{i=1}^{n} (x_i)_k^{2-2p}$$

that corresponds to minimize the ℓ^{2-2p}-norm. Since we said that small entries will go to zero when they are elevated by a power between 0 and 1 we need to require that $0 < 2-2p < 1$ which brings to the constrain $0.5 < p < 1$. The convergence of the algorithms to a sparse solution is proven in [8], we only underline that in general there are many sparse solutions for system (3.4) and the one obtained through FOCUSS strongly depends from the choice of the starting solution $\mathbf{x}_0$. Given the space $\mathbb{R}^n$, one has to think that all the sparse solutions of (3.4) are points of this space which are basins of attraction of the algorithm, therefore the algorithm will automatically converge to the sparse solution which basin of attraction contains $\mathbf{x}_0$.

The first version of FOCUSS, the one which is also presented in [8], is reported in Algorithm 1.

3.3.1 How to adapt FOCUSS to be an efficient cubature method

FOCUSS, as it is provided in Algorithm 1, is not optimal for our propose since:

- theoretically, most of the entries of $\mathbf{x}_k$ asymptotically converge to zero but never reach it, cause we work in finite precision arithmetic;
- nothing ensure that, even taking an initial solution with positive entries, the entries remain positive for the final solution;
- $\mathbf{A}$ can be ill-conditioned, therefore the algorithm can be unstable due to the solution of the associated linear system of $(\mathbf{A}\mathbf{A}^T)^{-1}$.

The first problem can be easily overcome approximating to zero the entries of the solution that are smaller than a prescribed tolerance. For the second problem, since FOCUSS is based on the repeated application of linear continuous operators, we can introduce the dependence from a pseudo-time τ, and we can formulate FOCUSS as an algorithm which discretize the ordinary differential (ODE) equation

$$\frac{d\mathbf{x}}{d\tau} = F(\mathbf{x}, \tau) \tag{3.8}$$

whose discrete version is

$$\frac{\Delta\mathbf{x}}{\Delta\tau} = \mathbf{W}\mathbf{W}\mathbf{A}^T(\mathbf{A}\mathbf{W}\mathbf{W}\mathbf{A}^T)^{-1}\mathbf{b} - \mathbf{x}. \tag{3.9}$$

Note that taking $\Delta\tau = 1$ we recover the standard version of FOCUSS. In order to keep all the entries positive, we can use a method to solve ODE which is able to adapt the discrete step $\Delta\tau$ to not obtain negative entries in the solution, e.g. `ode23` or `ode45` of Matlab.

Another approach to enforce positiveness of the entries of the solution is based on the under relaxation. Indeed, given the solution of step k, if it has negative entries while the solution at step $k-1$ has not, we can impose

$$\mathbf{x}_k = \alpha\mathbf{x}_k + (1-\alpha)\mathbf{x}_{k-1}; \tag{3.10}$$

where α is such that the largest negative entry j of $\mathbf{x}_k$ becomes zero, i.e.

$$\alpha = -\frac{(x_j)_{k-1}}{(x_j)_k - (x_j)_{k-1}}. \tag{3.11}$$

This is repeated until all the entries of $\mathbf{x}_k$ are non-negative.
Note that, since $(x_j)_{k-1}$ is negative, we speak of under-relaxation cause it holds that $0 \leq \alpha \leq 1$; moreover, the new $\mathbf{x}_k$ defined by (3.10) is a linear combination of two vector solutions of (3.4), and therefore it is still a solution of the same system.
We decided to use the under-relaxation on FOCUSS in the following way: first we run the version without constrain on the non-negativeness of the weights, then, if the final solution $\mathbf{x}$ has some negative entries, we run again the algorithm but with a version where is implemented the under-relaxation. We do not run directly the "under-relaxation FOCUSS" for two reasons: (i) if at step k an entry is negative, it does not mean that the final values of the entry will be negative, (ii) to force an entry to be zero from one side ensures the non-negativeness of the weights at each step of the algorithm, but from the other side, it makes us lose the bias information of the matrix $\mathbf{A}$ when we initialize with an uniform value in all the entries of the initial solution. We explain the meaning of this last observation at the end of this subsection.
For the third problem two solutions are already provided in [8], in particular the one regarding the *Tikhonov Regularization* is always implemented in our versions of FOCUSS. It consists in replacing problem (3.6) with the following one

$$\min_{\mathbf{x}\in\mathbb{R}^n} \left[\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_{\ell^2}^2 + \lambda \left\|\mathbf{W}^{-1}\mathbf{x}\right\|_{\ell^2}^2 \right], \tag{3.12}$$

where λ is the regularization parameter that has to be chosen before solving (3.12). Basically, setting in a proper way λ, we decide if we prefer to integrate well the family of functions chosen or to have a quadrature rule with a greater number of zero entries. Indeed, with a high value of λ, it will be better for the objective of our minimization to have a solution with more zero entries. Problem (3.12) is solved in the same way done by the first version of FOCUSS and the system to determine iteratively is

$$\mathbf{W}^{-1}\mathbf{x} = \mathbf{A}_W^T(\mathbf{A}_W\mathbf{A}_W^T + \lambda\mathbf{I})^{-1}\mathbf{b}, \tag{3.13}$$

where $\mathbf{A}_W = \mathbf{A}\mathbf{W}$. Now the good conditioning of the system that has to be solved is enforced by the addition of the term $\lambda\mathbf{I}$.
The *Tikhonov Regularization* will be used for a comparison with the LASSO algorithm of Matlab that aims to minimize an object as (3.12), with the only difference in the term multiplying λ that in LASSO is replaced with the ℓ^1-norm of $\mathbf{x}$.
Here we propose two other approaches which demonstrated to work well in

our test problems. This approaches are related with the methods described in Section 1.5 for the construction of a reduced base. Matrix $\mathbf{A}$ is ill-conditioned when some of its rows are almost collinear, this means that we don't need of all the rows of the matrix and therefore the strategies presented in Section 1.5 can be useful to find a base which generates the space containing all the rows. Once the base has been found we can build a new matrix which will be well conditioned.

The first approach is related with the POD method since it involves the use of the orthogonal matrix $\mathbf{U}$, coming from a truncated singular values decomposition. The difference with the POD is that now we do not directly take $\mathbf{U}$ as the matrix containing our base but we use it as a preconditioner for system (3.4); this will also provide a significant reduction in the number of rows of $\mathbf{A}$.

The other approach is based on a reduction of the total number of rows of $\mathbf{A}$ selecting the ones which linear combinations are able to well approximate the other rows; this selection is performed through a greedy algorithm. At each step we put the selected rows in the new matrix $\hat{\mathbf{A}}$ and then we orthogonalize it to get $\mathbf{\Phi}$, whose condition number will be equal 1.

Last modification made on Algorithm 1 comes from this reflection: given two columns of matrix $\mathbf{A}$, $\mathbf{c}_1$ and $\mathbf{c}_2$, such that $\mathbf{c}_1 \approx \beta \mathbf{c}_2$ with $\beta >> 1$, they are clearly linear dependent but $\mathbf{c}_2$ has more probability to be chosen (i.e. to be related with a not nil entry) since the associated entries of the vector of weights has to be greater than the one associated to $\mathbf{c}_1$. If one wishes not giving any priority preference to one column over the others which are linear dependent from it, some operations are necessary. There are two ways to achieve this result: (i) we scale the columns of $\mathbf{A}$ in order to keep as uniform as possible the range of values of the entries of each column or (ii) we choose an initial solution $\mathbf{x}_0$ such that value of each entry x_i is inversely proportional to the norm of the corresponding $\mathbf{A}$ columns. Clearly, the choose of one of the two approach exclude the use of the other since if we rescale the values of the columns we just need to take an $\mathbf{x}_0$ with an uniform value of the entries and if we choose the initial solution according to the values of the columns, rescaling these columns will bring back to the problem we highlighted above. For the first strategy we follow the approach used in [14] while for the second one we refer to [8]. Note that the scaling is not a mandatory process, indeed we can interpret the fact that some columns have larger entries than other ones as a kind of "a priori information" carried by the matrix that we can catch choosing an initial solution which has the same value in all the entries.

Algorithm 2 `CONTINUOUS FOCUSS`

```
Input: $A$, $b$, $x_0$, $p$, $\lambda$, I, tol
Output: $x$
$n \leftarrow \text{size}(A, 2)$
$S(i) = 1/(\text{norm}(A(:, i))\sqrt{n}), \quad i = 1, ..., n$
$S \leftarrow \text{diag}(S)$
$A \leftarrow AS$
tspan$\leftarrow$ [0 inf]
options=odeset('RelTol', $10^{-2}$, 'AbsTol', tol, 'NonNegative', 'vector',
    'Events', @$(t, x)$ myEventsFcn($t$, $x$, $A$, $b$, $\lambda$, $p$, I, tol))'
$[t, X]$ =ode23(@$(t, x)$ odefun($t$,$x$,$A$,$b$,$\lambda$,$p$),tspan,x0,options);
$x \leftarrow SX(\text{end}, :)^T$
for $i = 1$ to $n$ do
    if $|x(i)|$ <tol·max$|x|$ then
        $x(i) \leftarrow 0$
    end if
end for
```

This topic concerning the scaling is more important than what it seems. Indeed, the result of a scaling operation changes the basin of attraction of the sparse solution of the system and therefore an initialization from the same $\mathbf{x}_0$ can lead to two different final solutions if once we apply a scaling and the other time no. We only give a general idea about the influence of the scaling, enough to understand some choices done in our applications; if one wants to go deeply on this topic it is strongly recommended to read the section about the basin of attraction in [8].

Taking into account some of the adjustments we have described, Algorithm 2 provides the continuum version of FOCUSS. Note that the event function `myEventsFcn` is implemented exactly as `odefun` with the addition of the conditions to stop the computation of `ode23` when a maximum values of t is reached or when the changes in the solution are lower than a prescribed tolerance.

Algorithm 3 `ODEFUN`

```
Input: $t$ $x$ $A$, $b$, $p$, $\lambda$
Output: $f$
$x_{k-1} \leftarrow x$
$[m,\ n] \leftarrow \text{size}(A)$
$x_k \leftarrow \text{zeros}(n, 1)$
$L \leftarrow \lambda\text{speye}(m)$
$w_i \leftarrow (x_i^{2p})_{k-1}, \quad i = 1, ..., n$
$\text{ind} \leftarrow \text{find}(w^2)$
$w \leftarrow w(\text{ind})$
$A \leftarrow A(:, \text{ind})$
$(A\text{diag}(w)A^T + L)y = b \Rightarrow y$ (Solve)
$x_k(\text{ind}) = wA^Ty$
$f = x_k - x_{k-1}$
```

3.3.2 On the Offline number of computational operations of FOCUSS

The principal computational effort of each iteration of FOCUSS is given by the matrix multiplication $\mathbf{A}_W\mathbf{A}_W^T$, where we remember $\mathbf{A}_W \in \mathbb{R}^{m\times n}$. Consequentially the operation has a cost of $\mathcal{O}(m^2n)$ arithmetic computation. We should multiply this cost for the total number of iterations k_c necessary for the convergence of the algorithm, but since in each iteration we delete the rows of the matrix $\mathbf{A}_W$ associated with the components of $\mathbf{x}$ already converged to zero, the final total number of operations it is expected to be less than $\mathcal{O}(k_cm^2n)$.

We point that the number of operations involved in the SIMPLEX algorithm to solve a linear programming problem is of the same order of the one of FOCUSS. Indeed, it requires in each iteration i to compute $\mathbf{B^{-1}N}_i$ with $\mathbf{B}^{-1} \in \mathbb{R}^{m\times m}$ and $\mathbf{N}_i \in \mathbb{R}^{m\times n}$.

3.4 Polynomial test case

For this test problem we take as $\mathcal{F}$ a set of polynomial functions with the same degree, defined over the domain $[-1,\ 1]$. Note that choosing this interval is not restrictive since once we know a quadrature rule over $[-1,\ 1]$ we can use it for every generic interval $[a,\ b]$, indeed

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x = \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b-a}{2}x+\frac{a+b}{2}\right)\,\mathrm{d}x \approx \\ \approx \frac{b-a}{2}\sum_{i=1}^{N} w_i f\left(\frac{b-a}{2}x_i+\frac{a+b}{2}\right); \end{aligned} \tag{3.14}$$

and therefore the relation of nodes and weights with the ones over the interval $[-1,\ 1]$ is

$$\bar{w}_i = \frac{b-a}{2}w_i \text{ and } \bar{x}_i = \frac{b-a}{2}x_i+\frac{a+b}{2} \quad \forall i = 1,...,N. \tag{3.15}$$

The exact value of (3.1) is calculable by the use of a *Gauss-Legendre* (GL) *quadrature rule*, indeed if we choose a rule of this type involving t points we are able to integrate exactly every polynomial function with degree d less or equal $2t-1$. Taking n points where we evaluate each polynomial of $\mathcal{F}$, which total number we remember to be m with $m \ll n$, we obtain the system (3.4) where the vector of the exact integrals $\mathbf{b}$ is computed using the Gauss-Legendre quadrature rule. In particular, t can be taken such that $t < m$ and consequentially one sparse solution of (3.4), assuming to evaluate the polynomial also in the Gauss-Legendre points, is given by the Gauss-Legendre quadrature rule.

3.4.1 FOCUSS applied on the polynomial test case

In this subsection we show the results obtained with some of the different versions of our algorithm. For now, we consider $t = 10$ Gauss-Legendre points, 101 polynomials with random coefficients of degree $d = 19$ and $n = 2000$ equispaced points, plus the Gauss-Legendre ones, where the polynomials have been evaluated. The matrix obtained in this way is of dimension 101×2000, in order to reduce the number of rows so to enforce sparsity in the solution (the number of non zero entries found has to be less or equal the number of rows of the matrix), a snapshot selection through greedy approach is adopted and it allows to reduce the number of rows to only 20, that is also the dimension of the vectorial space of polynomial of degree 19. The value of p taken is

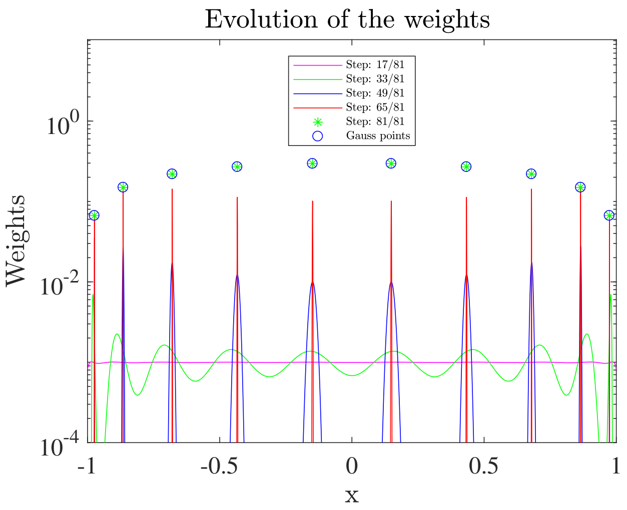

Figure 3.6: Evolution of the solution weights using FOCUSS without columns' scaling. One can see how the algorithm perfectly catches the Gauss-Legendre quadrature rule.

0.7 while $\lambda = 0$ since we run the computation with $\mathbf{\Phi}$. No scaling is adopted over the columns of $\mathbf{\Phi}$ and the initial solution $\mathbf{x}_0$ is chosen such that it has the same value in all the entries and its norm is equal to the measure of the domain, i.e. 2.

Figure 3.6 shows the most important result we obtained with FOCUSS. With the set up previously described, FOCUSS is not only able to find a sparse solution but it also, among all the sparse quadrature rule, catches the Gauss-Legendre one which, using only 10 nodes, is the most sparse solution possible beyond that being able to exact integrate all the polynomial functions with degree less or equal 19. To get the Gauss-Legendre quadrature rule is fundamental the decision of not applying a scaling operation so that we can take advantage of the information carried by the matrix $\mathbf{\Phi}$.

Figure 3.7 shows the result when a scaling operation is applied. It can be seen that we always get a sparse quadrature rule, but not any more the Gauss-Legendre one; note that this does not mean that scaling the columns is not possible to get the Gauss-Legendre rule, rather according to [8] the probability to get the more sparse solution (as is the G.L. rule for our problem) taking a random initial solution is maximized in this way, but the uniform initialization does not converge any more to the Gauss-Legendre quadrature

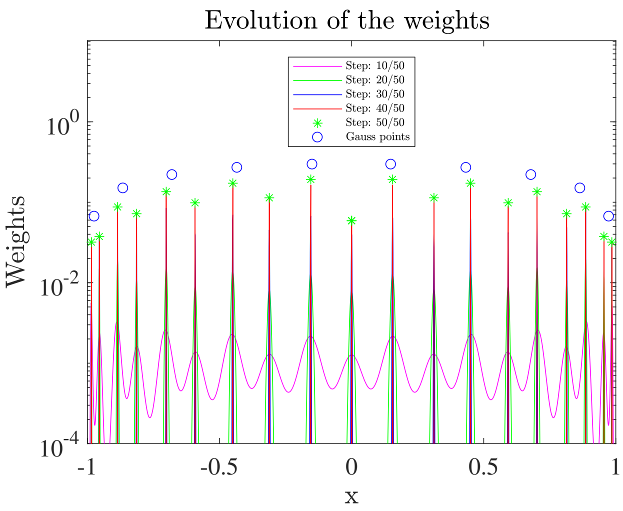

Figure 3.7: Evolution of the solution weights using our cubature algorithm FOCUSS with the application of the scaling operations on the columns.

rule.

From the two previous figures it is also possible to observe the evolution of the weights. At the begin the solution is flattened in all the domain to then shows peaks around those that are the points of the quadrature rule to which it converges.

We showed that setting the problem as we described, the algorithm we propose is able to recover the Gauss-Legendre quadrature rule. This represents a step forward compared to some of the other known algorithms, indeed using EIM [4], HEURISTIC [10] and dual-SIMPLEX [16] methods, with the same set up, we do not recover the G.L. rule as reported by Figure 3.8. The other three techniques are able to find a sparse quadrature rule but not the Gauss-Legendre one, as instead we are able to do with FOCUSS. Moreover, we have the proof of what we have already mentioned in Chapter 2, the interpolation then integration methods, such as EIM, can not ensure the positivity of the weights.

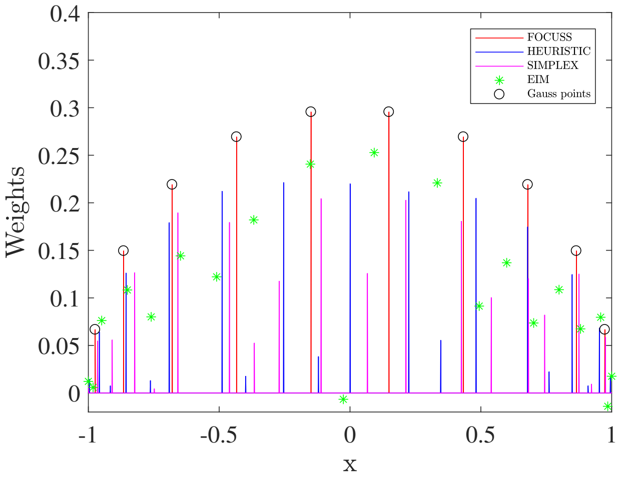

Figure 3.8: Sparse quadrature rule obtained with: FOCUSS, HEURISTIC, SIMPLEX and EIM algorithms: among all, only FOCUSS is able to recover the G.L. rule and it is also shown the fact that EIM returns negative weights.

3.4.2 Comparison between the implicit ℓ^p-norm minimization of FOCUSS and the explicit ℓ^1-norm minimization of LASSO

In this subsection we point the attention on the comparison between the implicit ℓ^p-norm minimization of FOCUSS and the explicit ℓ^1-norm minimization of LASSO algorithm.

Before showing the results of our conclusion we believe we need to clarify a statement used in the previous chapter, i.e.: ℓ^1-norm based minimization algorithms have an intrinsic problem in the contest of sparse quadrature rule recovery.

In Section 3.2 of this chapter we have already highlighted, for a general contest, that a minimization through ℓ^p-norm would be most desirable than the ℓ^1. Moreover, in the framework of sparse quadrature rule recovery, the power of the ℓ^1 approach is further limited by this constrain: we require that the rule is at least able to exactly integrate constant functions. This means that, if the function is equal one in all the domain and the weight are constrained to be positive

$$|\Omega| = \int_\Omega 1 d\Omega = \sum_{j=1}^{n} w_j = ||\mathbf{w}||_{\ell^1}, \tag{3.16}$$

it follows that the ℓ^1-norm of the vector of the weights has to be equal to the measure of the domain of integration. Equation (3.16), if not already included by (3.4), it is usually imposed as an additional constrain when searching for a sparse quadrature rule, consequentially minimizing the ℓ^1-norm loses its meaning if we already know which has to be the value of the norm. The strength of FOCUSS is that it replaces the use of ℓ^1-norm with an implicit ℓ^p-norm for which there are no constrains to take into account (indeed, none of the equations of (3.4) can be interpreted as a constrain on the ℓ^p-norm).
Here, using this simple test problem, we show that the performance of FOCUSS with *Tikhonov Regularization* (3.12) totally overcomes the ones of LASSO in terms of residual and number of non zero elements of the solution.
The objective function minimized by LASSO is

$$\min_{\mathbf{x}\in\mathbb{R}^n}\left[\frac{1}{2n}\left\|\mathbf{A}\mathbf{x}-\mathbf{b}\right\|_{\ell^2}^2+\lambda\left\|\mathbf{x}\right\|_{\ell^1}\right]; \tag{3.17}$$

the value of λ we selected for this test goes from 10^{-4} to 10^0, the quadrature rules found with both the methods for different λ are tested computing the relative residual norm

$$\frac{\left\|\mathbf{A}\mathbf{x}-\mathbf{b}\right\|_{\ell^2}}{\left\|\mathbf{b}\right\|_{\ell^2}}.$$

Results are shown in Figure (3.9).
From this example it is clear that FOCUSS works much better respect

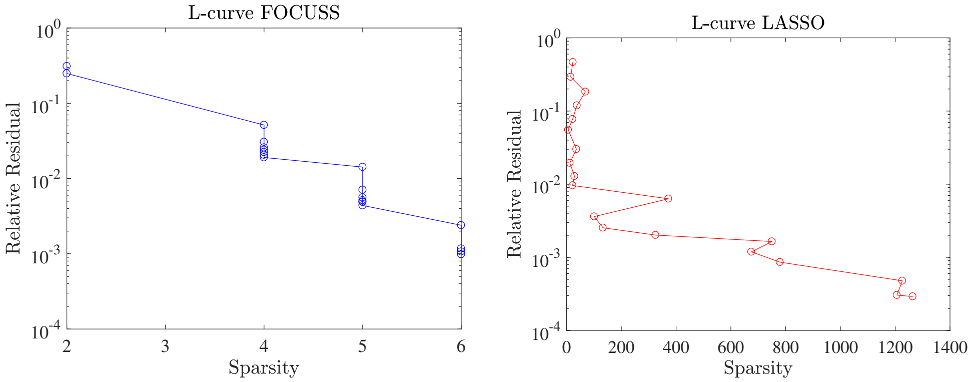

Figure 3.9: Comparison between the FOCUSS' L-curve (left) and the LASSO's L-curve (right).

LASSO since it provides a better residual and more sparse solutions. Indeed, for the lower values of λ, LASSO is not even able to recover sparsity. We also report the fact that, even if we set a greater number of maximum iterations and a lower relative tollerance for convergence than the FOCUSS' ones, the LASSO algorithm is not able to converge and it always reaches the maximum number of iterations.

Chapter 4

Applications

To test the efficiency of FOCUSS as an hyper-reduction technique, we applied the finite elements method (FEM), through Matlab, on two different physical problems combined with the reduced basis approach. The first is a simple Poisson equation defined over a 2D square, the main variable is the temperature and the non-linear parameter is the thermal conductivity κ. The second, more complex problem, is related to the poro-elasticity and it is also defined over a 2D domain. Here the parameters we consider are four, the one defined with a non-linearity is the permeability κ[1].
The structure we follow in the chapter is: (*i*) first we explain our strategy for the finite elements discretization, (*ii*) we clarify how to build the reduced base for hyper-reduction and then (*iii*) we give the complete description of the two chosen problems and we report the results we obtained using our quadrature rule. We also test some of the other quadrature methods previously mentioned in order to make a comparison with FOCUSS.

4.1 Assembly of the discrete operator through Finite Elements Method

For the applications chosen, the physical 2D domains have been discretized with a mesh of triangular elements using the open source software Gmsh. From the file generated by Gmsh we have the number of mesh nodes N_n and

---

[1] In the literature we refereed both thermal conductivity and permeability are indicated with the Greek letter κ, we remark that there is no correlation between this two parameters.

the number of triangular elements N_{ele}.
The FEM discretization has been performed in Matlab through the pre written function `fem_assemble_test1` which makes use of the piecewise linear test functions. It performs the integrations over each element with a quadrature rule involving 4 nodes, which generates automatically a global rule of $N_w = 4N_{ele}$ nodes, and takes as input arguments the topology of the mesh, understood as triangular elements associated with the respective nodes, and the coordinates of the nodes. The outputs are four sparse matrices $\mathbf{D}_x, \mathbf{D}_y, \mathbf{In}$ and $\mathbf{Q}$ defined as follow:

$$\begin{aligned}
&\mathbf{D}_x \in \mathbb{R}^{N_w \times N_n} \quad [D_x]_{ij} = \frac{\partial \phi_j}{\partial x}(\mathbf{x}_i),\\
&\mathbf{D}_y \in \mathbb{R}^{N_w \times N_n} \quad [D_y]_{ij} = \frac{\partial \phi_j}{\partial x}(\mathbf{x}_i),\\
&\mathbf{In} \in \mathbb{R}^{N_w \times N_n} \quad [In]_{ij} = \phi_j(\mathbf{x}_i),\\
&\mathbf{Q} \in \mathbb{R}^{N_w \times N_w} \quad [Q]_{ik} = \mathrm{w}_i \delta_{ik};
\end{aligned} \tag{4.1}$$

where ϕ_j for $j = 1, ..., N_n$ are the piecewise linear test functions and w_i for $i = 1, ..., N_w$ are the weights of the global quadrature rule.
It remains to clarify why `fem_assemble_test1` returns these four matrices instead of directly assembling the discrete operator of our problem. To understand this we make use of an example.
Let us consider the stiffness matrix $\mathbf{K}$ in the simple case in which it relays to the discretization of the bilinear form $a(u, v; \mu)$ when the parameter μ is

constant respect the domain. This means we can write, $\forall v \in V$:

$$\begin{aligned} a(u,v;\mu) &= \int_\Omega \mu \nabla \mathbf{u} \cdot \nabla \mathbf{v} \, \mathrm{d}\Omega = \mu \sum_{k=1}^{N_{ele}} \int_{T_{ref}} \nabla \mathbf{u} \cdot \nabla \mathbf{v} J_k \, \mathrm{d}S = \\ &\approx \mu \sum_{k=1}^{N_{ele}} \int_{T_{ref}} \nabla \left(\sum_{j=1}^{N_n} u_j \phi_j \right) \cdot \nabla \left(\phi_i\right) J_k \, \mathrm{d}S = \\ &= \mu \sum_{j=1}^{N_n} u_j \sum_{k=1}^{N_{ele}} \int_{T_{ref}} \nabla \boldsymbol{\phi}_j \cdot \nabla \boldsymbol{\phi}_i J_k \, \mathrm{d}S \approx \\ &\approx \mu \sum_{j=1}^{N_n} u_j \sum_{k=1}^{N_{ele}} \sum_{l=1}^{4} \nabla \boldsymbol{\phi}_j(\mathbf{x}_l) \cdot \nabla \boldsymbol{\phi}_i(\mathbf{x}_l) J_k(\mathbf{x}_l) w_l \, \mathrm{d}S = \\ &\approx \mu \sum_{j=1}^{N_n} K_{ij} u_j \quad \forall i = 1, ..., N. \end{aligned} \tag{4.2}$$

This in matrix form is equivalent to the matrix-vector product $\mu \mathbf{K}\mathbf{u}$. Now, looking at the expression of K_{ij} and defining $\mathrm{w}_{k \cdot l} = J_k(\mathbf{x}_l) w_l$ we have

$$\begin{aligned} K_{ij} &= \sum_{k=1}^{N_{ele}} \sum_{l=1}^{4} \nabla \boldsymbol{\phi}_j(\mathbf{x}_l) \cdot \nabla \boldsymbol{\phi}_i(\mathbf{x}_l) \mathrm{w}_{k \cdot l} \, \mathrm{d}S = \\ &= \sum_{k=1}^{N_{ele}} \sum_{l=1}^{4} \left(\frac{\partial \phi_j}{\partial x}(\mathbf{x}_l) \frac{\partial \phi_i}{\partial x}(\mathbf{x}_l) + \frac{\partial \phi_j}{\partial y}(\mathbf{x}_l) \frac{\partial \phi_i}{\partial y}(\mathbf{x}_l) \right) \mathrm{w}_{k \cdot l} = \\ &= \sum_{k \cdot l=1}^{N_w} \left(\frac{\partial \phi_j}{\partial x}(\mathbf{x}_{k \cdot l}) \frac{\partial \phi_i}{\partial x}(\mathbf{x}_{k \cdot l}) + \frac{\partial \phi_j}{\partial y}(\mathbf{x}_{k \cdot l}) \frac{\partial \phi_i}{\partial y}(\mathbf{x}_{k \cdot l}) \right) \mathrm{w}_{k \cdot l} = \\ &= \sum_{k \cdot l=1}^{N_w} \left([D_x]_{k \cdot l, j} Q_{k \cdot l, k \cdot l} [D_x]_{k \cdot l, i} + [D_y]_{k \cdot l, j} Q_{k \cdot l, k \cdot l} [D_y]_{k \cdot l, i} \right); \end{aligned} \tag{4.3}$$

and therefore the stiffness matrix can be written as

$$\mathbf{K} = \mathbf{D}_x^T \mathbf{Q} \mathbf{D}_x + \mathbf{D}_y^T \mathbf{Q} \mathbf{D}_y. \tag{4.4}$$

The same procedure can be repeated for any discrete operator coming from a bilinear form of second order differential equations. Another example can be given considering the mass matrix $\mathbf{M}$, for which

$$M_{ij} = \sum_{k \cdot l=1}^{N_w} \left(\phi_j(\mathbf{x}_{k \cdot l})\phi_i(\mathbf{x}_{k \cdot l})\right) \mathrm{w}_{k \cdot l}.$$

It follows that its expression using (4.1) becomes

$$\mathbf{M} = \mathbf{In}^T \mathbf{Q} \mathbf{In}. \tag{4.5}$$

Now it should be clear the vantages of having only these four matrices. Instead of writing a complicated function assembling every possible operator for partial differential equations of the second order, we have a function which only build 4 sparse matrices and then allows to build every operator we could need for problems of the second order.

4.1.1 Galerkin projection for the first reduction stage

When the affine decomposition (1.26) is possible, the first reduction stage, i.e. the Galerkin projection of the problem on a subspace of low dimension, can be algebraically interpreted as in (1.34). Regarding this expression we only have to add the decomposition of the discrete operators we defined above.
For example, the reduced stiffness matrix $\mathbf{K}_N$ can be easily expressed in this way

$$\mathbf{K}_N = \mathbf{V}^T \mathbf{D}_x^T \mathbf{Q} \mathbf{D}_x \mathbf{V} + \mathbf{V}^T \mathbf{D}_y^T \mathbf{Q} \mathbf{D}_y \mathbf{V}; \tag{4.6}$$

where $\mathbf{V}$ is the transformation matrix defined in (1.33).
We observe that the reduced stiffness matrix is not any more sparse but it conserves the spectral properties of the original operator. Indeed, from (4.6) it is immediate to verify the symmetry and since the weights in the diagonal matrix $\mathbf{Q}$ are all positive it is also ensured that $\mathbf{K}_N$ is positive definite.

4.2 The second reduction stage in FEM trough hyper-reduction

What we showed in the previous section works if our problem admits the affine decompositions (1.6) and (1.7), in this section we explain how the second reduction stage has to be performed in order to overcome the nonlinearity of the parameters.

Suppose we make use of M quadrature points for each element, so that we have a total of $N_w = N_{ele} \times M$ quadrature points for our problem. Since the non-linear dependence of our problem from the parameters, our FEM matrices will directly depend from them, implying that we cannot pre-compute Offline the reduced operators since we have to rebuild the matrices every time we change the values of the parameters.
To solve this problem the idea is to speed up the computation of the reduced operators with strongly reducing the number of quadrature nodes involved on their computation. To perform this, we collect snapshots of the reduced matrices as we were collecting snapshots of the solution for the first reduction stage. Indeed, given s set up of parameters $\boldsymbol{\mu}$, we evaluate the reduced operators s times and for each of them we collect their coefficients in a vector $\mathbf{b}$. This vector, in the case of relying to a symmetric matrix (as usually is the stiffness matrix), has $((N+1) \times N)/2$ entries for each choice of $\boldsymbol{\mu}$. Clearly, each coefficient of the reduced matrix comes from approximation of integrals in all the domain followed by the Galerkin projection on the reduced space. For example

$$(K_N)_{i,j} = \mathbf{V}(:,i)^T \mathbf{D}_x^T \mathbf{Q} \mathbf{D}_x \mathbf{V}(:,j) + \mathbf{V}(:,i)^T \mathbf{D}_y^T \mathbf{Q} \mathbf{D}_y \mathbf{V}(:,j), \tag{4.7}$$

where for the sake of simplicity we used the Matlab notation. Note that (4.7) does not take care of the non-linearity, in such case we could have something like this

$$(K_N)_{i,j} = \mathbf{V}(:,i)^T \mathbf{D}_x^T \mathbf{C} \mathbf{Q} \mathbf{D}_x \mathbf{V}(:,j) + \mathbf{V}(:,i)^T \mathbf{D}_y^T \mathbf{C} \mathbf{Q} \mathbf{D}_y \mathbf{V}(:,j), \tag{4.8}$$

where $\mathbf{C}$ is a diagonal matrix and each of its diagonal entry corresponds to the evaluation of the parameter (for example a conductivity coefficient which depends from the temperature) in one of the quadrature points.
In (4.8) if we fix a quadrature node, we can collect the value found in one of the rows entries of a matrix $\mathbf{A}$. Repeating this for every coefficient of the reduced operator and every snapshots, we obtain a matrix of dimension $[(N+1) \cdot N \cdot S/2] \times N_w$ which forms a system like (3.4) where the unknows are the weights of a global quadrature formula. Over this system it can now be applied an algorithm able to find sparse solutions so to obtain a sparse quadrature rule.
Some further clarifications are necessary to well pose the problem. First of all, as in the toy problem of Chapter 3, we do not work normally with all the matrix $\mathbf{A}$ but we adopt a selection of the most "significant" rows of the

matrix in the ways already described in Chapter 1. Second, the process of finding the sparse quadrature rule has to be performed Offline, in this way we can use in the Online phase the new simplified rule. Third, a different quadrature is required for each operator in which a parameter with a non-linear dependence is involved.

4.3 First FEM problem application: the thermal diffusion

The problem we consider is the following

$$\begin{cases} -\nabla \cdot (\kappa \nabla \mathrm{t}) = 0 & \text{on } \Omega, \\ \mathrm{t} = 0 & \text{on } \partial\Omega, \end{cases} \tag{4.9}$$

where t is the temperature, κ the thermal conductivity and Ω a square on the unit side.

As it can be observed the problem is very simple, the only deviation from the standard Poisson problem is in the conductivity κ, defined as follows:

$$\kappa(\mathrm{t}) = \begin{cases} 1 + \mu_1 \mathrm{t}, & \text{if } \mathbf{x} \in \Omega_1 \\ \mu_2 + \mu_2 \mathrm{t}^2, & \text{if } \mathbf{x} \in \Omega \setminus \Omega_1 \end{cases} \tag{4.10}$$

where Ω_1 is a circle of radius 0.25 centred in Ω and μ_1, μ_2 are the two parameters on which the conductivity depends. In Figure 4.1 we show the mesh created for the domain of the problem, consisting in $N_{ele} = 8288$ elements for a total of $N_w = 33152$ quadrature points. For fixed values of μ_1 and μ_2 the non-linearity of the equation is solved thanks to a loop, which means that we start from an uniform value of the temperature t_0 in all the domain (0 for example) and in this way we compute the conductivity $\kappa(\mathrm{t}_0)$ and after the new temperature t_1. We repeat this until at step k the norm of the difference between the temperatures t_k and t_{k-1} is smaller than a prescribed tollerance. This technique is commonly used when dealing with non linearity in parametrized partial differential equations discretization.

The tests of FOCUSS and the comparison with the other hyper-reduction methods for this application has been mostly performed with three scripts. The first is used to build the database containing the snapshots of the solution t and of the reduced stiffness matrix $\mathbf{K}_N$. The second computes the

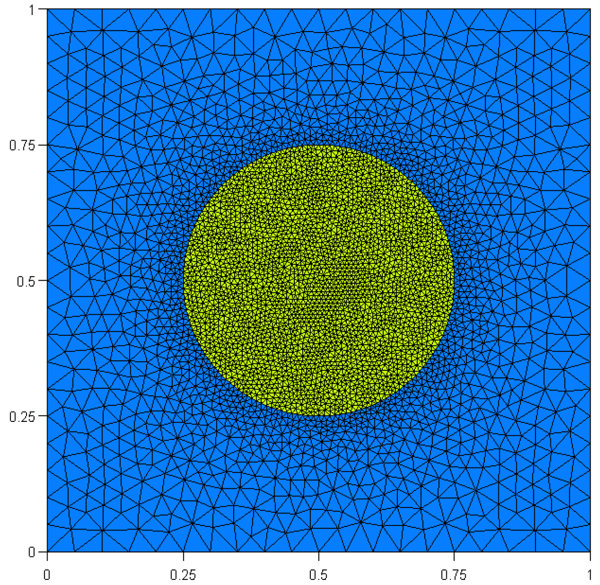

Figure 4.1: Mesh considered for the domain Ω of the thermal conduction problem.

sparse quadrature formulas, and finally in the third script we compare the high-fidelity solution with the reduced and hyper-reduced ones in terms of H^1_0-norm relative error and computational time.
Note that, making use of what exposed in Section 4.1, the approximation of the H^1_0-norm of a function u is easily evaluated as

$$\|u\|^2_{\mathrm{H}^1_0} \approx \mathbf{u}^T \mathbf{D}_x^T \mathbf{Q} \mathbf{D}_x \mathbf{u} + \mathbf{u}^T \mathbf{D}_y^T \mathbf{Q} \mathbf{D}_y \mathbf{u}.$$

4.3.1 Results

We built the database with an uniform train sampling Ξ^{train} of the parametric domain performed with 196 elements, then we applied the POD method for the construction of both the bases of first and second reduction stage and finally the comparison among high-fidelity, reduced and hyper-reduced solutions have been realized taking an uniform test sample Ξ^{test} of 36 elements. The result shown now testifies the efficiency of FOCUSS on this problem.
From Figure 4.2 we can see that the extra error introduced using the reduced quadrature rule is of several order less than the error committed with the reduced basis approximation, furthermore the time demanded by the

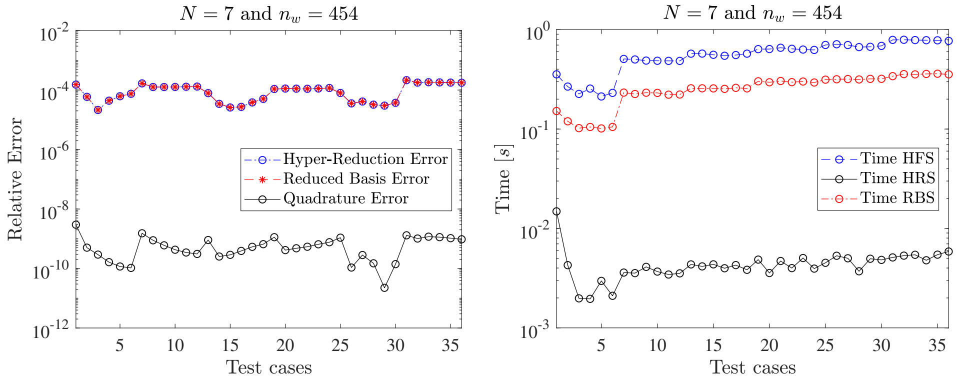

Figure 4.2: H_0^1-norm of the errors between the high-fidelity solution (HFS) and the hyper-reduced solution (HRS, with $n_w = 454$ quadrature points), the high-fidelity solution and the simple reduced basis solution (RBS, with $N = 7$ modes) and the hyper-reduced solution and the reduced basis solution (left). Computational time demanded for the high-fidelity, simple reduced basis and hyper-reduced solutions (right). Results collected using FOCUSS.

hyper-reduced solution is also strongly less than the one for the high-fidelity and simple reduced-basis solutions (which are more or less of the same order of seconds).

To obtain these pictures we used 7 modes and a 454 quadrature rules points, for both the two reduction stages we worked with the proper orthogonalized decomposition (POD). Both values are quite high for a simple problem like this one, indeed the errors we get are extremely low, especially the quadrature error which is several orders less than the one that can be considered acceptable for most of the applications. Nevertheless, we decided to take such values in order to show that our algorithm is able to find a rule which pushes the error very low keeping the boost in computational time.

Table 4.1 proofs that low errors appear also for quadrature rule with less nodes than 454, provided that the number of reduced basis functions is also taken lower than 7. We observe that reasonable results for applications are provided with the only use of $N = 3$ modes and $n_w = 15$ quadrature points.

Finally we can comment Figure 4.3 which is about the comparison of the performance of FOCUSS against the cubature methods introduced in Chapter 2 and then used for the comparison in the polynomial test problem. A greedy selection has been applied before calling the hyper-reduction algorithms.

N	n_w	$\max\limits_{\boldsymbol{\mu}\in\Xi^{\text{test}}} \frac{\left\|\mathrm{t}_N(\boldsymbol{\mu})-\mathrm{t}_N^w(\boldsymbol{\mu})\right\|_{\mathrm{H}_0^1}}{\left\|\mathrm{t}_h(\boldsymbol{\mu})\right\|_{\mathrm{H}_0^1}}$	$\max\limits_{\boldsymbol{\mu}\in\Xi^{\text{test}}} \frac{\left\|\mathrm{t}_h(\boldsymbol{\mu})-\mathrm{t}_N^w(\boldsymbol{\mu})\right\|_{\mathrm{H}_0^1}}{\left\|\mathrm{t}_h(\boldsymbol{\mu})\right\|_{\mathrm{H}_0^1}}$
1	2	$7.62 \cdot 10^{-2}$	$3.10 \cdot 10^{-1}$
2	8	$3.00 \cdot 10^{-3}$	$1.87 \cdot 10^{-1}$
3	15	$6.40 \cdot 10^{-3}$	$1.87 \cdot 10^{-2}$
4	30	$8.33 \cdot 10^{-4}$	$6.50 \cdot 10^{-3}$
5	60	$2.13 \cdot 10^{-4}$	$2.05 \cdot 10^{-3}$
6	80	$1.88 \cdot 10^{-4}$	$2.00 \cdot 10^{-3}$
7	100	$9.76 \cdot 10^{-5}$	$2.16 \cdot 10^{-4}$

Table 4.1: Thermal conduction example: the number of reduced-basis functions N, the number of reduced quadrature points n_w, the maximum difference in the reduced-basis approximation, over the test set Ξ^{test}, using the exact quadrature rule and the reduced quadrature, $\max_{\boldsymbol{\mu}\in\Xi^{\text{test}}} \frac{\left\|\mathrm{t}_N(\boldsymbol{\mu})-\mathrm{t}_N^w(\boldsymbol{\mu})\right\|_{\mathrm{H}_0^1}}{\left\|\mathrm{t}_h(\boldsymbol{\mu})\right\|_{\mathrm{H}_0^1}}$, and the error in the hyper-reduction approximation, $\max_{\boldsymbol{\mu}\in\Xi^{\text{test}}} \frac{\left\|\mathrm{t}_h(\boldsymbol{\mu})-\mathrm{t}_N^w(\boldsymbol{\mu})\right\|_{\mathrm{H}_0^1}}{\left\|\mathrm{t}_h(\boldsymbol{\mu})\right\|_{\mathrm{H}_0^1}}$.

The maximum quadrature error over the test sample chosen has the same trend with almost the same values for all the methods, the only exception is for the EIM which does not show a significant improve in the error when passing from a rule of 250 nodes to one with 400. In Table 4.2 we report those values for the different quadrature rules.
The main differences can be observed in the plot of the computational time demanded for the Offline stage in which the reduced quadrature is compute. The EIM method is the fastest until the number of quadrature points is kept under 150 and then its time graph grows significantly, more than the one of the other methods. We also have to remember that the interpolation-integration methods, like EIM, have the strong drawback of not insuring the positiveness of the weights, indeed in these simulations we found several negative weights in the sparse rules recovered by EIM. FOCUSS and HEURISTIC show the same trend and very close values until $n_w = 250$ quadrature points but, then, the chart of HEURISTIC algorithm stands out. This happens because HEURISTIC makes use of the constrained non-negative least

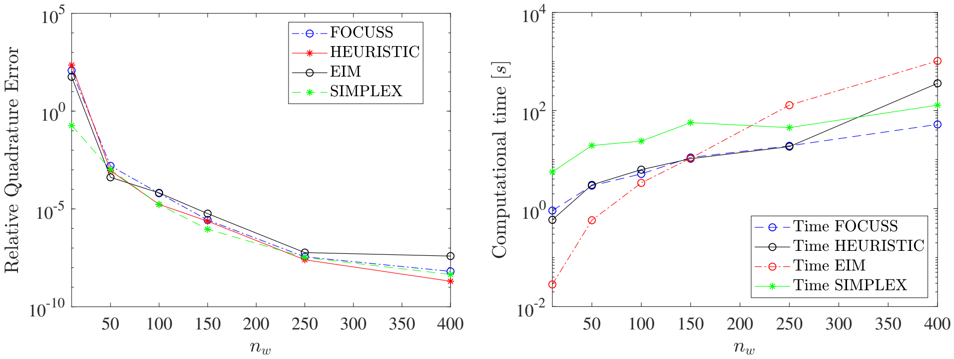

Figure 4.3: Comparison of hyper-reduction methods, over different n_w, concerning the maximum difference in the reduced-basis approximation, over the test set Ξ^{test}, using the exact quadrature rule and the reduced quadrature, $\max_{\boldsymbol{\mu} \in \Xi^{\text{test}}} \frac{\left\| \mathrm{t}_N(\boldsymbol{\mu}) - \mathrm{t}_N^w(\boldsymbol{\mu}) \right\|_{\mathrm{H}_0^1}}{\left\| \mathrm{t}_h(\boldsymbol{\mu}) \right\|_{\mathrm{H}_0^1}}$ (left). Computational time necessary in the offline stage for the recover of the sparse quadrature rule with n_w nodes (right). Results obtained with $N = 7$ reduced basis functions.

square method to ensure the positiveness of the weights which require a high computational effort. The more nodes are necessary, the more frequently the algorithm has to call the constrained non-negative least square method. SIMPLEX chart shows a similar trend compared FOCUSS but the time it requires is significantly higher for every n_w considered.

From the two figures we can extrapolate this final consideration: there is no a method which provides better performances than the others in terms of quadrature error but, in terms of computational time demanded, FOCUSS appears to be the optimal one. Indeed, it has an uniform increase (in the logarithmic scale) respect the number of quadrature points while HEURISTIC and EIM are hard time demanding with the growth of n_w and SIMPLEX, even if it shows the same trend of FOCUSS, it always requires more time than FOCUSS in the computation of the sparse rule.

n_w	FOCUSS	HEURISTIC	EIM	SIMPLEX
10	$1.18 \cdot 10^{\ 2}$	$2.27 \cdot 10^{\ 2}$	$5.68 \cdot 10^{\ 1}$	$1.84 \cdot 10^{-1}$
50	$1.59 \cdot 10^{-3}$	$9.00 \cdot 10^{-4}$	$4.13 \cdot 10^{-4}$	$1.00 \cdot 10^{-3}$
100	$6.33 \cdot 10^{-5}$	$1.71 \cdot 10^{-5}$	$6.62 \cdot 10^{-5}$	$1.70 \cdot 10^{-5}$
150	$2.73 \cdot 10^{-6}$	$2.37 \cdot 10^{-6}$	$5.76 \cdot 10^{-6}$	$9.13 \cdot 10^{-7}$
250	$3.56 \cdot 10^{-8}$	$2.51 \cdot 10^{-8}$	$5.87 \cdot 10^{-8}$	$3.35 \cdot 10^{-8}$
400	$6.42 \cdot 10^{-9}$	$2.01 \cdot 10^{-9}$	$3.90 \cdot 10^{-8}$	$4.49 \cdot 10^{-9}$

Table 4.2: The FOCUSS, HEURISTIC, EIM and SIMPLEX hyper-reduction methods comparison respect the maximum difference in the reduced-basis approximation, over the test set Ξ^{test}, using the exact quadrature rule and the reduced quadrature with n_w nodes, $\max_{\boldsymbol{\mu} \in \Xi^{\text{test}}} \frac{\left\| \mathrm{t}_N(\boldsymbol{\mu}) - \mathrm{t}_N^w(\boldsymbol{\mu}) \right\|_{\mathrm{H}_0^1}}{\| \mathrm{t}_h(\boldsymbol{\mu}) \|_{\mathrm{H}_0^1}}$.

4.4 Second FEM problem application: the poro-elasticity benchmark

Being a quite more complex problem than the simple thermal conduction, we made use of the 2D poro-elasticity benchmark model presented in [12]. The equations describing this problem are:

$$\begin{aligned} -\mu\tilde{\Delta}\mathbf{u} - (\lambda + \mu)\nabla(\nabla \cdot \mathbf{u}) + \nabla p = \mathbf{g}(\mathbf{x}, t) \text{ on } \Omega, \\ S\dot{p} + \nabla \cdot \dot{\mathbf{u}} - \frac{\kappa}{\eta}\Delta p = f(\mathbf{x}, t) \text{ on } 0 < t < T, \end{aligned} \tag{4.11}$$

where $\tilde{\Delta}$ represents the vector Laplace operator, λ and μ are the Lamé coefficients, S the storage parameter, κ the permeability of the porous medium, η is the dynamic viscosity of the fluid, $\mathbf{u}$ is the displacement vector and p is the pore pressure. The right-hand term $\mathbf{g}$ is the density of applied body forces and the source term f represents a forced fluid extraction or injection process. Here, Ω denotes a square 2D domain with a circular cavity on the bottom-left corner; mesh created through Gmsh and boundary conditions imposed following [28]. Domain, mesh and boundary conditions are all visible in Figure 4.4.

The two Lamé coefficients are related with the material properties of the

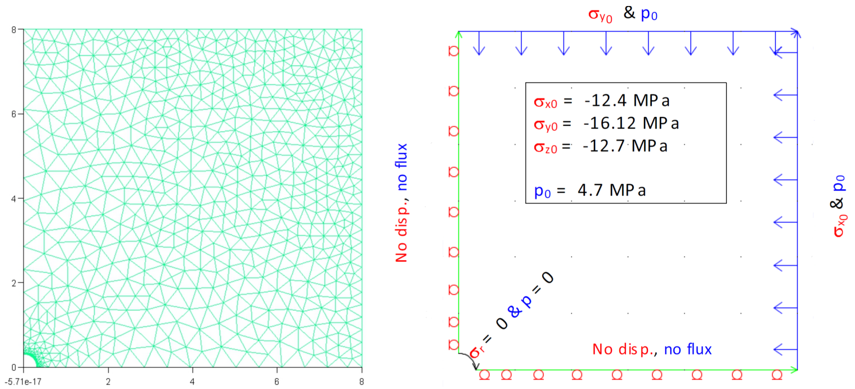

Figure 4.4: 2D domain with circular cavity considered for the poro-elasticity problem. Mesh created with Gmsh (left) and boundary conditions imposed (right).

porous media such as the Young's modulus E and the Poisson's ration ν by

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \qquad \mu = \frac{E}{2(1+\nu)}.$$

Both the E and ν are considered as parameters of the problem together with the storage parameter S and the permeability κ, for a total number of 4 parameters. The parameter we selected to introduce the non linearity is the permeability κ, we made it to depend on the porosity parameter ϕ, which is fixed as $\phi = 0.008$, and on the variation of the volume of the solid, that is the trace of the deformation tensor $\boldsymbol{\varepsilon}$. Given these dependences, the expression of κ is

$$\kappa = \kappa_{\text{ref}} \frac{(\text{tr}(\boldsymbol{\varepsilon}) + \phi)^3}{(1 + \text{tr}(\boldsymbol{\varepsilon}))(1-\phi)^2}. \tag{4.12}$$

where k_{ref} is the permeability at time $t = 0$.
In the following subsections we focus on the implementation issues in terms of finite elements and reduced basis method of the poro-elasticity benchmark to show then the results obtained testing on this problem FOCUSS. Also here the main of the work has been performed with three scripts, one for the construction of the database (snapshots matrices), one for the recover of the sparse quadrature rules and finally one to test the efficiency of the sparse rules computing the L^2-norm relative error introduced by the reduced quadrature rule and the computational time.

.

4.4.1 FEM implementation issues

In this subsection we explain how we deal with the poro-elasticity benchmark implementation through FEM in Matlab; for the sake of simplicity we forget, by the moment, about the non linearity introduced in κ.
The mesh we used at first consists on $N = 888$ nodes and $N_{ele} = 1586$ elements for a total of $N_w = 6344$ quadrature points. No forcing functions have been considered, therefore $\mathbf{g} = \mathbf{0}$ and $f = 0$. Since the problem is time dependent, we took as time solver the Matlab function `ode23t`, the discrete equations in which it works are

$$\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{M}_1^T & \mathbf{M}_2^T & S\mathbf{U} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{u}} \\ \dot{\mathbf{v}} \\ \dot{\mathbf{p}} \end{bmatrix} =$$
$$\begin{bmatrix} -\mu\mathbf{K} - (\lambda+\mu)\mathbf{S}_1 & -(\lambda+\mu)\mathbf{S}_3 & \mathbf{M}_1 \\ -(\lambda+\mu)\mathbf{S}_2 & -\mu\mathbf{K} - (\lambda+\mu)\mathbf{S}_4 & \mathbf{M}_2 \\ \mathbf{0} & \mathbf{0} & -\kappa/\eta\mathbf{K} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{p} \end{bmatrix} + \mathbf{c}_d + \mathbf{c}_n; \quad (4.13)$$

where, remembering what seen in Section 4.1, we have

$$\begin{aligned} &\mathbf{K} = \mathbf{D}_x^T\mathbf{Q}\mathbf{D}_x + \mathbf{D}_y^T\mathbf{Q}\mathbf{D}_y, \quad \mathbf{U} = \mathbf{In}^T\mathbf{Q}\mathbf{In}; \\ &\mathbf{M}_1 = \mathbf{D}_x^T\mathbf{Q}\mathbf{In}, \quad \mathbf{M}_2 = \mathbf{D}_y^T\mathbf{Q}\mathbf{In}; \\ &\mathbf{S}_1 = \mathbf{D}_x^T\mathbf{Q}\mathbf{D}_x, \quad \mathbf{S}_2 = \mathbf{D}_y^T\mathbf{Q}\mathbf{D}_x; \\ &\mathbf{S}_3 = \mathbf{D}_x^T\mathbf{Q}\mathbf{D}_y, \quad \mathbf{S}_4 = \mathbf{D}_y^T\mathbf{Q}\mathbf{D}_y; \end{aligned} \qquad (4.14)$$

while $\mathbf{c}_d$ and $\mathbf{c}_n$ are respectively the vectors carrying the Dirichlet and Neumann boundary conditions.
It is well known that choosing the same approximation space, as we do, for displacements and pressure, strong non-physical oscillations may appear in the approximation of the pressure field; this oscillations manifest themselves when the element size h is not small enough compared to the time step τ. The oscillatory behaviour of the FEM can be minimized if stabilized methods are used. For our numerical implementation of the poro-elasticity benchmark, we retained to be optimal the stabilized method proposed in [1], indeed, it consists on the addition of a perturbation term $\beta\Delta\dot{p}$, with parameter $\beta = h^2/4(\lambda + 2\mu)$, to the flow equation and therefore, according to the way of discretization we performed in (4.13), we only need to add in the mass matrix on the left side the term $\beta\mathbf{K}$ to stabilize our discretization.

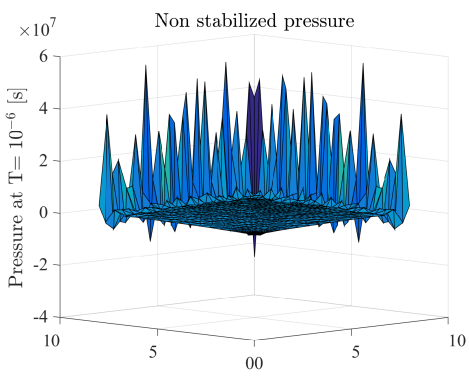

Figure 4.5: Pressure at T=10^{-6} seconds when no stabilization methods are applied (left) and when the perturbation term $\beta\Delta\dot{p}$, which brings to stabilization, is applied (right).

In Figure 4.5 we find the oscillations that manifests when no stabilization methods are applied and the stabilized solution achieved following [1] for $T = 10^{-6}$ seconds. Note that, to highlight the oscillatory behaviour, we amplified the values of the pressure in those nodes which do not refer to the Dirichlet boundary conditions of a $4 \cdot 10^3$ factor for both the imagines.
Now that all the aspects concerning the discretization of (4.11) and the stabilization of the problem have been clarified, we can present in Figure 4.7 the pictures of the displacements and of the pressure for $T = 100$; we have chosen such time cause it is big enough so that the system can reach a stationary configuration. In Figure 4.6 is shown how the mesh is deformed by the application of the Neumann boundary conditions, making use of 10^3 as amplification factor.
Last imagine, in Figure 4.8 is a vector plot of the velocity of the fluid in the porous material when a stationary state is reached (T= 100). We decide to

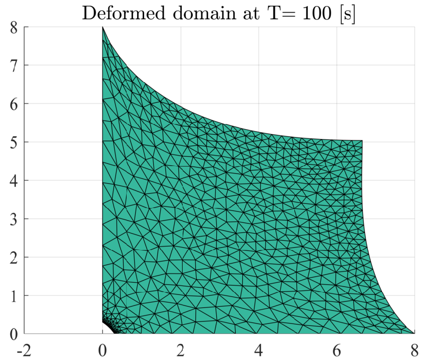

Figure 4.6: Domain's deformation due to the application of the Neumann boundary conditions. A scaling factor of 10^3 has been used to better show the final deformation result.

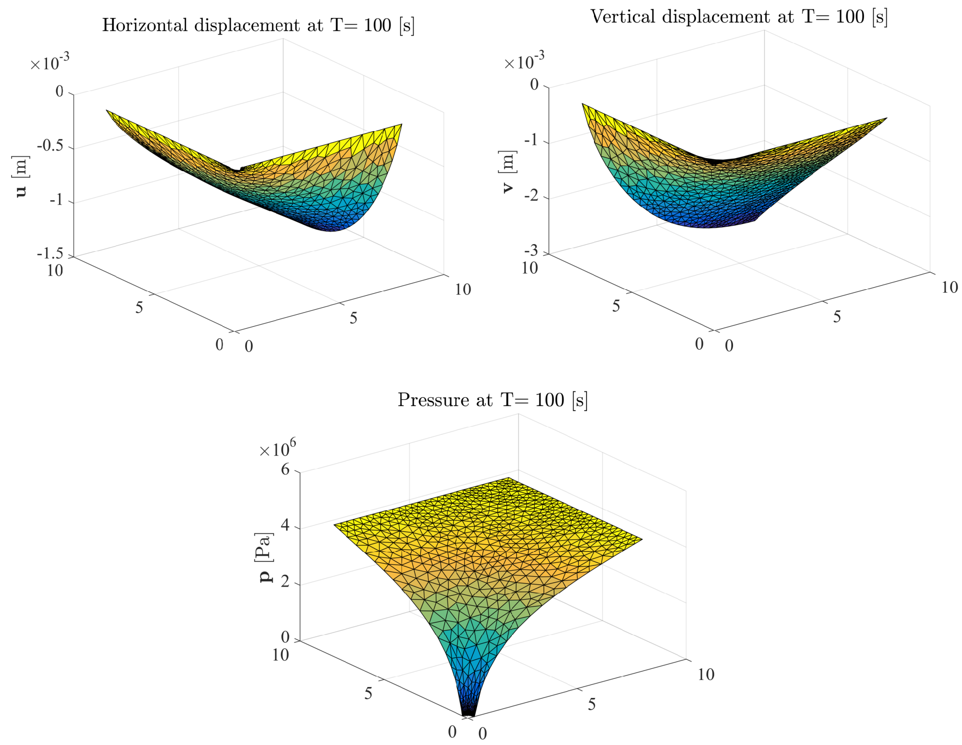

Figure 4.7: The two displacements and the pressure when T= 100.

show this picture because we think that it is very efficient in giving the intuition about the physical meaning of equations (4.11); indeed, remembering

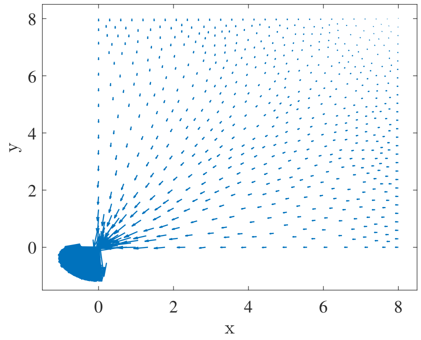

Figure 4.8: Vector plot of the fluid velocity $\mathbf{q}$ in the domain of the porous media.

that the velocity of the fluid $\mathbf{q}$ in a porous media is given by the Darcy's law

$$\mathbf{q} = -\frac{\kappa}{\eta}\nabla p \quad \text{in } \Omega; \tag{4.15}$$

we can observe how the application of the stresses in the right and top sides of the domain makes the fluid flow towards the hole in the bottom-left corner, where no stresses are applied. This is also possible since on the bottom and left sides a no flowing condition, $\nabla p = \mathbf{0}$, is imposed. Obviously, the length of the vectors is proportional to their modulus and it can be seen how it increases in the nearby of the circular cavity. This is coherent with the plot of the pressure in Figure 4.6, indeed it is almost flat in the proximity of the right and top sides while it strongly decreases going to the cavity where the pressure is zero.
Note that all the imagines are obtained considering the non linearity introduced in (4.12) for the permeability, in the following subsection we explain how this has been treated in the numerical implementation.

4.4.2 Reduced Basis implementation issues

As we declaimed before, the vector of the parameters $\boldsymbol{\mu}$ we considered for the poro-elasticity benchmark, is made by four components: the Young's modulus E, the Poisson's ratio ν, the storage capability S and finally the permeability κ, which is also the one where the non-linearity has been introduced (4.12). For each of these parameters a reference value has been considered, see Table 4.3, and the snapshots matrix has been constructed selecting the value of each parameter, such as the Poisson's ratio, in the following way

$$\nu = \nu_{\text{ref}} + \mathcal{U}(-1, 1)\nu_{\text{ref}}, \tag{4.16}$$

where $\mathcal{U}(-1, 1)$ is the uniform distribution between -1 and 1. The fluid dynamic viscosity is instead chosen to be fixed and equal to $\eta = 1 \cdot 10^{-3}$ [Pa·s]. Note that we are challenging a time dependent problem, therefore the snapshots matrix has to be constructed collecting also the solution at each step of the time discretization for a given $\boldsymbol{\mu}$. Then, from such matrix, we can build the reduced space, using for example POD or a greedy algorithm or even both of them distinguishing the time snapshots from the parameters snapshots as suggested in [21]. For the poro-elasticity problem we made use of the POD algorithm in the construction of the reduced space.

Reference parameters			
E_{ref}	ν_{ref}	S_{ref}	κ_{ref}
$3 \cdot 10^{10}$ [Pa]	$2 \cdot 10^{-1}$ [-]	$1 \cdot 10^{-6}$ [Pa^{-1}]	$1 \cdot 10^{-1}$ [m^2]

Table 4.3: Parameters' reference value of the poro-elasticity benchmark.

Last words we spent for this subsection concern how the non-linearity in the permeability has been treated. From the definition we gave of it (4.12), it is clear that κ changes both in space and in time, therefore the matrix in the right side of (4.13) is not any more constant since κ cannot be evaluated outside the matrix $\mathbf{K}$. As for the thermal diffusion, the solution are loops to solve the non-linearity, but, this time we can take advantage of the use of `ode23t` and compute the permeability directly inside `odefun`. In each step the matrix $\mathbf{K}$ is evaluated like the example in (4.8) where now $\mathbf{C}$ is the diagonal matrix which evaluates the permeability on each quadrature point.

4.4.3 Results

The comparison among high-fidelity, reduced and hyper-reduced solutions have been realized taking a test sample Ξ^{test} of 20 vectors of parameters $\boldsymbol{\mu}$, selected according expression (4.16); the modes used for all three the variables are 7 while the reduced rule is made of 500 nodes. The result we now show testifies the efficiency of FOCUSS also on this problem.
From Figure 4.9 we can extrapolate several observations. First, the error due to the combination of the two stages of reduction is not significantly influenced by the quadrature error introduced with the reduced rule; moreover the computational time is reduced. This means that FOCUSS succeed, as well as in the thermal diffusion, also in the poro-elasticity benchmark even if, compared the first application, here the errors are quite larger. This is probably due to the fact that the poro-elasticity is a quite more complex problem, which involves a mechanic-hydraulic coupling and a dynamic, therefore there is a necessity of a larger number of modes, compared to the thermal conduction, to achieve smaller error in the reduced basis approximation. It can be also observed how the quadrature error is much smaller for the two displacements variables than for the pressure; this is because the non-linearity in the permeability acts directly on the pressure and only secondarily, through the

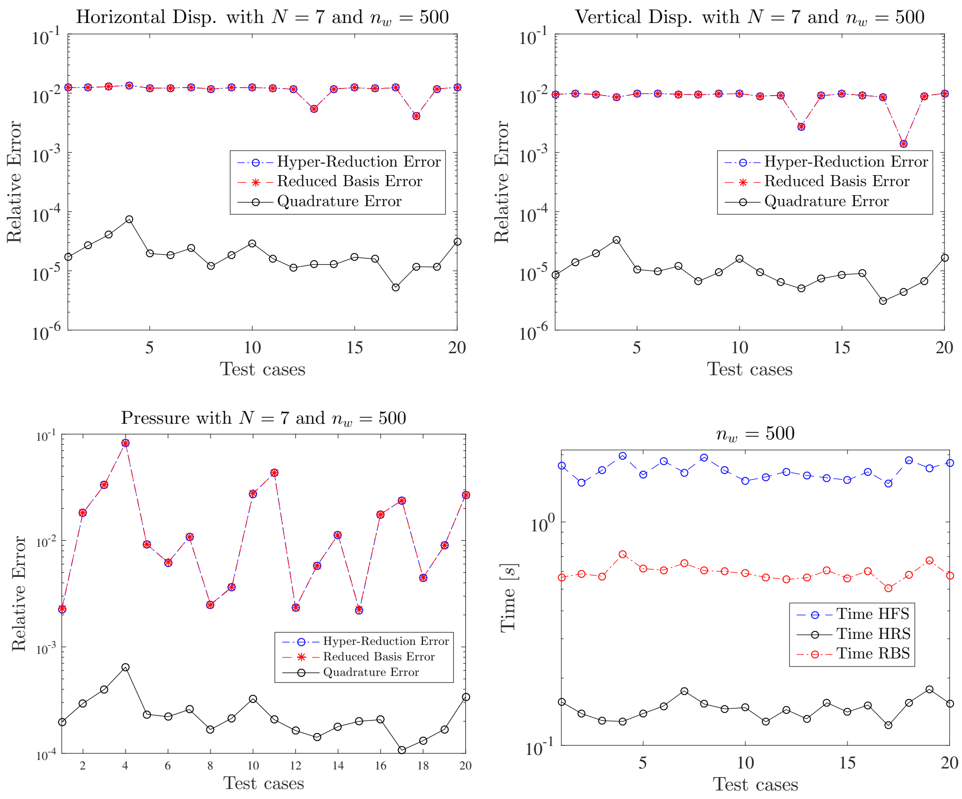

Figure 4.9: L^2-norm of the errors between the high-fidelity solution (HFS) and the hyper-reduced solution (HRS, with $n_w = 454$ quadrature points), the high-fidelity solution and the simple reduced basis solution (RBS, with $N = 7$ modes) and the hyper-reduced solution and the reduced basis solution (left). Computational time demanded for the high-fidelity, simple reduced basis and hyper-reduced solutions (right on the bottom). Results collected using FOCUSS for the poro-elasticity benchmark at time T$= 100$ s where the variables are: horizontal displacement $\mathbf{u}$, vertical displacement $\mathbf{v}$ and pressure $\mathbf{p}$.

coupling, on the two displacements.
For this reason, in Table 4.4, we only considered the pressure to proof that also reduced quadrature rule with less nodes than 500 can not substantially influence the total error, provided that the number of reduced basis functions is also taken lower than 7. All the errors have been evaluated at T$= 100$ seconds.

N	n_w	$\max_{\boldsymbol{\mu}\in\Xi^{\text{test}}} \dfrac{\lVert \mathrm{p}_N(\boldsymbol{\mu}) - \mathrm{p}_N^w(\boldsymbol{\mu})\rVert_{\mathrm{L}^2}}{\lVert \mathrm{p}_h(\boldsymbol{\mu})\rVert_{\mathrm{L}^2}}$	$\max_{\boldsymbol{\mu}\in\Xi^{\text{test}}} \dfrac{\lVert \mathrm{p}_h(\boldsymbol{\mu}) - \mathrm{p}_N^w(\boldsymbol{\mu})\rVert_{\mathrm{L}^2}}{\lVert \mathrm{p}_h(\boldsymbol{\mu})\rVert_{\mathrm{L}^2}}$
1	2	$8.22 \cdot 10^{-2}$	$6.50 \cdot 10^{-1}$
2	10	$1.91 \cdot 10^{-1}$	$8.47 \cdot 10^{-1}$
3	100	$5.90 \cdot 10^{-2}$	$2.15 \cdot 10^{-1}$
4	100	$4.51 \cdot 10^{-2}$	$8.88 \cdot 10^{-2}$
7	250	$9.50 \cdot 10^{-3}$	$2.71 \cdot 10^{-2}$

Table 4.4: Poro-elasticity benchmark at T= $100s$ for the pressure variable. The number of reduced-basis functions N, the number of reduced quadrature points n_w, the maximum difference in the reduced-basis approximation, over the test set Ξ^{test}, using the exact quadrature rule and the reduced quadrature, $\max_{\boldsymbol{\mu}\in\Xi^{\text{test}}} \frac{\lVert \mathrm{p}_N(\boldsymbol{\mu}) - \mathrm{p}_N^w(\boldsymbol{\mu})\rVert_{\mathrm{L}^2}}{\lVert \mathrm{p}_h(\boldsymbol{\mu})\rVert_{\mathrm{L}^2}}$, and the error in the hyper-reduction approximation, $\max_{\boldsymbol{\mu}\in\Xi^{\text{test}}} \frac{\lVert \mathrm{p}_h(\boldsymbol{\mu}) - \mathrm{p}_N^w(\boldsymbol{\mu})\rVert_{\mathrm{L}^2}}{\lVert \mathrm{p}_h(\boldsymbol{\mu})\rVert_{\mathrm{L}^2}}$.

Another interesting aspect, which can be observed in the poro-elasticity benchmark, is the evolution of the errors along the time, since until now we limited ourselves to evaluate the errors for the solutions at T= 100 seconds. Table 4.5 collects the results we found using the quadrature rule of 500 nodes.

The error in the hyper reduction approximation does not significantly change for the two displacements going forward in time, while it increases for the pressure. The "surprising" element is the quadrature error, that, for all the variables, is of several orders lower in the first instants of time to then grow up to the values we already observed in Figure 4.9; this testifies that the dynamic of the problem acts as an amplification factor for the quadrature error introduced with the reduced rule. The error of the quadrature is higher for the pressure variable than for the displacements in all the instants of time considered in the table; we have already commented this phenomena for $T = 100$ observing that the non-linearity directly acts on the pressure and only "indirectly" on the displacements.

We now can look at Figure 4.10, which is about the comparison of the performance of FOCUSS compared to the other cubature methods used until now. This time the POD technique has been applied before calling the hyper-

Evolution in time of the errors

T	$\max\limits_{\boldsymbol{\mu}\in\Xi^{\text{test}}} \dfrac{\lVert \mathrm{t}_N(\boldsymbol{\mu}) - \mathrm{t}_N^w(\boldsymbol{\mu})\rVert_{\mathrm{L}^2}}{\lVert \mathrm{t}_h(\boldsymbol{\mu})\rVert_{\mathrm{L}^2}}$			$\max\limits_{\boldsymbol{\mu}\in\Xi^{\text{test}}} \dfrac{\lVert \mathrm{t}_h(\boldsymbol{\mu}) - \mathrm{t}_N^w(\boldsymbol{\mu})\rVert_{\mathrm{L}^2}}{\lVert \mathrm{t}_h(\boldsymbol{\mu})\rVert_{\mathrm{L}^2}}$		
	u	**v**	**p**	**u**	**v**	**p**
10^{-6}	$1.38\cdot10^{-9}$	$1.38\cdot10^{-10}$	$1.50\cdot10^{-8}$	$3.11\cdot10^{-2}$	$2.26\cdot10^{-2}$	$3.82\cdot10^{-5}$
10^{-5}	$2.00\cdot10^{-8}$	$2.26\cdot10^{-9}$	$1.41\cdot10^{-7}$	$2.43\cdot10^{-2}$	$1.55\cdot10^{-2}$	$6.11\cdot10^{-4}$
10^{-3}	$7.00\cdot10^{-7}$	$1.32\cdot10^{-7}$	$1.23\cdot10^{-5}$	$3.24\cdot10^{-2}$	$1.92\cdot10^{-2}$	$5.80\cdot10^{-3}$
10^{-1}	$2.13\cdot10^{-5}$	$5.64\cdot10^{-6}$	$4.32\cdot10^{-4}$	$3.61\cdot10^{-2}$	$3.22\cdot10^{-2}$	$3.22\cdot10^{-2}$
10^{+1}	$5.95\cdot10^{-5}$	$2.36\cdot10^{-5}$	$1.30\cdot10^{-3}$	$4.40\cdot10^{-2}$	$1.30\cdot10^{-3}$	$1.66\cdot10^{-1}$
10^{+2}	$4.56\cdot10^{-5}$	$5.00\cdot10^{-6}$	$8.79\cdot10^{-4}$	$6.48\cdot10^{-2}$	$4.05\cdot10^{-2}$	$1.44\cdot10^{-2}$

Table 4.5: Poro-elasticity example: evolution along the time T of the maximum difference in the reduced-basis approximation, over the test set Ξ^{test}, using the exact quadrature rule and the reduced quadrature, $\max_{\boldsymbol{\mu}\in\Xi^{\text{test}}} \frac{\lVert \mathrm{t}_N(\boldsymbol{\mu})-\mathrm{t}_N^w(\boldsymbol{\mu})\rVert_{\mathrm{L}^2}}{\lVert \mathrm{t}_h(\boldsymbol{\mu})\rVert_{\mathrm{L}^2}}$, and of the error in the hyper-reduction approximation, $\max_{\boldsymbol{\mu}\in\Xi^{\text{test}}} \frac{\lVert \mathrm{t}_h(\boldsymbol{\mu})-\mathrm{t}_N^w(\boldsymbol{\mu})\rVert_{\mathrm{L}^2}}{\lVert \mathrm{t}_h(\boldsymbol{\mu})\rVert_{\mathrm{L}^2}}$. All the errors have been computed with the same reduced quadrature rule of $n_w = 500$.

reduction algorithms.

The plot referring to the maximum quadrature error over the test sample chosen shows different behaviours compared to the one of the thermal diffusivity. First of all, the trend is not the same for each method as in the thermal problem, in particular HEURISTIC and EIM work significantly better with a rule of 150 nodes than one of 200, a result not intuitive since the more nodes we use, the better errors we should obtain. We can explain this phenomena. Indeed, during the computation we have observed that with these two algorithms the problem stated to be ill-conditioned if a rule of 200 or more nodes was demanded; this is also confirmed looking at the plot of the computational time: as the number of nodes increases, HEURISTIC and EIM require an always higher computational time compared to FOCUSS and SIMPLEX. We conclude that the irregular trend shown by HEURISTIC and EIM is due to the perturbation caused by the instability that rises when a too high number of nodes is demanded. Moreover, it has also been observed

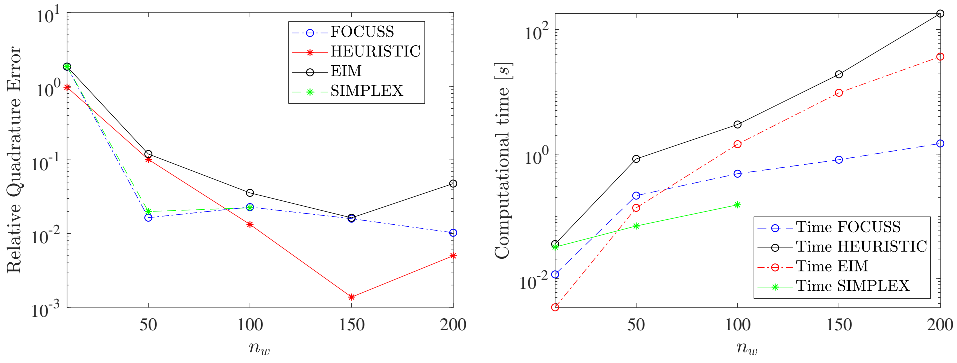

Figure 4.10: Comparison of hyper-reduction methods, over different n_w, concerning the maximum difference in the reduced-basis approximation, over the test set Ξ^{test}, using the exact quadrature rule and the reduced quadrature for the pressure variable, $\max_{\boldsymbol{\mu}\in\Xi^{\text{test}}} \frac{\left\|\mathrm{p}_N(\boldsymbol{\mu})-\mathrm{p}_N^w(\boldsymbol{\mu})\right\|_{\mathrm{L}_0^1}}{\left\|\mathrm{p}_h(\boldsymbol{\mu})\right\|_{\mathrm{L}_0^1}}$ (left). Computational time necessary in the offline stage for the recover of the sparse quadrature rule with n_w nodes (right). Results obtained with $N = 7$ reduced basis functions for the pressure.

here, as in the thermal diffusion, that EIM does not ensure the positiveness of the weights.

Another interesting result, which seems to confirm our statement concerning the inefficiency of the ℓ^1-norm minimization in the contest of sparse quadrature recovery, is the fact that during the computation of rules with more than 100 nodes, we have experimented the failure of the dual-SIMPLEX algorithm of Matlab. Indeed, even using very large tolerances for the convergence of the method, the call of the algorithm returned to us an error message when rules with 150 and 200 nodes were demanded.

In conclusion, from our tests on the poro-elasticity benchmark, it appears that FOCUSS is the best method among the used, concerning the trade-off between precision and computational time demanded in offline stage of sparse rule recovering.

The last results we present in this work aim to show what happens when the mesh is refined. We used the Gmsh option `refine by splitting` to create a new mesh, from the previous used, with $N = 3361$ nodes, $N_{ele} = 6344$ elements and $N_w = 25376$ quadrature points. Then, we built a new database containing the snapshots matrices with the objective of finding a new reduced base and a new quadrature rule of 500 nodes. At this point we tested our rule defined on the refined mesh on a test sample Ξ^{test} of 10 parameters vector $\boldsymbol{\mu}$. Figure 4.11 shows what we obtained.

The reduced rule produces quadrature errors of approximately an order greater than the ones produced when we used the course grid, but it is still lower than the one of the reduced basis approximation. Since the number of degree of freedom strongly increased compared to before, and the reduced rule is always made by the same number of nodes, the boost on the computational time given by the hyper-reduction is greater for these simulations.

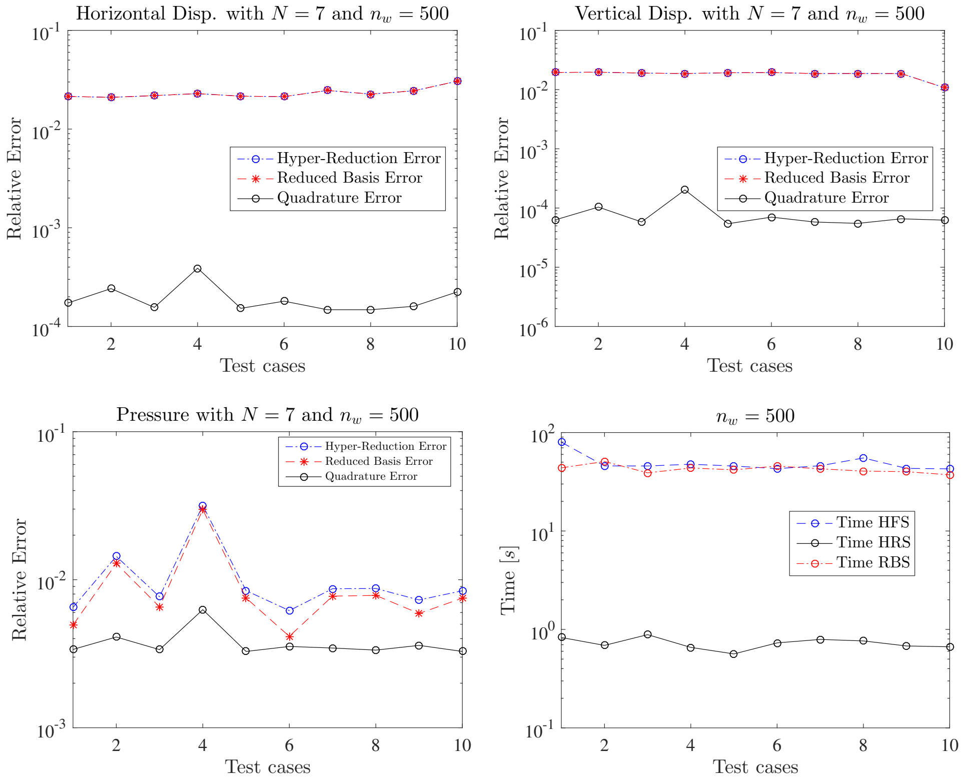

Figure 4.11: Refined grid. L^2-norm of the errors between: the high-fidelity solution (HFS) and the hyper-reduced solution (HRS, with $n_w = 454$ quadrature points), the high-fidelity solution and the simple reduced basis solution (RBS, with $N = 7$ modes) and the hyper-reduced solution and the reduced basis solution (left). Computational time demanded for the high-fidelity, simple reduced basis and hyper-reduced solutions (right on the bottom). Results collected using FOCUSS for the poro-elasticity benchmark at time T= 100 s where the variables are: horizontal displacement $\mathbf{u}$, vertical displacement $\mathbf{v}$ and pressure $\mathbf{p}$.

Closing remarks and possible future works

In this work we presented the problem of recovering the efficiency of the Reduced Basis method when dealing with non-linear PDEs and we mentioned all the methods which try to fix this problem like the empirical interpolation method (EIM), the empirical cubature method (HEURISTIC) or the dual-simplex based methods. We underlined that it is still not clear which method should be recommanded among the others since all of them show drawback, such as: the presence of negative weights, high computational effort demanded in the Offline stage or luck of precision in the approximation of the non-linearity.

For this reason we proposed a new hyper-reduction algorithm, which convergence to sparse quadrature rules has already been proved in [8], with the ambition of overcoming all the issues of the others hyper-reduction techniques. From all the tests we did on the three problems proposed, our implementation of FOCUSS always succeed in returning a sparse quadrature rule with positive weights. About the comparison with other hyper-reduction methods, for both FEM problems, our method was the best in the trade-off of the number of weights of the reduced rule against computational time to compute such rule, while in terms of precision the performance of the rules found with FOCUSS were lined with the ones of the other methods. It was very interesting the result found for the polynomial test problem where our algorithm was the only one able to recover the Gauss-Legendre quadrature rule, that is the more sparse and exact integrating rule for the problem considered.

To conclude this master thesis we would like to suggest possible extensions of this work. Most of all, to complete the work on FOCUSS, it would be necessary to have an estimation of the error introduced with the sparse quadrature rule recovered. Such estimate would also simplify the comparison with the other methods. Another aspect that could be interesting to investigate, at least more than what we did here, are the performances of the sparse rules when the number of nodes required is fixed but the mesh is refined time after time. With the poro-elasticity benchmark we tried to do this for one simple refinement of the mesh previously used and we observed that, fixed the number of nodes, the reduced quadrature loses an order of accuracy in relation to the high-fidelity solution. What we conjecture that would happen if the mesh is refined again and again is that the accuracy of the reduced rule would not be altered. This hypothesis comes from the fact that the refinement of the grid influences the high-fidelity solution in such a way that it is closer to the one that is the "true" solution of problem, while the variation of quadrature points in the reduced rule acts only on the error between the high-fidelity solution and its approximation via hyper-reduction; therefore, if the refinement produces changes in the high-fidelity solution of values of order less than the error carried by the reduced quadrature, we should not experiment significantly variations in the quadrature error when keeping the same number of nodes in the reduced rule.

Bibliography

[1] G. Aguilar, F. Gaspar, F. Lisbona, and C. Rodrigo. Numerical stabilization of biot's consolidation model by a perturbation on the flow equation. *Int. J. Numer. Meth. Engng.*, 75(11):1282–1300, August 2008.

[2] Radermacher Annika and Reese Stefanie. Pod-based model reduction with empirical interpolation applied to nonlinear elasticity. *Int. J. Numer. Meth. Engng*, 107(6):477–495, July 2018.

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

[4] Maxime Barrault, Yvon Maday, Ngoc Cuong Nguyen, and Anthony T. Patera. An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667–672, November 2004.

[5] S. Campbell and C. Meyer. *Generalized Inverses of Linear Transformations.* Society for Industrial and Applied Mathematics, July 2018.

[6] Farhat Charbel, Chapman Todd, and Avery Philip. Structure-preserving, stability, and accuracy properties of the energy-conserving sampling and weighting method for the hyper reduction of nonlinear finite element dynamic models. *Int. J. Numer. Meth. Engng*, 102(5):1077–1110, July 2018.

[7] S. Chaturantabut and D. C. Sorensen. Discrete empirical interpolation for nonlinear model reduction. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 4316–4321.

[8] I. F. Gorodnitsky and B. D. Rao. Sparse signal reconstruction from limited data using focuss: a re-weighted minimum norm algorithm. *IEEE Transactions on Signal Processing*, 45(3):600–616, Mar.

[9] Martin A. Grepl, Yvon Maday, Ngoc C. Nguyen, and Anthony T. Patera. Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. *ESAIM: M2AN*, 41(3):575–605, May 2007.

[10] J. A. Hernández, M. A. Caicedo, and A. Ferrer. Dimensional hyper-reduction of nonlinear finite element models via empirical cubature. *Computer Methods in Applied Mechanics and Engineering*, 313:687–722, January 2017.

[11] J. A. Hernández, J. Oliver, A. E. Huespe, M. A. Caicedo, and J. C. Cante. High-performance model reduction techniques in computational multiscale homogenization. *Computer Methods in Applied Mechanics and Engineering*, 276:149–189, July 2014.

[12] Ekkehard Holzbecher. *Poroelasticity Benchmarking for FEM on Analytical Solutions*. October 2013.

[13] K. Ito and S. S. Ravindran. A reduced-order method for simulation and control of fluid flows. *Journal of Computational Physics*, 143(2):403–425, July 1998.

[14] J. F. Murray and K. Kreutz-Delgado. An improved focuss-based learning algorithm for solving sparse linear inverse problems. In *Conference Record of Thirty-Fifth Asilomar Conference on Signals, Systems and Computers (Cat.No.01CH37256)*, volume 1, pages 347–351 vol.1, 4-7.

[15] C. Nguyen N., T. Patera A., and J. Peraire. A 'best points' interpolation method for efficient approximation of parametrized functions. *Int. J. Numer. Meth. Engng.*, 73(4):521–543, July 2007.

[16] Anthony T. Patera and Masayuki Yano. An lp empirical quadrature procedure for parametrized functions. *Comptes Rendus Mathematique*, 355(11):1161–1167, November 2017.

[17] T. A. Porsching. Estimation of the error in the reduced basis method solution of nonlinear equations. *Mathematics of Computation*, 45(172):487–496, 1985.

[18] C. Prud'homme, D. V. Rovas, K. Veroy, L. Machiels, Y. Maday, A. T. Patera, and G. Turinici. Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods. *Journal of Fluids Engineering*, 124(1):70–80, November 2001.

[19] Alfio Quarteroni. *Modellistica Numerica per Problemi Differenziali*, volume 9. January 2006.

[20] Alfio Quarteroni, Andrea Manzoni, and F. Negri. *Reduced basis methods for partial differential equations: An introduction.* January 2015.

[21] Alfio Quarteroni, Gianluigi Rozza, and Andrea Manzoni. Certified reduced basis approximation for parametrized partial differential equations and applications. *Journal of Mathematics in Industry*, 1(1):3, June 2011.

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

[23] Gianluigi Rozza, D. B. Phuong Huynh, and Andrea Manzoni. Reduced basis approximation and a posteriori error estimation for stokes flows in parametrized geometries: roles of the inf-sup stability constants. *Numerische Mathematik*, 125(1):115–152, September 2013.

[24] D. Ryckelynck. Hyper-reduction of mechanical models involving internal variables. *Int. J. Numer. Meth. Engng.*, 77(1):75–89, July 2018.

[25] Ernest K. Ryu and Stephen P. Boyd. Extensions of gauss quadrature via linear programming. *Foundations of Computational Mathematics*, 15(4):953–971, August 2015.

[26] Steven S An, Theodore Kim, and Doug James. *Optimizing Cubature for Efficient Integration of Subspace Deformations*, volume 27. December 2009.

[27] L. A. W. R. E. N. C. E. SIROVICH. Turbulence and the dynamics of coherent structures part i: Coherent structures. *Quarterly of Applied Mathematics*, 45(3):561–571, 1987.

[28] Mountaka Souley and Alain Thoraval. *Nonlinear mechanical and poromechanical analyses : comparison with analytical solutions.* October 2011.

[29] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[30] Masayuki Yano and Anthony T. Patera. An lp empirical quadrature procedure for reduced basis treatment of parametrized nonlinear pdes. *Computer Methods in Applied Mechanics and Engineering*, March 2018.