MOX-Report No. 67/2022

A greedy MOR method for the tracking of eigensolutions to parametric elliptic PDEs

Alghamdi, M. M.; Boffi, D.; Bonizzoni, F.

A greedy MOR method for the tracking of eigensolutions to parametric elliptic PDEs

Moataz M. Alghamdi[1], Daniele Boffi[1,2], and Francesca Bonizzoni[3]

[1]Applied Mathematics and Computational Sciences (AMCS), King Abdullah University of Science and Technology. Thuwal, 23955-6900, Kingdom of Saudi Arabia
[2]Department of Mathematics "F. Casorati", University of Pavia. Via Ferrata 1, 27100 Pavia, Italy
[3]MOX - Department of Mathematics, Politecnico di Milano. Via Bonardi 9, 20133 Milano, Italy

Abstract

In this paper we introduce an algorithm based on a sparse grid adaptive refinement, for the approximation of the eigensolutions to parametric problems arising from elliptic partial differential equations. In particular, we are interested in detecting the crossing of the hypersurfaces describing the eigenvalues as a function of the parameters.

The a priori matching is followed by an a posteriori verification, driven by a suitably defined error indicator. At a given refinement level, a sparse grid approach is adopted for the construction of the grid of the next level, by using the marking given by the a posteriori indicator.

Various numerical tests confirm the good performance of the scheme.

Key words: Eigenvalue problem; Parameter-dependent partial differential equation; Model reduction; Eigenvalue matching; A posteriori error indicator; Sparse grid.
AMS subject classification: 65N25, 65N30, 35B30, 78M34, 35P15, 62K25

1 Introduction

Many engineering applications require the knowledge of resonance frequencies of the considered structure. Prime examples are vibration problems in mechanical engineering, where the vibration of buildings or bridges at their natural frequencies might cause damage and structural failure.

During the last decades, computation power has substantially increased. Nevertheless, the computation effort required for the solution of large-scale eigenproblems is still considerable. The situation gets more difficult when manufacturing imperfections and geometric or material variability are included in the mathematical model as parameters or random

fields. In particular, when the solution of the same eigenproblem is of interest for many different values of the parameters, the direct computation of eigenpairs by means of standard numerical techniques entails an unaffordable computational effort.

Model Order Reduction (MOR) methods aim at reducing the overall computational effort by producing a surrogate of the parameter-to-eigenpair map, which is accurate and at the same time fast to be evaluated. They develop in two phases: several snapshots of eigenpairs (corresponding to an appropriate set of parameter values) are computed during the offline phase, and used to construct the surrogate, which is then evaluated at any new parameter value during the online phase.

A very popular ROM method developed during the last few decades is the Reduced Basis (RB) method (see, e.g., [24, 13]) which constructs the surrogate by projection onto the set of precomputed snapshots selected either adaptively by means a greedy algorithm or by Proper Orthogonal Decomposition (POD).

Up to now, a tremendous effort has been made in developing ROM for source problems, whereas model reduction for parametric/stochastic eigenproblems is still a largely unexplored field. The way was paved by the pioneering work [19], where an RB approach is proposed to approximate the smallest eigenvalue. This methodology has been further developed to deal with more eigenvalues in [22]. A component-based RB method for eigenvalue problems is proposed in [26] and an a posteriori estimator for eigenvalues is studied in [15]. All the previously mentioned contributions do not cover the case of multiple eigenpairs. Instead, in [14], an a posteriori error bound for multiple eigenvalues (but not eigenvectors) is studied under the assumption of affine parametric dependence of the eigenproblem. Finally, in [10] a greedy RB method for both affine and non-affine parametric eigenproblems is proposed, with a focus on the smallest (single) eigenpair, only. The same task (namely, the approximation of the first eigenpair) in the context of stochastic eigenproblems has been solved in [12] and [2] by means of the Stochastic Galerkin and Stochastic Collocation method, respectively.

The aim of the present paper is the development of an algorithm to match (and interpolate) snapshots of eigenpairs of symmetric problems as the parameter varies in a given p-dimensional subset of interest $\mathcal{M} \subset \mathbb{R}^p$, and as the eigenvalues lie in a fixed window of interest $I_\lambda \subset \mathbb{R}$. In particular, the issues connected with multiple eigenvalues as well as crossings of the hypersurfaces described by the eigenvalues as the parameter varies in $\mathcal{M}$ are thoroughly analyzed.

The parameter space is sampled adaptively, following a two-phase procedure. First, a suitably adapted version of the a priori matching proposed in [21] is applied. Numerical examples (see Section 4.1) show that this technique might produce an incorrect matching. Hence, we have developed a novel a posteriori indicator based on the orthogonality of the snapshots of eigensolutions, which drives the adaptive sampling. The algorithm is first presented on a one-dimensional (in the parametric space) case, and then extended to the high-dimensional setting by means of hierarchical locally refined sparse grids.

It is worth mentioning that the a priori matching that we use is connected with MOR techniques developed in a different framework, namely, the parametric-in-frequency Helmholtz boundary value problem. In particular, we mention [3, 4, 5, 9], where rational-based surrogates for the Helmholtz solution map are constructed. Indeed, the roots of the denominator of the surrogate are approximations to the resonances of the Helmholtz

2

problem, which in turn are eigenvalues of the corresponding elliptic problem. A matching strategy in the same spirit as [21] and the a priori matching of the present paper is studied in [28]. Moreover, a mode tracking method for the parametrized Maxwell eigenvalue problem is presented in [11, 29, 27, 16].

Our work is the first step toward a deep understanding of Uncertainty Quantification for stochastic eigenvalue problems arising from elliptic PDEs with stochastic diffusion coefficients. This field of research, even though relatively unexplored, is extremely important in various application areas.

The paper is organized as follows. In Section 2 we describe the problem of interest. Section 3 is dedicated to the adaptive algorithm: first we give a general overview of the main steps of the proposed algorithm, details on all the steps then follow. In Section 4 we present both one and two dimensional results to validate the proposed strategy. Conclusions are finally drawn in Section 5.

2   Setting of the Problem

Let us consider the Hilbert triplet

$$V \subset H \simeq H' \subset V'$$

and a parameter space $\mathcal{M} \subset \mathbb{R}^d$. It is out of the aims of this work to identify the most general assumptions on the parameter space. Very often in the applications we have in mind, it is a tensor product of intervals (like a hypercube). In any case in what follows we will need that it is connected and that it supports an initial grid with neighboring points as described later in this section.

For each $\mu \in \mathcal{M}$, we consider two symmetric and bilinear forms

$$a(\cdot, \cdot; \mu) : V \times V \to \mathbb{R},$$
$$b(\cdot, \cdot; \mu) : H \times H \to \mathbb{R}.$$

Furthermore, we make the following assumptions

$$V \text{ compact in } H,$$
$$a(\cdot, \cdot; \mu) \text{ elliptic in } V \quad \forall \mu \in \mathcal{M},$$
$$b(\cdot, \cdot; \mu) \text{ equivalent to the inner product in } H \quad \forall \mu \in \mathcal{M}.$$

More general assumptions could be made. However, the features of our strategy are better described in this simpler setting.

Our aim is to approximate the solutions of the following parametric eigenvalue problem: for all $\mu \in \mathcal{M}$, find real eigenvalues $\lambda(\mu)$ and non-vanishing eigenfunctions $u(\mu) \in V$ such that

$$a(u, v; \mu) = \lambda(\mu) b(u, v; \mu) \quad \forall v \in V. \tag{2.1}$$

Our assumptions ensure that the problem is associated with a compact solution operator so that all eigenvalues $\{\lambda_j(\mu)\}_{j=1}^{\infty}$ correspond to finite-dimensional eigenspaces. In particular, we are interested in detecting the behavior of the hypersurfaces defined by $\lambda_j(\mu)$

in the region $\mathcal{M} \times \mathbb{R}$. These hypersurfaces may intersect, leading in general to multiple eigenvalues at one point of intersection. Indeed, the situation can be very complicated when the dimension d of the parametric space gets large. In the simplest case, $d = 1$, we are dealing with intersections of curves.

To better define our problem, we restrict the range of the eigenvalues we are interested in to an interval $I_\lambda = [\lambda_{\min}, \lambda_{\max}]$, also referred to as *window of interest*. Consequently, we only examine the hypersurfaces in the region $\mathcal{M} \times I_\lambda$. This implies in particular that hypersurfaces can enter or exit this region of interest when the corresponding eigenvalues cross the values of $\lambda_{\min}$ or $\lambda_{\max}$. It follows then that the number of eigenvalues considered for μ_1 and μ_2 in $\mathcal{M}$ can be different from each other when $\mu_1 \neq \mu_2$.

Problem (2.1) is discretized by finite elements (FEs). That is, we consider a finite dimensional subspace $V_h \subset V$ and for all $\mu \in \mathcal{M}$ we consider the matrix generalized eigenvalue problem: find real eigenvalues $\lambda_h(\mu)$ and non-vanishing eigenfunctions $u_h(\mu) \in V_h$ such that

$$a(u_h, v; \mu) = \lambda_h(\mu)b(u_h, v; \mu) \quad \forall v \in V_h. \tag{2.2}$$

This will be considered as our *high-fidelity* solution.

3 Description of the adaptive algorithm

Before examining the details of our approach, we give a general overview of the main steps taken to track the behavior of the hypersurfaces in the region $\mathcal{M} \times I_\lambda$ described by the varying eigenvalues. We assume that we are given a grid in the parametric space $\mathcal{M}$ and that we have computed the eigenvalues in the interval I_λ as well as the corresponding eigenfunctions for each point of the grid.

The first phase consists in the **a priori matching** of the eigenvalues of each pair of neighboring parameters μ_i and μ_k where the indices i and k are defined based on the used grid. The matching is performed by considering, in a suitable sense, how close the eigenvalues and the eigenfunctions are to each other. This phase is prone to error in particular if the distance between μ_i and μ_k is large with respect to the variability of the eigenvalues. The a priori phase is followed by an **a posteriori verification** of the matching. We introduce a suitable a posteriori indicator that is based on the orthogonality of the eigenfunctions. This phase aims to confirm whether the a priori matching was performed correctly or not. If not, the corresponding interval is marked for refinement. Finally, a sparse grid approach is used to drive the **refinement strategy**, leading to an adaptive procedure that is terminated when a suitable stopping criterion is met.

The three phases of our adaptive strategy are executed using three interrelated algorithms. Algorithm 3 describes the global refinement procedure. In turn, this algorithm relies on two additional algorithms. The first one, Algorithm 1, describes the local a priori matching procedure while the second one, Algorithm 2, enforces the local a posteriori test.

3.1 The a Priori Matching

The FE method (2.2) leads to an algebraic problem as follows: find $\lambda(\mu) \in \mathbb{R}$ and $u(\mu) \in \mathbb{R}^N$ with $u(\mu) \neq 0$ such that $A(\mu)u(\mu) = \lambda(\mu)B(\mu)u(\mu)$, where $A(\mu)$ and $B(\mu)$ are matrices

in $\mathbb{R}^{N \times N}$ where N is the dimension of our finite element space. $A(\mu)$ and $B(\mu)$ are symmetric and positive definite for all values of $\mu \in \mathcal{M}$. To avoid heavy notation, we denote discrete quantities without indicating the space mesh index h (λ instead of λ_h, etc.).

We consider two different parameter values μ_i and μ_k that are the endpoints of what we are going to call from now on a *local subinterval*. For these two points, we generate the two sets of FE eigenpairs $\{(\lambda_j(\mu_i), u_j(\mu_i))\}_{j=1}^{n_i}$, $\{(\lambda_\ell(\mu_k), u_\ell(\mu_k))\}_{\ell=1}^{n_k}$. Note that the values n_i and n_k may be different from each other in particular since we are looking for all eigenvalues within the window of interest I_λ. We assumed that the bilinear form $b(\cdot, \cdot, \mu)$ is equivalent to the scalar product in H for all $\mu \in \mathcal{M}$; we denote the associated norm by

$$\|v\|_{b,\mu} = b(v, v, \mu)^{1/2}$$

The eigenfunctions are normalized with respect to this norm so that

$$\|u_j(\mu_i)\|_{b,\mu_i} = \|u_\ell(\mu_k)\|_{b,\mu_k} = 1.$$

Definition 3.1. The cost matrix $D^{(i,k)}$ associated with the local subinterval with endpoints μ_i and μ_k has size $n_i \times n_k$ and entries

$$D_{j,\ell}^{(i,k)} := w_1|\lambda_j(\mu_i) - \lambda_\ell(\mu_k)| + w_2 \min\{\|u_j(\mu_i) - u_\ell(\mu_k)\|_{b,\bar{\mu}}, \|u_j(\mu_i) + u_\ell(\mu_k)\|_{b,\bar{\mu}}\}, \quad (3.1)$$

where $w_1, w_2 \in \mathbb{R}_+$ are weights and $\bar{\mu}$ is a fix parameter value between μ_i and μ_k.

Remark 3.2. A similar definition to (3.1) can be found in [21]. However, some modifications are necessary in the present context, since the normalization of the eigenfunctions doesn't necessarily imply the same choice of sign. Hence we have to compare both the sum and the difference of corresponding eigenfunctions.

Once we have introduced the cost matrix, we want to minimize its entries by solving the following optimization problem: find the permutation $\sigma^\star = (\sigma_1, \ldots, \sigma_{\bar{n}}) \in (1, \ldots, \bar{n})!$ such that

$$\sigma^\star := \mathrm{argmin}_{\sigma \in (1,\ldots,\bar{n})!} \sum_{\alpha=1}^{n_i} \sum_{\beta=1}^{n_k} D_{\sigma_\alpha, \sigma_\beta}^{i,k}, \quad (3.2)$$

with $\bar{n} := \min\{n_i, n_k\}$. Between the several options available to compute the solution of (3.2), we adopt the the so-called Hungarian Algorithm [17]. The permutation solution is then used to reorder the eigensolutions so that there is a one-to-one correspondence between the matched eigensolutions

The structure of our code is reported in the following Algorithm 1, where we assume, without loss of generality, that $n_i \geq n_k$, i.e., $\bar{n} = n_k$.

3.2 The a Posteriori Verification

After the a priori matching, we introduce an a posteriori verification phase, which is based on the following projection matrix.

Algorithm 1 Local a priori matching

Require: $\mu_i, \mu_k \in \mathcal{M}$, $\{(\lambda_j(\mu_i), u_j(\mu_i))\}_{j=1}^{n_i}$, $\{(\lambda_\ell(\mu_k), u_\ell(\mu_k))\}_{\ell=1}^{n_k}$, $w_1, w_2 \in \mathbb{R}_+$
Ensure: Reordered eigenpairs $\{(\lambda_j^\star(\mu_i), u^\star(\mu_i))\}_{j=1}^{n_i}$, $\{(\lambda_\ell^\star(\mu_k), u_\ell^\star(\mu_k))\}_{\ell=1}^{n_k}$
  1: Compute $D^{(i,k)} \in \mathbb{R}^{n_i \times n_k}$ with weights w_1, w_2 ▷ See (3.1)
  2: Find $\boldsymbol{\sigma}^\star$ solution to (3.2)   ▷ Hungarian algorithm
  3: Set $(\lambda_j^\star(\mu_i), u^\star(\mu_i)) = (\lambda_j(\mu_i), u_j(\mu_i))$ for $j = 1, \ldots, n_i$
  4: Set $(\lambda_\ell^\star(\mu_k), u_\ell^\star(\mu_k)) = (\lambda_{\sigma_\ell^\star}(\mu_k), u_{\sigma_\ell^\star}(\mu_k))$ for $\ell = 1, \ldots, n_k$

Definition 3.3. The projection matrix $\Pi^{(i,k)}$ associated with the local subinterval with endpoints μ_i and μ_k has size $n_i \times n_k$ and its entries are given by

$$\Pi_{j,\ell}^{(i,k)} = |b(u_j(\mu_i), u_\ell(\mu_k), \bar{\mu})|, \tag{3.3}$$

with $\bar{\mu}$ being a fixed parameter value between μ_i and μ_k.

Ideally, if the matching was performed correctly, the projection matrix should be close to diagonal: two matching eigenfunctions should be similar to each other and two non-matching eigenfunctions should be close to orthogonal with respect to the bilinear form b. In order to check whether the projection matrix is close to diagonal, we make use of a positive tolerance value t_π. This is used to truncate $\Pi^{(i,k)}$ — inside a loop over $j = 1, \ldots, \min\{n_i, n_k\}$ — in accordance with lines 4-13 of Algorithm 2. Let r_1, r_2 be the vectors containing the non-zero elements of the j-th column and j-th row of $\Pi^{(i,k)}$, respectively (line 14). If both r_1 and r_2 contain just one element, the a priori matching is considered correct. Instead, if their length differs, the interval is marked for refinement (lines 16-19). Checking the orthogonality of eigenfunctions is a good stopping criterion in general, but might fail when we are close to multiple eigensolutions. In such a case, the orthogonality between distinct eigenfunctions depends on the solver and is not immediate to check in practice. For this reason, we introduce a second positive tolerance value t_λ that is responsible for verifying if two (or more) eigenvalues belong to a cluster. This scenario corresponds in Algorithm 2 to the case where r_1 and r_2 have the same length, larger than 1. Lines 20-25 introduce a specific definition for when multiple eigensolutions are to be considered as one cluster of indistinguishable eigenfunctions. The local subinterval is marked for refinement when it fails the t_λ stopping criterion.

Remark 3.4 (Choice of the tolerances t_π, t_λ). Note that in the limit $t_\pi \to 0$, the projection matrix $\Pi^{(i,k)}$ is diagonal. On the other hand, the truncation process will leave the projection matrix unchanged as $t_\pi \to 1$, possibly leading to an infinite loop over the refinement level (see Section 3.4). There is then a threshold between between t_π being large enough to capture potential errors in matching choices, and small enough to minimize the number of refinements. The selection of the optimal value for t_π becomes more delicate as the dimension of the parameter space gets larger.

A similar balance must hold for t_λ. When there is considerable overlap between two (or more) eigenfunctions, this will translate into non-zero off-diagonal elements of $\Pi^{(i,k)}$. This can lead to a very large number of refinements in order for the subinterval to be certified, unless t_λ is chosen such that those overlapping eigenfunctions are identified as a

cluster. However, if t_λ is chosen to be too large, then wrong matching choices of orthogonal eigenfunctions will be certified by the a posteriori estimator because such eigenfunctions will be incorrectly considered indistinguishable. This premature termination can lead to wrong results.

Algorithm 2 Local a posteriori verification

Require: $\mu_i, \mu_k \in \mathcal{M}$, $\{(\lambda_j(\mu_i), u_j(\mu_i))\}_{j=1}^{n_i}$, $\{(\lambda_\ell(\mu_k), u_\ell(\mu_k))\}_{\ell=1}^{n_k}$, $t_\pi, t_\lambda \in \mathbb{R}_+$
Ensure: if_refinement = 0 or if_refinement = 1
 1: Set if_refinement=0
 2: Compute $\Pi^{(i,k)} \in \mathbb{R}^{n_i \times n_k}$ ▷ See Equation (3.3)
 3: **for** $j = 1$ to $\texttt{min}(n_i, n_k)$ **do**
 4:      **for** $\ell = 1$ to n_k **do**
 5:          **if** $\Pi_{j,j}^{(i,k)} \geq \Pi_{j,\ell}^{(i,k)} + t_\pi$ **then**
 6:              Set $\Pi_{j,\ell}^{(i,k)} = 0$ ▷ Truncate $\Pi_{j,:}^{(i,k)}$ up to tolerance t_π
 7:          **end if**
 8:      **end for**
 9:      **for** $\ell = 1$ to n_i **do**
10:          **if** $\Pi_{j,j}^{(i,k)} \geq \Pi_{\ell,j}^{(i,k)} + t_\pi$ **then**
11:              Set $\Pi_{\ell,j}^{(i,k)} = 0$ ▷ Truncate $\Pi_{:,j}^{(i,k)}$ up to tolerance t_π
12:          **end if**
13:      **end for**
14:      Let $r_1 = \texttt{find}(\Pi_{j,:}^{(i,k)})$, $r_2 = \texttt{find}(\Pi_{:,j}^{(i,k)})$ ▷ Indices of the non-zero elements
15:      **if** $(\texttt{length}(r_1)) > 1$ or $(\texttt{length}(r_2)) > 1$ **then**
16:          **if** $\texttt{length}(r_1) \neq \texttt{length}(r_2)$ **then**
17:              if_refinement = 1   ▷ Mark $[\mu_i, \mu_k]$ for refinement
18:              **break**
19:          **end if**
20:          Let $\alpha_{j,\gamma_1}^i := \frac{|\lambda_j(\mu_i) - \lambda_{\gamma_1}(\mu_i)|}{\lambda_j(\mu_i)}$ for $\gamma_1 \in r_1$
21:          Let $\alpha_{j,\gamma_2}^k := \frac{|\lambda_j(\mu_k) - \lambda_{\gamma_2}(\mu_k)|}{\lambda_j(\mu_k)}$ for $\gamma_2 \in r_2$
22:          **if** $\texttt{max}_{\gamma_1 \in r_1, \gamma_2 \in r_2}\{\alpha_{j,\gamma_1}^i, \alpha_{j,\gamma_2}^k\} > t_\lambda$ **then**   ▷ No cluster is identified
23:              if_refinement = 1   ▷ Mark $[\mu_i, \mu_k]$ for refinement
24:              **break**
25:          **end if**
26:      **end if**
27: **end for**

3.3 The Sparse Grid-based Adaptive Sampling

In the present contribution we aim at a refinement strategy that performs well also when the dimension d of our problem is large. To mitigate the curse of dimensionality, it is essential to pay particular attention to the way the grid of the parameter space $\mathcal{M}$ is refined. One possibility is to use locally-refined sparse grids, in the spirit of what was proposed in [1, 21]. In particular, the refinement step described in Section 3.4 relies on
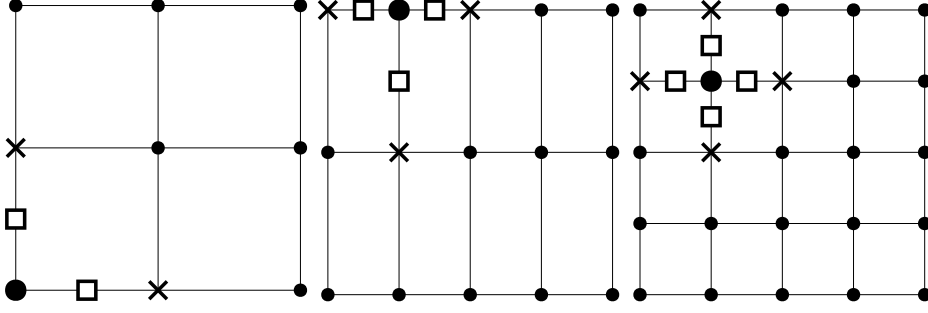
Figure 3.1: Forward points (squares) and neighbours (crosses) of $\mu = (-1, -1) \in \Gamma(1,1)$ (left); $\mu = (-\frac{1}{2}, 1) \in \Gamma(2,1)$ (middle); $\mu = (-\frac{1}{2}, \frac{1}{2}) \in \Gamma(2,2)$ (right).

some notions and features related to sparse grids. For the readers' convenience, we recall them here.

The main ideas of our sparse grid approach are better explained, without loss of generality, when $\mathcal{M} = [-1, 1]^2$. With small modifications, the case $\mathcal{M} = [a, b] \times [c, d]$, for $a, b, c, d \in \mathbb{R}$ can be handled. Moreover, the following discussion can be easily extended to the high-dimensional framework, i.e., for $d \geq 3$.

Let us define the sequence $\{\Gamma(m)\}_{m \in \mathbb{N}_0}$ of nested sets of points in $[-1, 1]$ as follows:

$$\Gamma(m) := \begin{cases} \{0\} & \text{if } m = 0, \\ \{2^{1-m} j\}_{j=-2^{m-1}}^{2^{m-1}} & \text{if } m > 0. \end{cases} \tag{3.4}$$

By tensor product, we get grids of points in $\mathcal{M}$. In particular, given a two-dimensional multi-index $\mathbf{m} = (m_1, m_2) \in \mathbb{N}_0^2$, the corresponding two-dimensional grid is defined as

$$\Gamma(\mathbf{m}) = \Gamma(m_1) \times \Gamma(m_2) = \{(\alpha_1, \alpha_2), \ \alpha_k \in \Gamma(m_k), \ k = 1, 2\}.$$

Let $\mu = (\mu_1, \mu_2) \in \Gamma(\mathbf{m})$ be given, and assume that both its entries are fractions in lowest terms. We define the set of *forward points* $\mathcal{V}(\mu)$ of μ as

$$\mathcal{V}(\mu) := \{(\mu_1 \pm 2^{-m_1}, \mu_2), (\mu_1, \mu_2 \pm 2^{-m_2})\} \cap \mathcal{M}, \tag{3.5}$$

and the set of neighbours $\mathcal{U}(\mu)$ of μ as

$$\mathcal{U}(\mu) := \{(\mu_1 \pm 2^{-(m_1-1)}, \mu_2), (\mu_1, \mu_2 \pm 2^{-(m_2-1)})\} \cap \mathcal{M}. \tag{3.6}$$

Note that both sets $\mathcal{V}(\mu)$ and $\mathcal{U}(\mu)$ contain up to 2^d points for any $d \geq 2$. Some examples are depicted in Figure 3.1.

3.4 The Refinement Strategy

We start with an initial partition $P^{(0)}$ or the parameter space $\mathcal{M}$. The procedure described in Algorithm 3 describes how to go from the ℓ-th partition $P^{(\ell)}$ to the next partition $P^{(\ell+1)}$ containing $P^{(\ell)}$.

Knowing that $P^{(\ell)} = P^{(\ell-1)} \cup P_\delta^{(\ell)}$, the refinement strategy takes into account only the points in $P_\delta^{(\ell)}$. This is essential in order to mitigate the curse of dimensionality when the

8

value of d is large. On the other hand, we do not need to consider the points in $P^{(\ell)} \setminus P_\delta^{(\ell)}$ since we know from the previous level that the a posteriori indicator confirmed the a priori matching there.

Each element (parameter) in $P_\delta^{(\ell)}$ has a collection of neighboring points (see equation (3.6)). This defines a collection of local subintervals on which we apply Algorithms 1 and 2, leading to either a marking of the subinterval for further refinement or not. The output of one iteration of this algorithm defines the output of a **level**. We perform this algorithm iteratively for increasing values of the level $\ell \in \mathbb{N}_0$ (with the convention $P^{(-1)} = \emptyset$, so that $P^{(0)} = P_\delta^{(0)}$) until no extra refinements for any subinterval within the current level take place.

Algorithm 3 Refinement Step

Require: Level $\ell \in \mathbb{N}_0$, initial grid $P^{(\ell)} = P^{(\ell-1)} \cup P_\delta^{(\ell)}$ with $P^{(\ell-1)} = \{\mu_s^{(\ell-1)}\}_{s=1}^{N_{\ell-1}}$ and $P_\delta^{(\ell)} = \{\mu_s^{(\ell,\delta)}\}_{s=1}^{N_{\ell,\delta}}$, and eigenpairs $\Lambda_\delta^{(\ell)} := \{(\lambda_j(\mu_s^{(\ell,\delta)}), u_j(\mu_s^{(\ell,\delta)}))\}_{j=1}^{n_s}$

Ensure: Refined grid $P^{(\ell+1)}$ and eigenpairs $\Lambda_\delta^{(\ell+1)}$

1: Set $P_\delta^{(\ell+1)} = \emptyset$, $\Lambda_\delta^{(\ell+1)} = \emptyset$
2: **for** $s = 1 : N_{\ell,\delta}$ **do** $\triangleright$ Loop on the points in $P_\delta^{(\ell)}$
3:     **for** $\nu_{(r,s)} \in \mathcal{V}(\mu_s^{(\ell,\delta)})$ **do** $\triangleright$ Loop on the neighbours of $\mu_s^{(\ell,\delta)}$ (3.6)
4:         $\Lambda_s^\star := \{(\lambda_j(\mu_s^{(\ell,\delta)}), u_j(\mu_s^{(\ell,\delta)}))\}_{j=1}^{n_s} \leftarrow \{(\lambda_j(\mu_s^{(\ell,\delta)}), u_j(\mu_s^{(\ell,\delta)}))\}_{j=1}^{n_s}$
5:         $\Lambda_r^\star := \{(\lambda_j^\star(\nu_{(r,s)}), u_j^\star(\nu_{(r,s)}))\}_{j=1}^{n_s} \leftarrow \{(\lambda_j(\nu_{(r,s)}), u_j(\nu_{(r,s)}))\}_{j=1}^{n_r}$ $\triangleright$ See
   Algorithm 1
6:         if_refinement = a-posteriori check on $\Lambda_s^\star$, $\Lambda_r^\star$ $\triangleright$ See Algorithm 2
7:         **if** if_refinement = 1 **then**
8:             $\bar{P}_\delta^{(\ell+1)} \leftarrow \bar{\nu}_{(r,s)}$ $\triangleright$ $\bar{\nu}_{(r,s)}$ is the corresponding forward point of $\mu_s^{(\ell,\delta)}$
9:             Compute $\bar{\Lambda} := \{(\lambda_j(\bar{\nu}_{(r,s)}), u_j(\bar{\nu}_{(r,s)}))\}_{j=1}^{n_r}$
10:            $\Lambda_\delta^{(\ell+1)} \leftarrow \bar{\Lambda}$
11:        **end if**
12:    **end for**
13: **end for**
14: Set $P^{(\ell+1)} = P^{(\ell)} \cup P_\delta^{(\ell+1)}$

---

4 Numerical results

Section 4.1 is devoted to a numerical example illustrating the behavior of the two local procedures (a priori matching and a posteriori verification). The global refinement algorithm is discussed in one- and two- dimensional numerical examples in Sections 4.2 and 4.3.

In all the presented numerical experiments we consider the following setting: let $\Omega = [0,1]^2$ be the physical domain and let $V = H^1(\Omega)$ and $H = L^2(\Omega)$ endowed with the usual inner products and norms. For all $\mu \in \mathcal{M} \subset \mathbb{R}^d$ we look for eigenpairs $(\lambda(\mu), u(\mu)) \in$

$\mathbb{R}^+ \times H^1(\Omega)$ such that

$$\begin{cases} \nabla \cdot (c(\mu)\nabla u(\mu)) = \lambda(\mu)u(\mu) & \text{in } \Omega, \\ u(\mu) = 0 & \text{on } \partial\Omega, \end{cases} \qquad (4.1)$$

where $c(\mu)$ is a matrix with size 2×2 and positive definite for all possible values of the parameter μ. Integrating by parts, we find the weak formulation of (4.1) of the form (2.1) with the bilinear forms

$$a(w, v; \mu) := \int_\Omega c(\mu)\nabla w \cdot \nabla v \, dx,$$

$$b(w, v; \mu) := \int_\Omega wv \, dx.$$

The FE method on a sufficiently refined mesh is employed to numerically approximate the eigenvalues and eigenfunctions at fixed values of the parameter $\mu \in \mathcal{M}$. All the computations are performed in MATLAB on a laptop with four cores (eight logical processors), 16 GB of RAM. Furthermore, we make use of the Partial Differential Equation toolbox [20] and Sparse Grids Kit [23].

4.1 Operations on a subinterval

In the first numerical test we take $\mathcal{M} = [0.4, 1]$,

$$c(\mu) = \begin{pmatrix} \mu^{-2} & 1 \\ 1 & 0.7^{-2} \end{pmatrix}, \qquad (4.2)$$

and $I_\lambda = [0, 270]$. We first detail the local steps of the proposed algorithm, namely, the a priori matching and the a posteriori verification, on the subinterval with endpoints $\mu_1 = 0.4$ and $\mu_2 = 0.7$. The FE method at μ_1 (respectively μ_2) delivers four (respectively nine) eigenvalues in I_λ (and corresponding eigenvectors), given by

$$\lambda(\mu_1) = (80.8, \ 137.9, \ 230.6, \ 265.9)^\top,$$

$$\lambda(\mu_2) = (38.2, \ 81.1, \ 109.7, \ 129.4, \ 188.6, \ 189.9, \ 214.8, \ 260.9, \ 261.9)^\top.$$

The 4×9 cost matrix (with weights $w_1 = 1$, $w_2 = 200$) reads as follows:

$$D = \begin{bmatrix} \textcolor{red}{57.7} & 282.2 & 311.6 & 318.8 & 390.6 & 391.5 & 415.6 & 462.8 & 463.3 \\ 381.9 & \textcolor{red}{189.4} & 202.6 & 290.1 & 333.1 & 324.8 & 359.5 & 396.6 & 406.6 \\ 473.2 & 431.9 & 403.4 & 288.7 & \textcolor{red}{204.8} & 323.0 & 220.3 & 313.1 & 310.7 \\ 509.9 & 359.2 & 290.3 & 418.4 & 360.0 & 345.3 & 333.8 & \textcolor{red}{278.3} & 286.7 \end{bmatrix}$$

To minimize the total cost, namely, to solve the optimization problem (3.2), we employ the Hungarian algorithm, whose output is the permutation $\sigma^\star = (1, \ 2, \ 5, \ 8)$ (the entries of the cost matrix identified by $\sigma^\star$ are depicted in red). As a consequence, the vectors $\{\lambda_j(\mu_1), \ j = 1, \dots, 4\}$ and $\{\lambda_\ell(\mu_2), \ \ell = \sigma_1^\star, \dots, \sigma_4^\star\}$ are matched as follows:

$$\begin{array}{ccc} 80.8 & \leftrightarrow & 38.2 \\ 137.9 & \leftrightarrow & 81.1 \\ 230.6 & \leftrightarrow & 188.6 \\ 265.9 & \leftrightarrow & 260.9 \end{array}$$

10

and the vector $\{\lambda_\ell(\mu_2),\ \ell = 1, \ldots, 9\}$ is reordered accordingly:

$$\lambda^\star(\mu_2) = (38.2,\ 81.1,\ 188.6,\ 260.9,\ 109.7,\ 129.4,\ 189.9,\ 214.8,\ 261.9)^\top.$$

The a priori matching is then followed by the a posteriori verification. Here, we start with the 4×9 projection matrix.

$$\Pi = \begin{bmatrix} 0.997 & 0.000 & 0.001 & 0.000 & 0.000 & 0.070 & 0.000 & 0.018 & 0.014 \\ 0.000 & 0.778 & 0.000 & 0.052 & 0.622 & 0.000 & 0.071 & 0.000 & 0.000 \\ 0.030 & 0.000 & 0.663 & 0.000 & 0.000 & 0.564 & 0.000 & 0.481 & 0.007 \\ 0.000 & 0.617 & 0.000 & 0.051 & 0.778 & 0.000 & 0.092 & 0.000 & 0.000 \end{bmatrix}$$

We fix $t_\pi = 0.21$, and we enter the loop over j (line 3 in Algorithm 2. For $j = 1$, the truncation conditions in line 5, 10 are fulfilled, hence all the off-diagonal entries of the first row and column of Π are truncated to 0. For $j = 2$, instead, the same process identifies two non-zero elements both in the second row and in the second column of Π:

$$\Pi = \begin{bmatrix} 0.997 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.778 & 0 & 0 & 0.622 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.663 & 0.000 & 0.000 & 0.564 & 0.000 & 0.481 & 0.007 \\ 0 & 0.617 & 0.000 & 0.051 & 0.778 & 0.000 & 0.092 & 0.000 & 0.000 \end{bmatrix}$$

As a consequence, the first matching choice $80.8 \leftrightarrow 38.2$ is certified but not the second matching $137.9 \leftrightarrow 81.1$. Here, the second eigenvalue at μ_1, namely 137.9, is permitted to be matched to both the second (81.1 since $\Pi_{2,2} = 0.778$) and fifth (109.7 since $\Pi_{2,5} = 0.622$) eigenvalues at μ_2. Furthermore, the second eigenvalue at μ_2, 81.1, can be matched to the second and fourth (265.9 since $\Pi_{4,2} = 0.617$) eigenvalues at μ_1.

We then check whether the second and fourth eigenvalues at μ_1 form a cluster given the fixed tolerance $t_\lambda = 0.001$. A quick computation yields the following

$$\frac{|\lambda_2(\mu_1) - \lambda_4(\mu_1)|}{\lambda_2(\mu_1)} > 0.001$$

and similar results hold at μ_2 for the second and fifth eigenvalues. Thus, these eigenvalues are distinguishable from each other and no eigenvalue clusters are identified. Consequently, the a posteriori verification does not confirm the results of the a priori matching, and the subinterval is not certified and is marked for further refinement.

4.2   Full 1D example

The main purpose of this section is to show how refinement in a full 1D parametric domain takes place. We continue using the same set-up presented in Section 4.1.

For the sake of comparison, we numerically compute the reference solution (see Figure 4.1). This is obtained by applying the a priori matching (Algorithm 1) on the uniform grid of the parametric space $\mathcal{M}$ containing 129 points. The matching information is then propagated from left to right along all the 128 subintervals of $\mathcal{M}$.

In contrast, we apply the adaptive refinement algorithm presented in Section 3. The output is the (coarse) adapted sparse grid of $\mathcal{M}$ containing enough points to detect all
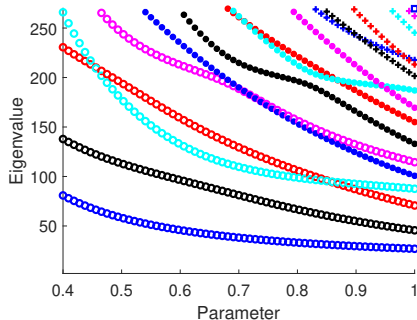
11

Figure 4.1: Reference solution.

the features of the reference solution. Figure 4.2 depicts the evolution of the parametric grid as the level increases, whereas in each subfigure of Figure 4.3 the result of the local checks are represented: matched eigenvalues are plotted using the same color and marker; if the projection matrix suggests that more than one matching is possible, the possible matchings are highlighted by means of black dashed lines. We detail now the level-by-level procedure, which is summarized in Table 4.1.

Level 0 The initial grid is $P^{(0)} = \{0.4, 0.7, 1\}$ (see the blue points in Figure 4.2). In particular, the local checks (both the a priori matching and the a posteriori verification) are performed on the two local subintervals $[0.4, 0.7]$ and $[0.7, 1]$, which are both marked for further refinement.

Level 1 The grid at level one contains five points $P^{(1)} = \{0.4, 0.55, 0.7, 0.85, 1\}$ (see the black dots in Figure 4.2) and it is given by $P^{(1)} = P^{(0)} \cup P^{(1)}_\delta$, with $P^{(1)}_\delta = \{0.55, 0.85\}$. Following Algorithm 3, the local checks are performed on the four subintervals $[0.4, 0.55]$, $[0.85, 1]$, $[0.55, 0.7]$ and $[0.7, 0.85]$. Only the last two subintervals are marked for further refinement.

Level 2 The grid at level 2 is $P^{(2)} = \{0.4, 0.55, 0.625, 0.7, 0.775, 0.85, 1\}$ (see the red dots in Figure 4.2) and it is given by $P^{(2)} = P^{(1)} \cup P^{(2)}_\delta$, with $P^{(2)}_\delta = \{0.625, 0.775\}$. Local checks are then performed on the following subintervals: $[0.55, 0.625]$, $[0.625, 0.7]$, $[0.7, 0.775]$, and $[0.775, 0.85]$. Only one subinterval is marked for further refinement, namely $[0.775, 0.85]$.

Level 3 The grid at level 3 is $P^{(3)} = \{0.4, 0.55, 0.625, 0.7, 0.775, 0.8125, 0.85, 1\}$ (see the magenta dots in Figure 4.2) and it is given by $P^{(3)} = P^{(2)} \cup P^{(3)}_\delta$, with $P^{(3)}_\delta = \{0.8125\}$. The local checks on the local subintervals $[0.775, 0.8125]$ and $[0.8125, .085]$ are performed, and none of the two is marked for refinement. As a consequence, the adaptive algorithm terminates.

It is worth mentioning that, even though the subinterval $[0.775, 0.85]$ was marked for refinement at level 2, the a priori matching was correct. To explain the extra refinement, we examine the projection matrix associated with this subinterval, more precisely the 2×2

Error By Level				
Level	Total no. of points	No. of wrongly matched points	No. of subintervals	No. of uncertified subintervals
0	3	2	2	2
1	5	3	4	2
2	7	0	4	1
3	8	0	2	0

Table 4.1: Error table for the 1D example.

sub-matrix with entries corresponding to the black and cyan eigenvalues:

$$\begin{bmatrix} \Pi_{7,7} & \Pi_{7,8} \\ \Pi_{8,7} & \Pi_{8,8} \end{bmatrix} = \begin{bmatrix} 0.721 & 0.682 \\ 0.680 & 0.714 \end{bmatrix}.$$

The off-diagonal terms $\Pi_{7,8}$ and $\Pi_{8,7}$ are non-zero due to a non-negligible overlap between the corresponding eigenfunctions, which are depicted in Figure 4.4. Therefore, we can identify two causes for the a posteriori verification to suggest the refinement: (i) the ordering of the eigenvalues proposed by the a priori matching is incorrect; (ii) the eigenfunctions are not orthogonal. The second situation typically occurs in the presence of small gaps between the eigenvalue hypersurfaces. Recall that lines 20-25 in Algorithm 2 are devoted to case (ii). In this example, no clusters are identified, and the algorithm consequently terminates.

Once the adaptive algorithm has terminated, the collected information can be exploited to construct a surrogate for the map $\mu \mapsto \lambda(\mu)$. Between the various possibilities, we propose to construction this surrogate simply by piecewise linear interpolation. The result is depicted in Figure 4.5.
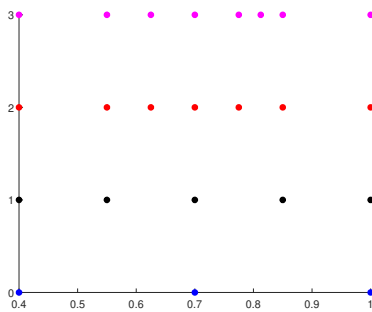

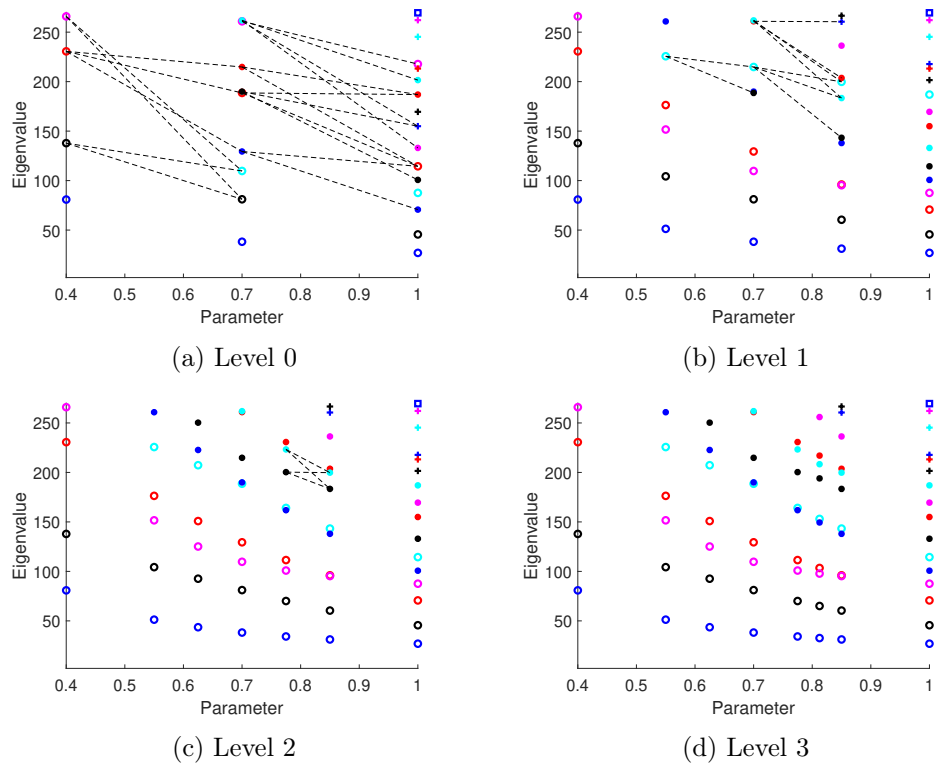Figure 4.2: Evolution of parametric grid (x-axis) as the level increases (y-axis).

(a) Level 0

(b) Level 1

(c) Level 2

(d) Level 3

Figure 4.3: Visual summary of the projection matrices.

14

(a) $\lambda = 200.3$, $\mu = 0.775$

(b) $\lambda = 223.23$, $\mu = 0.775$

(c) $\lambda = 183.369$, $\mu = 0.85$

(d) $\lambda = 199.786$, $\mu = 0.85$
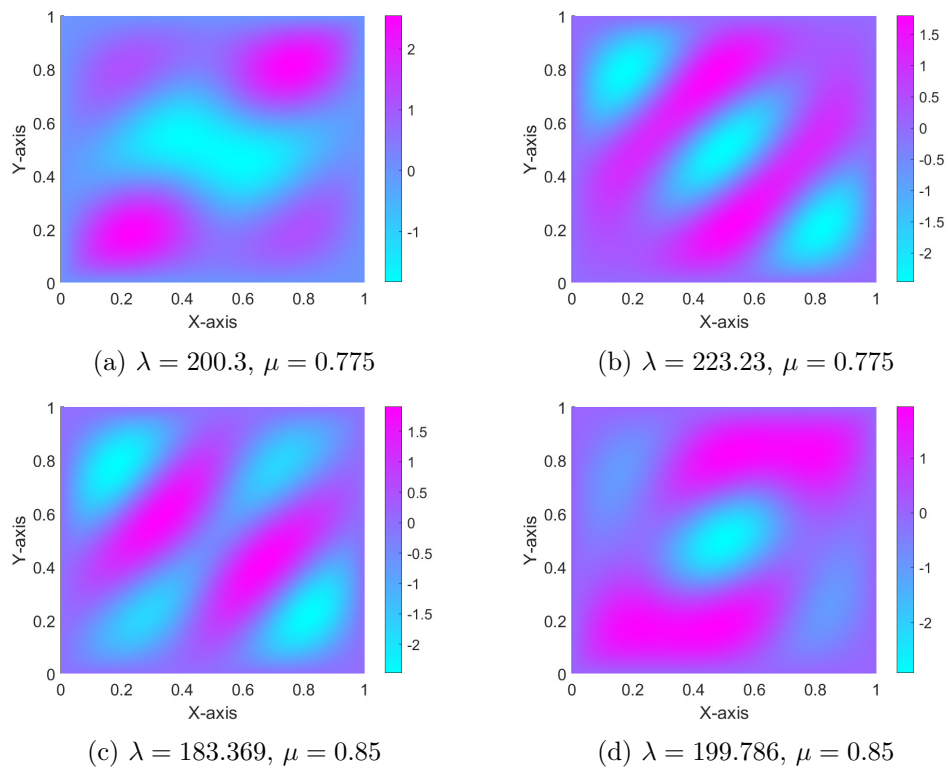
Figure 4.4: Comparison between four non-orthogonal eigenfunctions.

(a) Level 0

(b) Level 1

(c) Level 2

(d) Level 3
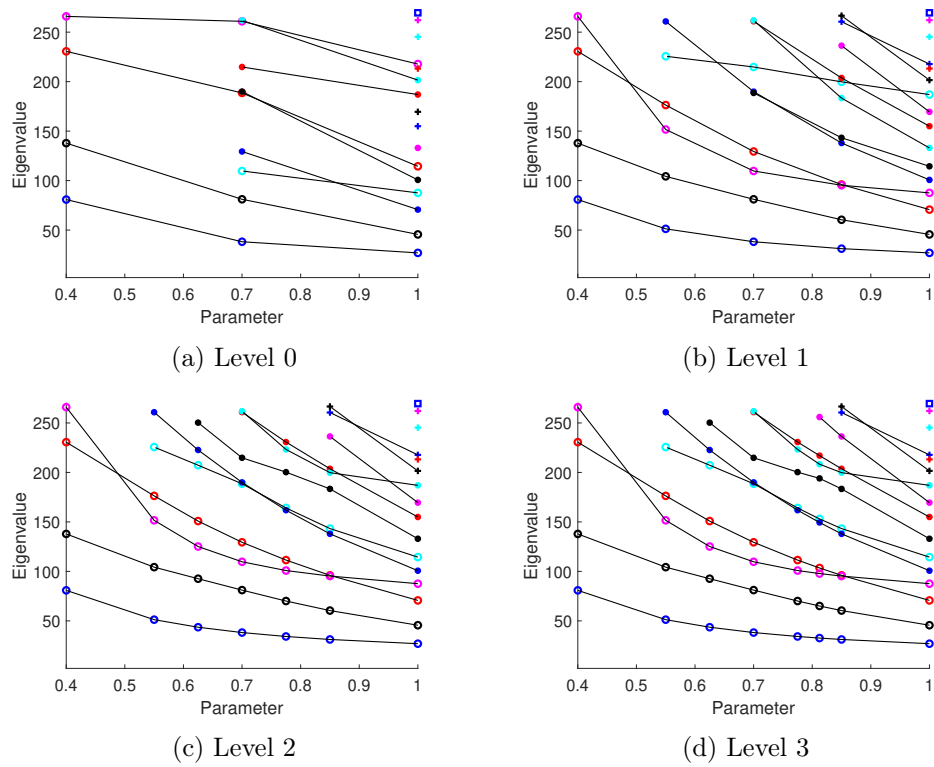
Figure 4.5: Output of each level for the 1D example.

16

4.3   A 2D example

In this numerical test we consider the two dimensional parameter space $\mathcal{M} = [0.8, 1.05] \times [0.8, 1.05]$, and we take the (positive definite) diffusion matrix

$$c(\mu) = c(\mu_1, \mu_2) = \begin{pmatrix} \mu_1^{-2} & 0.8\mu_2^{-1} \\ 0.8\mu_2^{-1} & \mu_2^{-2} \end{pmatrix}.$$

The computation of the reference solution relies on the uniform tensor product grid of $\mathcal{M}$ containing 129 points (see Figure 4.6 for the eigenvalue hypersurfaces). For eigenvalue problems depending on two parameters, the computation of the reference solution is still affordable. However, for increasing dimension d of the parametric space, such an approach entails prohibitive computational costs, becoming out of reach and extremely more expensive than the proposed sparse grid-based approach. In the following discussion, we focus on the two important features of the reference solution, namely, crossings and small gaps of the eigenvalue hypersurfaces.
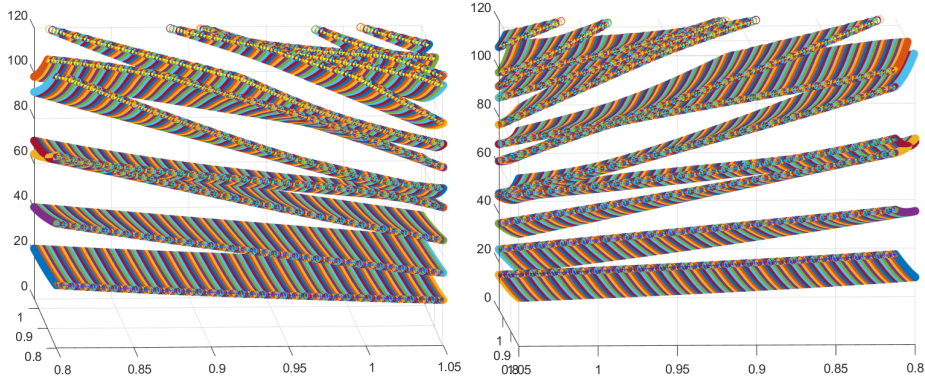
Figure 4.6: Reference solution for the 2D example; All hypersurfaces.

Given the initial grid $P^{(0)}$ being the 3×3 uniform lattice, and the tolerances $t_\pi = 0.57$, $t_\lambda = 0.015$, we let the adaptive algorithm run. The level-by-level output is displayed in Figure 4.7. The parametric grid produced by the adaptive algorithm is clearly non-uniform, and two regions in the parametric space can be recognized. At the left-half of $\mathcal{M}$, where no refinement happens, the eigenvalue hypersurfaces are well separated except for the crossing of the 3rd and 4th ones (see Figure 4.8 (c)), which is however correctly identified by the a priori matching and subsequently certified by the a posteriori verification already at level 0. On the other hand, all the added points lie in the right-half of $\mathcal{M}$, even though no crossings are present. This refinement pattern is due to small gaps between the eigenvalue hypersurfaces (see Figure 4.8 (a)-(b)). In particular, the extra grid points added at each level are meant to help the a posteriori test certify the identity a priori matching.

Table 4.2 summarizes the level-by-level information. We can observe that, from level 5 on, five points are added to the grid, even though all the eigenpairs were correctly matched. These points serve the purpose of allowing the algorithm to fully certify all subintervals, and produce a correct solution after termination. An important feature to notice is that some refinements can lead to an increase in the error (represented by the number of wrongly

17

(a) Level 0

(b) Level 1

(c) Level 2
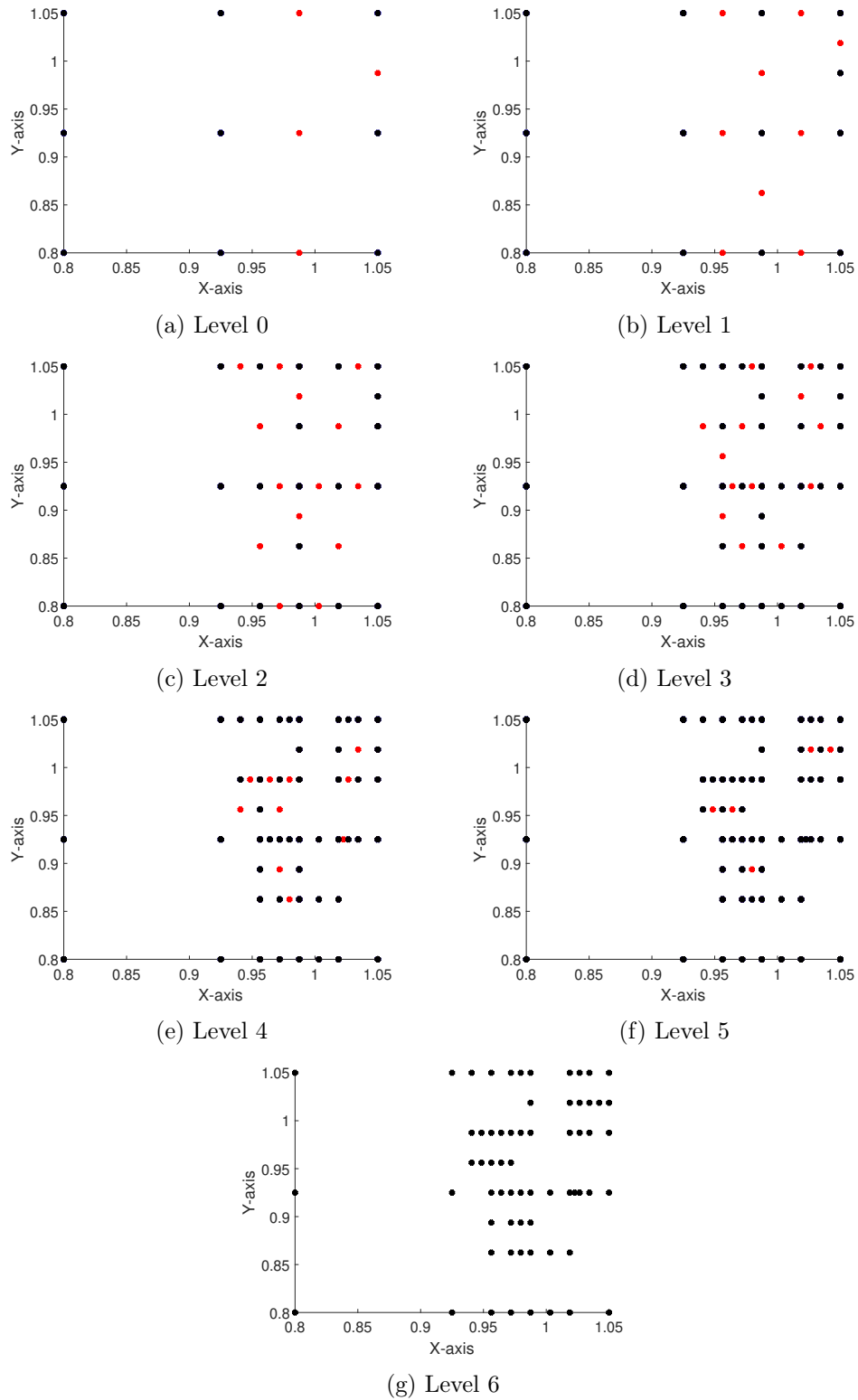
(d) Level 3

(e) Level 4

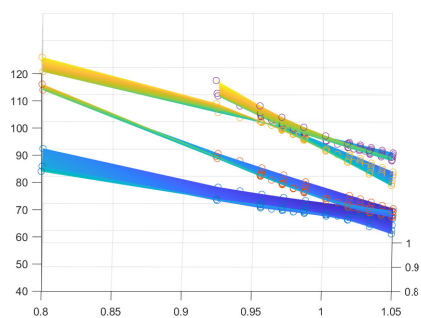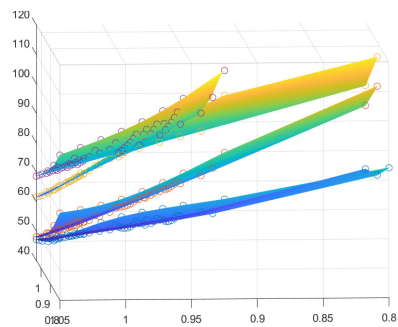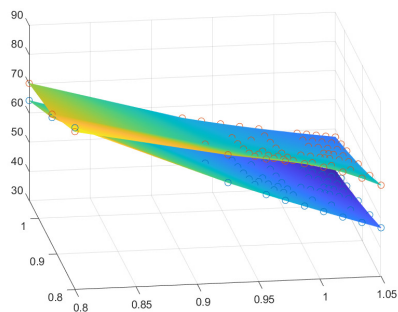(f) Level 5

(g) Level 6

Figure 4.7: Evolution of grid in the 2D example.

(a) Hypersurfaces 4,7; Side 1

(b) Hypersurfaces 4,7; Side 2

(c) Hypersurfaces 3,4

Figure 4.8: Final solution for the 2D example; Detailed view.

Error By Level				
Level	Total no. of points	No. of wrongly matched points	No. of subintervals	No. of uncertified subintervals
0	9	7	12	4
1	13	2	10	9
2	22	1	22	11
3	36	4	36	10
4	49	2	39	4
5	59	0	34	2
6	64	0	20	0

Table 4.2: Error table for the 2D example.

matched eigenpairs). This happens for example at level 3. Additional refinements at each level generate smaller subintervals. While increased accuracy is obtained, this doesn't translate into a continuous decay in the error. A correct solution is only guaranteed when all subintervals are certified.

We conclude the section by bringing to the surface one last issue, namely how to propagate the matching information. In the one dimensional case, this issue was easily solved by defining the propagation direction from left to right. In the two dimensional case, the way to proceed is not clear anymore, and it becomes even more complicated for $d > 2$.

Our technique relies on the creation of a path connecting all the points of the adapted grid. Such a path (which in principle is not guaranteed to be unique) must fulfill good properties (e.g., connected, no cycles are allowed) since the information must propagate distinctively. This step is performed in the code making use of the MATLAB commands `minspantree` and `shortestpath`.

As an example, we display the initial path corresponding to level 0 in Figure 4.9. We generate this path by applying the `minspantree` function on the graph

$$G = \texttt{graph}([1\ 1\ 2\ 4\ 2\ 3\ 5\ 4\ 5\ 6\ 7\ 8], [2\ 4\ 5\ 5\ 3\ 6\ 6\ 7\ 8\ 9\ 8\ 9]).$$

We then find our one-dimensional sub-paths using `shortestpath`. In this case, the three sub-paths all start at (0.8, 0.8) and end at (0.8, 1.05), (0.925, 1.05) and (1.05, 1.05) respectively.
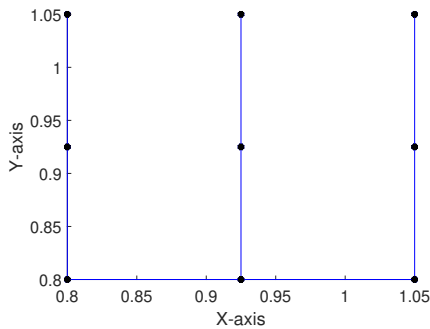

Figure 4.9: Initial path.

5 Conclusions

The present paper introduces a novel adaptive algorithm for the numerical treatment of parametric eigenvalue problems arising from elliptic partial differential equations. It is composed of two phases: locally, we look at a specific subinterval and we decide (by means of an a priori matching followed by an a posteriori verification) whether to mark it for refinement or not; globally, we perform a sparse grid-based refinement step, which delivers an adapted grid in the parameter space refined where needed in order to detect the features and crossings of the eigenvalue hypersurfaces. Finally, a surrogate for the parameter-to-eigenvalue

map is constructed simply by piecewise linear interpolation. Notably, the construction of a surrogate for the parameter-to-eigenfunction (or parameter-to-eigenvector) map is more delicate, and it is left for future investigations.

Even though the algorithm is written in an arbitrary dimension of the parameter space, numerical examples are performed in 1D and 2D, only, with the scope of attesting the validity and verifying the performances of the proposed numerical scheme. Higher-dimensional numerical tests will be presented in a forthcoming contribution.

This work paves the way towards the treatment of stochastic eigenvalue problems, i.e., eigenvalue problems arising from elliptic partial differential equations with random coefficients. In recent years a huge effort has been made in the study of uncertainty quantification (UQ) techniques for the source problem, and various methods have been developed (we mention, e.g., the Monte Carlo method [25], non-intrusive and Galerkin methods [18] and perturbation methods [6, 7, 8]. However, the field of stochastic eigenvalue problems is still quite unexplored, and we believe that the combined use of UQ techniques together with the algorithm proposed here represents a promising way to go.

Acknowledgements

References

[1] Alsayyari, F. and Perkó, Z. and Lathouwers, D. and Kloosterman, J.L. A nonintrusive reduced order modelling approach using Proper Orthogonal Decomposition and locally adaptive sparse grids. *Journal of Computational Physics* (2019) **399**, pp. 108912

[2] Andreev, R. and Schwab, C. Sparse Tensor Approximation of Parametric Eigenvalue Problems. *Numerical Analysis of Multiscale Problems*, Springer Berlin Heidelberg (2012), pp. 203–241˘

[3] Bonizzoni, F. and Nobile, F. and Perugia, I. Convergence analysis of Padé approximations for Helmholtz frequency response problems. *ESAIM: Mathematical Modelling and Numerical Analysis* (2018) **52**: 4, pp. 1261 – 1284

[4] Bonizzoni, F. and Nobile, F. and Perugia, I. and Pradovera, D. Fast Least-Squares Padé approximation of problems with normal operators and meromorphic structure. *Mathematics of Computation* (2020) **89**, pp. 1229–1257

[5] Bonizzoni, F. and Nobile, F. and Perugia, I. and Pradovera, D. Least-Squares Padé approximation of parametric and stochastic Helmholtz maps. *Advances in Computational Mathematics* (2020) **46**: 46

[6] Bonizzoni, F. and Nobile, F. Regularity and sparse approximation of the recursive first moment equations for the lognormal Darcy problem. *Computers & Mathematics with Applications* (2020) **80**: 12, pp. 2925-2947

[7] Bonizzoni, F. and Nobile, F. and Kressner, D. Tensor train approximation of moment equations for elliptic equations with lognormal coefficient. *Computer Methods in Applied Mechanics and Engineering* (2016) **308**, pp. 349 - 376

[8] Bonizzoni, F. and Nobile, F. Perturbation Analysis for the Darcy Problem with Log-Normal Permeability. *SIAM/ASA Journal on Uncertainty Quantification* (2014) **1**, pp. 223-244

[9] Bonizzoni, F. and Pradovera, D. and Ruggeri, M. Rational-based model order reduction of Helmholtz frequency response problems with adaptive finite element snapshots *arXiv:2112.04302* (2021)

[10] Fumagalli, I. and Manzoni, A. and Parolini, N. and Verani, M. Reduced basis approximation and a posteriori error estimates for parametrized elliptic eigenvalue problems. *ESAIM: M2AN* (2016), **50**: 6, pp. 1857-1885

[11] Georg, N. and Ackermann, W. and Corno, J. and Schöps, S. Uncertainty quantification for Maxwell's eigenproblem based on isogeometric analysis and mode tracking. *Computer Methods in Applied Mechanics and Engineering* (2019), **350**, pp. 228-244

[12] Hakula, H. and Kaarnioja, V. and Laaksonen, M. Approximate methods for stochastic eigenvalue problems. *Applied Mathematics and Computation* (2015) **267**, pp. 664-681

[13] Hesthaven, J.S. and Rozza, G. and Stamm, B. *Certified reduced basis methods for parametrized partial differential equations* SpringerBriefs in Mathematics. Springer, Cham; BCAM Basque Center for Applied Mathematics, Bilbao. (2016), pp. xiii+131

[14] Horger, T. and Wohlmuth, B. and Dickopf, T. Simultaneous reduced basis approximation of parameterized elliptic eigenvalue problems. *ESAIM: M2AN* (2017) **51**: 2, pp. 443-465

[15] Huynh, D.B.P. and Knezevic, D.J. and Patera, A.T. A Static condensation Reduced Basis Element method: approximation and a posteriori error estimation. *ESAIM: Mathematical Modelling and Numerical Analysis* (2013) **47**: 1, pp. 213–251

[16] Jorkowski, P. and Schuhmann, R. Mode tracking for parametrized eigenvalue problems in computational electromagnetics. *2018 International Applied Computational Electromagnetics Society Symposium (ACES)* (2018) pp. 1-2

[17] Kuhn, H. W. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly* (1955) **2**: 1-2, pp. 83-97

[18] Le Maître, Olivier and Knio, Omar M. Spectral methods for uncertainty quantification: with applications to computational fluid dynamics. *Springer Science & Business Media* (2010)

[19] Machiels, L. and Maday, Y. and Oliveira, I.B. and Patera, A.T. and Rovas, D.V. Output bounds for reduced-basis approximations of symmetric positive definite eigenvalue problems. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics* (2000) **331**: 2, pp. 153-158

[20] The MathWorks, Inc. Partial Differential Equation Toolbox. (2022) https://www.mathworks.com/help/pde/

[21] Nobile, F. and Pradovera, D. Non-intrusive double-greedy parametric model reduction by interpolation of frequency-domain rational surrogates. *ESAIM: M2AN* (2021) **55**: 5, pp. 1895-1920

[22] Pau, G.S.H. Reduced-basis method for band structure calculations. *Phys. Rev. E* (2007) **76**: 4, pp. 046704

[23] Piazzola, C. and Tamellini, L. The Sparse Grids Matlab kit - a Matlab implementation of sparse grids for high-dimensional function approximation and uncertainty quantification. *arXiv 2203.09314* (2022)

[24] Quarteroni, A. and Manzoni, A. and Negri, F. *Reduced basis methods for partial differential equations: an introduction.* Springer (2015) **92**

[25] Robert, C.P. and Casella, G. Monte Carlo statistical methods. *Springer Location New York, NY* (2004)

[26] Vallaghé, S. and Huynh, P. and Knezevic, D.J and Nguyen, L. and Patera, A.T. Component-based reduced basis for parametrized symmetric eigenproblems. *Advanced Modeling and Simulation in Engineering Sciences* (2015) **2**: 1, pp. 1–30

[27] Yang, D. and Ajjarapu, V. Critical Eigenvalues Tracing for Power System Analysis via Continuation of Invariant Subspaces and Projected Arnoldi Method. *2007 IEEE Power Engineering Society General Meeting* (2007), pp. 1-9

[28] Yue, Y. and Feng, L. and Benner, P. Reduced-order modelling of parametric systems via interpolation of heterogeneous surrogates. *Adv. Model. and Simul. in Eng. Sci.* (2019) **6**: 10

[29] Ziegler, A. and Georg, N. and Ackermann, W. and Schöps, S. Mode Recognition by Shape Morphing for Maxwell's Eigenvalue Problem. *arXiv:2203.00499* (2022)

MOX Technical Reports, last issues

Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

64/2022 Massi, M.C., Dominoni, L., Ieva, F., Fiorito, G.
A Deep Survival EWAS approach estimating risk profile based on pre-diagnostic DNA methylation: an application to Breast Cancer time to diagnosis

63/2022 Corti, M.; Antonietti, P.F.; Dede', L.; Quarteroni, A.
Numerical Modelling of the Brain Poromechanics by High-Order Discontinuous Galerkin Methods

65/2022 Dassi, F.; Fumagalli, A.; Mazzieri, I.; Vacca, G.
Mixed Virtual Element approximation of linear acoustic wave equation

66/2022 Antonietti, P.F.; Liverani, L.; Pata, V.
Lack of superstable trajectories in linear viscoelasticity: A numerical approach

62/2022 Ciaramella, G.; Halpern, L.; Mechelli, L.
Convergence analysis and optimization of a Robin Schwarz waveform relaxation method for periodic parabolic optimal control problems

61/2022 Gregorio, C.; Cappelletto, C.; Romani, S.; Stolfo, D.; Merlo, M.; Barbati, G.
Using marginal structural joint models to estimate the effect of a time-varying treatment on recurrent events and survival: An application on arrhythmogenic cardiomyopathy

60/2022 Cortellessa, D.; Ferro, N.; Perotto, S.; Micheletti, S.
Enhancing level set-based topology optimization with anisotropic graded meshes

59/2022 Boon, W. M.; Fumagalli, A.
A multipoint vorticity mixed finite element method for incompressible Stokes flow

58/2022 Zingaro, A.; Bucelli, M.; Fumagalli, I.; Dede', L; Quarteroni, A.
Modeling isovolumetric phases in cardiac flows by an Augmented Resistive Immersed Implicit Surface Method

57/2022 Ruffino, L.; Santoro, A.; Sparvieri, S.; Regazzoni, F.; Adebo, D.A.; Quarteroni, A.; Vergara, C.; Corno, A.F.
Computational analysis of cardiovascular effects of COVID- 19 infection in children