Journal of Computational and Applied Mathematics 457 (2025) 116270

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

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

Moataz Alghamdi [a], Daniele Boffi [a,b,1], Francesca Bonizzoni [c,2,*]

[a] KAUST, Thuwal, 23955-6900, Kingdom of Saudi Arabia
[b] Department of Mathematics, University of Pavia, via Ferrata 1, Pavia, 27100, Italy
[c] MOX, Department of Mathematics, Politecnico di Milano, Piazza Leonardo da Vinci 32, Milano, 20133, Italy

ARTICLE INFO

*MSC:*
65N25
65N30
35B30
78M34
35P15
62K25

*Keywords:*
Eigenvalue problem
Parameter-dependent partial differential equation
Model reduction
Eigenvalue matching
A posteriori error indicator
Sparse grid

ABSTRACT

In this paper we introduce a Model Order Reduction (MOR) 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.

## 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

* Corresponding author.
*E-mail addresses:* moataz.alghamdi@kaust.edu.sa (M. Alghamdi), daniele.boffi@kaust.edu.sa (D. Boffi), francesca.bonizzoni@polimi.it (F. Bonizzoni).

[1] The work of D. Boffi was supported by the Competitive Research Grants Program CRG2020 "Synthetic data-driven model reduction methods for modal analysis" and by the Opportunity Fund Program OFP2023 "Model Reduction Methods for Modal Analysis in Computational Mechanics" awarded by the King Abdullah University of Science and Technology (KAUST). D. Boffi is member of the INdAM Research group GNCS and his research is partially supported by PRIN/MIUR.

[2] F. Bonizzoni is member of the INdAM Research group GNCS and her work is part of a project that has received funding from the European Research Council ERC under the European Union's Horizon 2020 research and innovation program (Grant agreement No. 865751). F Bonizzoni is partially funded by "INdAM - GNCS Project", codice CUP E53C23001670001 and codice CUP E53C22001930001. FB has received support from the project PRIN2022, MUR, Italy, 2023–2025, P2022N5ZNP "SIDDMs: shape-informed data-driven models for parametrized PDEs, with application to computational cardiology".

https://doi.org/10.1016/j.cam.2024.116270
Received 8 January 2024; Received in revised form 31 July 2024
Available online 17 September 2024

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., [1,2]) 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, ROM for source problems is a quite mature research area, whereas model reduction for parametric/stochastic eigenproblems is still a largely unexplored field. The way was paved by the pioneering work [3], where an RB approach is proposed to approximate the smallest eigenvalue. This methodology has been further developed to deal with more eigenvalues in [4]. A component-based RB method for eigenvalue problems is proposed in [5] and an a posteriori estimator for eigenvalues is studied in [6]. All the previously mentioned contributions do not cover the case of multiple eigenpairs. Instead, in [7], 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 [8] 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 in the context of stochastic eigenproblems has been solved in [9–11] by means of the Stochastic Galerkin and Stochastic Collocation method, respectively, and, more recently, in [12]. Another contribution in this context can be found in [13], where the particular structure of the problem is exploited, in the spirit of [14].

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.

Indeed, reduced bases were constructed in previous approaches by taking into account a combination of all eigenspaces involved with such crossings. However, this naively assumes that the chosen eigenspaces are well separated from the rest of the spectrum, which has to be shown beforehand. Our algorithm allows the user to find the intersection of eigenvalues and this can be used also to identify isolated clusters of crossing eigencurves in the spirit of a subspace approach.

The parameter space is sampled adaptively, following a two-phase procedure. First, a suitably adapted version of the a priori matching proposed in [15] 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 [16–19], 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 problem, which in turn are eigenvalues of the corresponding elliptic problem. A matching strategy in the same spirit as [15] and the a priori matching of the present paper is studied in [20]. Moreover, a mode tracking method for the parametrized Maxwell eigenvalue problem is presented in [21–24].

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 beyond the scope 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

$V$ compact in $H$,

$a(\cdot,\cdot;\mu)$ elliptic in $V \quad \forall \mu \in \mathcal{M}$,

$b(\cdot,\cdot;\mu)$ 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

$$a(u, v; \mu) = \lambda(\mu) b(u, v; \mu) \quad \forall v \in V. \tag{1}$$

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 $\mu_1$ and $\mu_2$ in $\mathcal{M}$ can be different from each other when $\mu_1 \neq \mu_2$.

Problem (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

$$a(u_h, v; \mu) = \lambda_h(\mu) b(u_h, v; \mu) \quad \forall v \in V_h. \tag{2}$$

This will be considered as our *high-fidelity* solution. We refer the interested reader to classical references for the study of the finite element solution of partial differential equations [25,26].

## 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 $I_\lambda$ 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 $\mu_i$ and $\mu_k$ 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 $\mu_i$ and $\mu_k$ 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.

To keep track of the outputs of such algorithms, we use an undirected graph. An undirected graph is a pair $(V, E)$ where $V$ is a set of nodes (grid points) and $E$ is a collection of unordered pairs between such nodes. We discuss how such a data structure is convenient for keeping track of all refinements at each level. Such a graph also provides a natural way for reconstructing the final solution at the end of the algorithm.

### 3.1. The a priori matching

The FE method (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 $h$ ($\lambda$ instead of $\lambda_h$, etc.).

We consider two different parameter values $\mu_i$ and $\mu_k$ 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 $I_\lambda$. 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 $\mu_i$ and $\mu_k$ has size $n_i \times n_k$ and entries

$$D^{(i,k)}_{j,\ell} := w_1 |\lambda_j(\mu_i) - \lambda_\ell(\mu_k)| + w_2 \min\{\|u_j(\mu_i) - u_\ell(\mu_k)\|_{b,\bar{\mu}}, \|u_j(\mu_i) + u_\ell(\mu_k)\|_{b,\bar{\mu}}\}, \tag{3}$$

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

**Remark 3.2.** A similar definition to (3) can be found in [15]. However, some modifications are necessary in the present context, since the normalization of the eigenfunctions does not 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 $\sigma^\star = (\sigma_1, \dots, \sigma_{\bar{n}}) \in (1, \dots, \bar{n})!$ such that

$$\sigma^\star := \operatorname{argmin}_{\sigma \in (1,\dots,\bar{n})!} \sum_{\alpha=1}^{n_i} \sum_{\beta=1}^{n_k} D^{i,k}_{\sigma_\alpha, \sigma_\beta}, \tag{4}$$

with $\bar{n} := \min\{n_i, n_k\}$. Between the several options available to compute the solution of (4), we adopt the so-called Hungarian Algorithm [27]. The permutation solution is then used to reorder the eigensolutions so that there is a one-to-one correspondence between the matched eigensolutions

For each subinterval, one can have three possible scenarios, namely $n_i < n_k$, $n_i > n_k$ and $n_i = n_k$. In all scenarios, the output of the Hungarian algorithm will be interpreted similarly. In other words, the one-to-one correspondence between the matched eigensolutions will be always established, and $\bar{n}$ matching eigenvalues will be detected. This is enforced by our choice of a high enough unassignment cost.

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

**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)
2: Find $\sigma^\star$ solution to (4) ▷ Hungarian algorithm
3: Set $(\lambda_j^\star(\mu_i), u^\star(\mu_i)) = (\lambda_j(\mu_i), u_j(\mu_i))$ for $j = 1, \dots, n_i$
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 $\ell = 1, \dots, n_k$

### 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.

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

$$\Pi^{(i,k)}_{j,\ell} = |b(u_j(\mu_i), u_\ell(\mu_k), \bar{\mu})|, \tag{5}$$

with $\bar{\mu}$ being a fixed parameter value between $\mu_i$ and $\mu_k$.

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 $t_\pi$. This is used to truncate $\Pi^{(i,k)}$ — inside a loop over $j = 1, \dots, \min\{n_i, n_k\}$ — 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 $t_\lambda$ 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 $t_\lambda$ stopping criterion.

**Remark 3.4** (*Choice of the Tolerances* $t_\pi, t_\lambda$)**.** 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 $t_\pi$ 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 $t_\pi$ becomes more delicate as the dimension of the parameter space gets larger.

A similar balance must hold for $t_\lambda$. 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 $t_\lambda$ is chosen such that those overlapping eigenfunctions are identified as a cluster. However, if $t_\lambda$ 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

```
Set if_refinement=0
Compute Π^(i,k) ∈ R^(n_i × n_k)                                   ▷ See Equation (5)
for j = 1 to min(n_i, n_k) do
    for ℓ = 1 to n_k do
        if Π^(i,k)_(j,j) ≥ Π^(i,k)_(j,ℓ) + t_π then
            Set Π^(i,k)_(j,ℓ) = 0                                 ▷ Truncate Π^(i,k)_(j,:) up to tolerance t_π
        end if
    end for
    for ℓ = 1 to n_i do
        if Π^(i,k)_(j,j) ≥ Π^(i,k)_(ℓ,j) + t_π then
            Set Π^(i,k)_(ℓ,j) = 0                                 ▷ Truncate Π^(i,k)_(:,j) up to tolerance t_π
        end if
    end for
    Let r_1 = find(Π^(i,k)_(j,:)), r_2 = find(Π^(i,k)_(:,j))      ▷ Indices of the non-zero elements
    if (length(r_1)) > 1 or (length(r_2)) > 1) then
        if length(r_1) ≠ length(r_2) then
            if_refinement = 1                                     ▷ Mark [μ_i, μ_k] for refinement
            break
        end if
        Let α^i_(j,γ_1) := |λ_j(μ_i) − λ_γ1(μ_i)| / λ_j(μ_i) for γ_1 ∈ r_1
        Let α^k_(j,γ_2) := |λ_j(μ_k) − λ_γ2(μ_k)| / λ_j(μ_k) for γ_2 ∈ r_2
        if max_(γ_1∈r_1, γ_2∈r_2) {α^i_(j,γ_1), α^k_(j,γ_2)} > t_λ then   ▷ No cluster is identified
            if_refinement = 1                                     ▷ Mark [μ_i, μ_k] for refinement
            break
        end if
    end if
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 [15,28]. In particular, the refinement step described in Section 3.4 relies on 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:

$$\Gamma(m) := \begin{cases} \{0\} & \text{if } m = 0, \\ \{2^{1-m}j\}_{j=-2^{m-1}}^{2^{m-1}} & \text{if } m > 0. \end{cases} \tag{6}$$

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

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

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 $\mu$ as

$$\mathcal{V}(\mu) := \{(\mu_1 \pm 2^{-m_1}, \mu_2),\ (\mu_1, \mu_2 \pm 2^{-m_2})\} \cap \mathcal{M}, \tag{7}$$

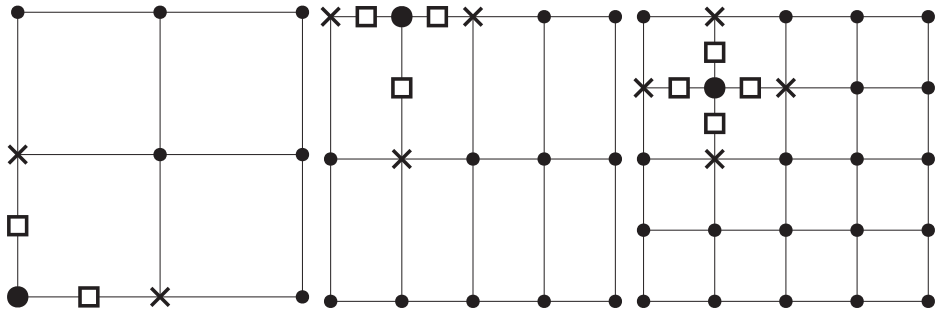


**Fig. 1.** Forward points (squares) and neighbors (crosses) of $\mu = (-1,-1) \in \Gamma(1,1)$ (left); $\mu = (-\frac{1}{2},1) \in \Gamma(2,1)$ (middle); $\mu = (-\frac{1}{2},\frac{1}{2}) \in \Gamma(2,2)$ (right).

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

$$\mathcal{U}(\mu) := \{(\mu_1 \pm 2^{-(m_1-1)}, \mu_2), (\mu_1, \mu_2 \pm 2^{-(m_2-1)})\} \cap \mathcal{M}. \tag{8}$$

Note that both sets $\mathcal{V}(\mu)$ and $\mathcal{U}(\mu)$ contain up to $2d$ points for any $d \geq 2$. Some examples are depicted in Fig. 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 $\ell$-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 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 Eq. (8)). 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, or until the maximum level $L$ has been reached.

### 3.5. Updating the graph

Observe that each local subinterval (at level $\ell \in \mathbb{N}_0$) defines a pair of parameters and thus an edge. As a consequence, we naturally have an undirected graph $(V, E)$, where $E$ is the collection of such edges and $V$ is the collection of the associated vertices, namely the grid points in $P^{(\ell)}$. The refinement strategy naturally defines a way to update this graph. In particular, the vertices will be updated so that they are the grid points in $P^{(\ell+1)}$ and the edges will be given by the local subintervals at the $(\ell+1)$ level.

Notably, it is always possible to find the minimum spanning tree associated with the graph at each level. In particular, this is a new graph connecting the same vertices without any cycles. For our purposes, we assume that the weights of this graph are equal. Thus, this tree defines a set of one-dimensional paths in any $d$-dimensional parametric space. The creation of the tree is particularly important when we exit the loop over the levels (either because no more refinements are needed or because the maximum level $L$ has been reached). At that point, we perform the a-priori matching procedure following the identified one-dimensional paths. Note that the removal of cycles allows for no ambiguities in defining the direction of the matching since we always work with a series of one-dimensional paths.

### 3.6. The complexity of the Algorithm

Let us first discuss about the number of points contained in the parameter grid of level $\ell$ in the worst-case scenario. We define the worst-case scenario to be when we refine everywhere, or equivalently, when the output of Algorithm 2 is always `if_refinement = 1`. In this case, we know that the total number of inner grid points at each level $\ell$ (for a $d$-dimensional problem) is $Q_{d,\ell} = \sum_{i=0}^{\ell-1} 2^i \binom{d-1+i}{d-1}$. Next, we count the outer (boundary) points. The formula for generating the number of $f$-faces on the boundary of a $d$-dimensional hypercube is $N_{d,f} = 2^{d-f}\binom{d}{f}$. Each such $f$-face contains a sparse grid of dimension $f$. Thus, the total number of outer grid points is $\sum_{f=1}^{d-1} N_{d,f} Q_{f,\ell}$. In addition, there are $N_{d,0}$ 0-faces or vertices. Thus, the formula for all grid points is

$$\sum_{f=1}^{d} N_{d,f} Q_{f,\ell} + N_{d,0}.$$

**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 $P_\delta^{(\ell)} = \{\mu_s^{(\ell,\delta)}\}_{s=1}^{N_{\ell,\delta}}$, eigenpairs $\Lambda_\delta^{(\ell)} := \{(\lambda_j(\mu_s^{(\ell,\delta)}), u_j(\mu_s^{(\ell,\delta)}))\}_{j=1}^{n_s}$ and graph $(V, E)$

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

Set $P_\delta^{(\ell+1)} = \emptyset$, $\Lambda_\delta^{(\ell+1)} = \emptyset$
**for** $s = 1 : N_{\ell,\delta}$ **do** ▷ Loop on the points in $P_\delta^{(\ell)}$
  **for** $\nu_{(r,s)} \in \mathcal{V}(\mu_s^{(\ell,\delta)})$ **do** ▷ Loop on the neighbors of $\mu_s^{(\ell,\delta)}$ (8)
    $\Lambda_s^\star := \{(\lambda_j(\mu_s^{(\ell,\delta)}), u_j(\mu_s^{(\ell,\delta)}))\}_{j=1}^{n_s} \leftarrow \{(\lambda_j(\mu_s^{(\ell,\delta)}), u_j(\mu_s^{(\ell,\delta)}))\}_{j=1}^{n_s}$
    $\Lambda_r^\star := \{(\lambda_j^\star(\nu_{(r,s)}), u_j^\star(\nu_{(r,s)}))\}_{j=1}^{n_s} \leftarrow \{(\lambda_j(\nu_{(r,s)}), u_j(\nu_{(r,s)}))\}_{j=1}^{n_r}$ ▷ See Algorithm 1
    if_refinement = a-posteriori check on $\Lambda_s^\star$, $\Lambda_r^\star$ ▷ See Algorithm 2
    **if** if_refinement = 1 **then**
      $P_\delta^{(\ell+1)} \leftarrow \bar{\nu}_{(r,s)}$ ▷ $\bar{\nu}_{(r,s)}$ is the corresponding forward point of $\mu_s^{(\ell,\delta)}$
      Compute $\bar{\Lambda} := \{(\lambda_j(\bar{\nu}_{(r,s)}), u_j(\bar{\nu}_{(r,s)}))\}_{j=1}^{n_r}$
      $\Lambda_\delta^{(\ell+1)} \leftarrow \bar{\Lambda}$
    **end if**
  **end for**
**end for**
Update the graph $(V, E)$

As for the asymptotic behavior of this formula, using $Q_{d,f} \approx 2^\ell \ell^{f-1}$ (see [29]), we derive

$$\begin{aligned}
\sum_{f=1}^{d} N_{d,f} Q_{f,\ell} + N_{d,0} &\approx \sum_{f=0}^{d} 2^\ell \ \ell^{f-1} \ 2^{d-f} \binom{d}{f} + 2^d \\
&\approx 2^\ell \ 2^d \ \frac{1}{\ell} \ \sum_{f=0}^{d} \ell^f \ 2^{-f} \binom{d}{f} + 2^d \\
&\approx 2^\ell \ 2^d \ \frac{1}{\ell} \ \sum_{f=0}^{d} \left(\frac{\ell}{2}\right)^f \binom{d}{f} + 2^d \\
&\le 2^\ell \ 2^d \ \frac{1}{\ell} \ \sum_{f=0}^{\infty} \left(\frac{\ell}{2}\right)^f \binom{d}{f} + 2^d \\
&\le 2^\ell \ 2^d \ \frac{1}{\ell} \left(1 + \frac{\ell}{2}\right)^d + 2^d \\
&\approx 2^\ell \ \ell^{d-1}.
\end{aligned}$$

Thus, the total number of points is $O(2^\ell \ell^{d-1})$.

Let us now move to the discussion about the complexity of the presented algorithms in the worst-case scenario. Observe that the local Algorithms 1 and 2 only depend on the number of eigenvalues considered in the associated subinterval. They do not depend on the dimension of the parametric domain, $\mathcal{M}$. Since both the parametric domain $\mathcal{M}$ and the window of interest $I_\lambda$ are bounded, the number of eigensolutions at any point in $\mathcal{M}$ is also bounded. Furthermore, let $C_{local}$ denote the time it takes to execute both Algorithms 1 and 2 for a certain subinterval. In addition, let $C_{solver}$ denote the time it takes to generate the eigensolutions at an arbitrary parameter $\mu$. Due to the boundedness described above, we know that such values are bounded from above. We denote such bounds by $C_p$ and $C_e$, respectively.

At any level $\ell$, the operations performed according to Algorithm 3 are: (i) solve the eigenvalue problem corresponding to all new points of the parametric grid; (ii) run Algorithms 1 and 2 at all local intervals having one new grid point as endpoint (up to $2d$); (iii) update the graph. Therefore, the estimate of the runtime of Algorithm 3 is

$$(2^\ell \ell^{d-1} - 2^{\ell-1}(\ell-1)^{d-1})(C_p + 2d\, C_e + 1).$$

When Algorithm 3 is repeated until the maximum level $L$ is reached, the cumulative runtime becomes

$$\sum_{\ell=1}^{L} (2^\ell \ell^{d-1} - 2^{\ell-1}(\ell-1)^{d-1})(C_p + 2d\, C_e + 1).$$

Finally, once the loop over the levels has terminated, we start from the graph corresponding to the created adapted grid, and we construct the minimum spanning tree. This operation is performed in MATLAB by means of the function `minspantree`. The algorithm used there is the Kruskal's algorithm [30,31] which has a worst-case performance of $O(ElogV)$ where $E$ and $V$ are the

numbers of edges/subintervals and vertices/grid points, respectively. Thus, this translates asymptotically (in the limit where the number of subintervals equals the number of grid points) to

$$O((2^L(L)^{d-1})\log(2^L L^{d-1})).$$

Putting everything together, we conclude that the runtime for the global algorithm is

$$(2^L(L)^{d-1})\log(2^L L^{d-1}) + \sum_{\ell=1}^{L}[(2^\ell \ell^{d-1} - 2^{\ell-1}(\ell-1)^{d-1})(C_p + 2d\,C_e + 1)].$$

Now, the number of inner grid points for a uniform grid is $O(2^{dL})$. The runtime using a uniform grid instead of a sparse grid is then:

$$(2^{dL})\log(2^{dL}) + \sum_{\ell=1}^{L}[(2^{d\ell} - 2^{d(\ell-1)})(C_p + 2d\,C_e + 1)].$$

Here, the exponential dependence on the dimension is obvious. This is in contrast to the exponential dependence on the level $\ell$ in the sparse grid case.

## 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

$$\begin{cases} \nabla \cdot (c(\mu)\nabla u(\mu)) = \lambda(\mu)u(\mu) & \text{in } \Omega, \\ u(\mu) = 0 & \text{on } \partial\Omega, \end{cases} \tag{9}$$

where $c(\mu)$ is a matrix with size $2 \times 2$ and positive definite for all possible values of the parameter $\mu$. Integrating by parts, we find the weak formulation of (9) of the form (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 [32] and Sparse Grids Kit [33].

### 4.1. Operations on a subinterval

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

$$c(\mu) = \begin{pmatrix} \mu^{-2} & 1 \\ 1 & 0.7^{-2} \end{pmatrix}, \tag{10}$$

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 $\mu_1$ (respectively $\mu_2$) delivers four (respectively nine) eigenvalues in $I_\lambda$ (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 $4 \times 9$ cost matrix (with weights $w_1 = 1$, $w_2 = 200$) reads as follows:

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

To minimize the total cost, namely, to solve the optimization problem (4), 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:

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

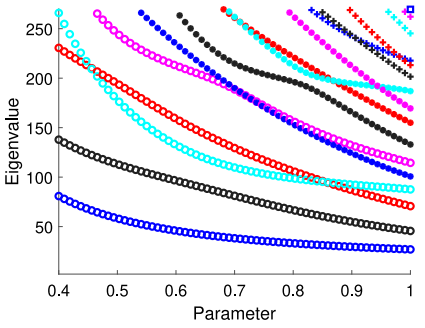



**Fig. 2.** Reference solution. Different colors and labels correspond to different eigencurves so that intersections are visible.

and the vector $\{\lambda_\ell(\mu_2),\ \ell = 1,\dots,9\}$ 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 $4 \times 9$ 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 $\Pi$ 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 $\Pi$:

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

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 $\mu_1$, 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 $\mu_2$. Furthermore, the second eigenvalue at $\mu_2$, 81.1, can be matched to the second and fourth (265.9 since $\Pi_{4,2} = 0.617$) eigenvalues at $\mu_1$.

We then check whether the second and fourth eigenvalues at $\mu_1$ 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 $\mu_2$ 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 Fig. 2). 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 the features of the reference solution. Fig. 3 depicts the evolution of the parametric grid as the level increases, whereas in each subfigure of Fig. 4 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 1.

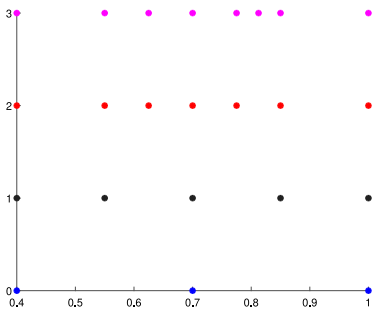



**Fig. 3.** Evolution of parametric grid ($x$-axis) as the level increases ($y$-axis).

**Table 1**
Error table for the 1D example.

| Error By Level | | | | | | |
|---|---|---|---|---|---|---|
| Level | Total no. of points | No. of wrongly matched points | No. of subintervals | No. of uncertified subintervals | No. of uniform grid points | No. of full sparse grid points |
| 0 | 3 | 2 | 2 | 2 | 3 | 3 |
| 1 | 5 | 3 | 4 | 2 | 5 | 5 |
| 2 | 7 | 0 | 4 | 1 | 9 | 9 |
| 3 | 8 | 0 | 2 | 0 | 17 | 17 |

**Level 0** The initial grid is $P^{(0)} = \{0.4, 0.7, 1\}$ (see the blue points in Fig. 3). 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 Fig. 3) and it is given by $P^{(1)} = P^{(0)} \cup P_\delta^{(1)}$, with $P_\delta^{(1)} = \{0.55, 0.85\}$. 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 Fig. 3) and it is given by $P^{(2)} = P^{(1)} \cup P_\delta^{(2)}$, with $P_\delta^{(2)} = \{0.625, 0.775\}$. 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 Fig. 3) and it is given by $P^{(3)} = P^{(2)} \cup P_\delta^{(3)}$, with $P_\delta^{(3)} = \{0.8125\}$. 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 $2 \times 2$ sub-matrix with entries corresponding to the black and cyan eigenvalues:

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

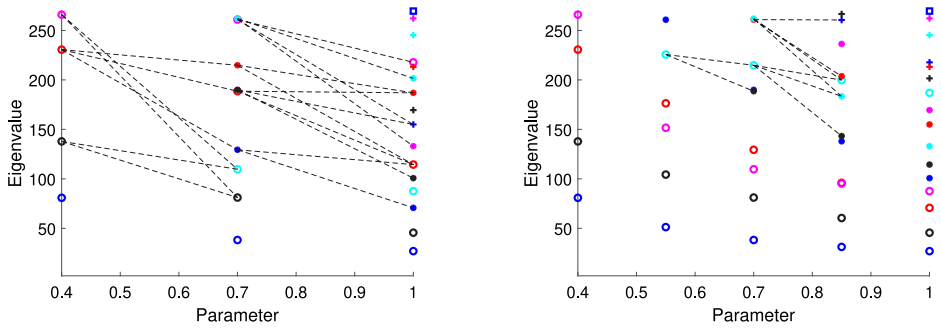
The off-diagonal terms $\Pi_{7,8}$ and $\Pi_{8,7}$ are non-zero due to a non-negligible overlap between the corresponding eigenfunctions, which are depicted in Fig. 5. Therefore, we can identify two causes for the a posteriori verification to suggest the refinement: (i) the ordering of the eigenvalues 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 Fig. 6.

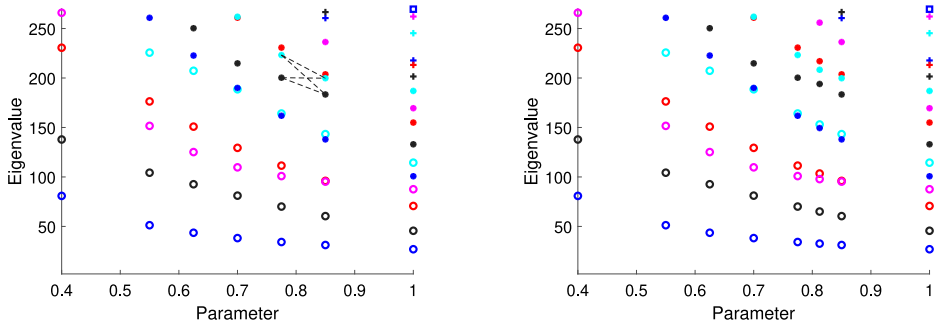
### *4.3. A 2D example*

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

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

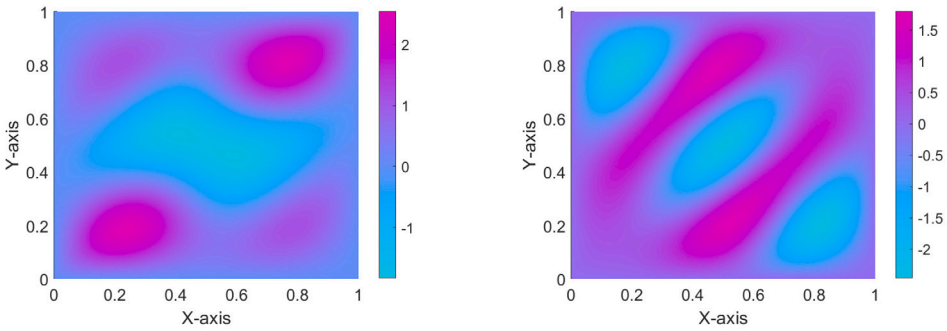



(a) Level 0 (b) Level 1

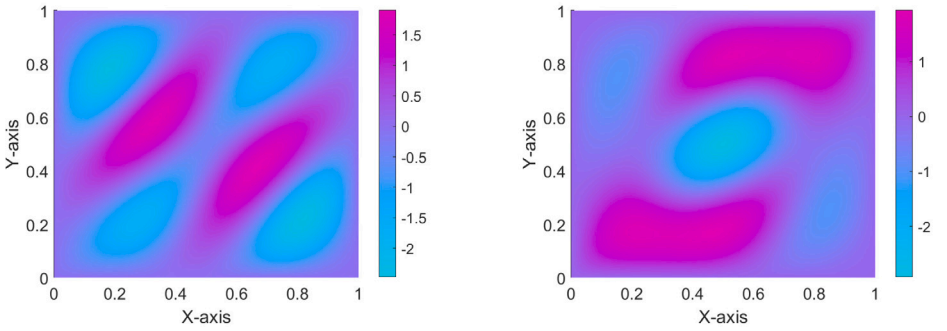



(c) Level 2 (d) Level 3

**Fig. 4.** Visual summary of the projection matrices.




(a) $\lambda = 200.3$, $\mu = 0.775$ (b) $\lambda = 223.23$, $\mu = 0.775$




(c) $\lambda = 183.369$, $\mu = 0.85$ (d) $\lambda = 199.786$, $\mu = 0.85$

**Fig. 5.** Comparison between four non-orthogonal eigenfunctions.

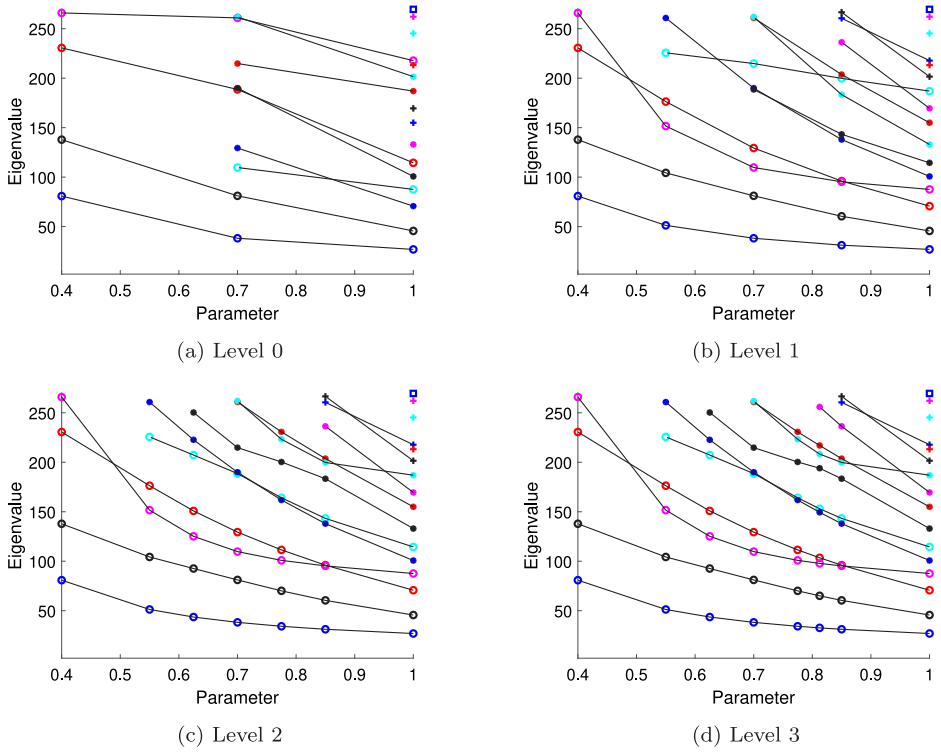



**Fig. 6.** Output of each level for the 1D example.

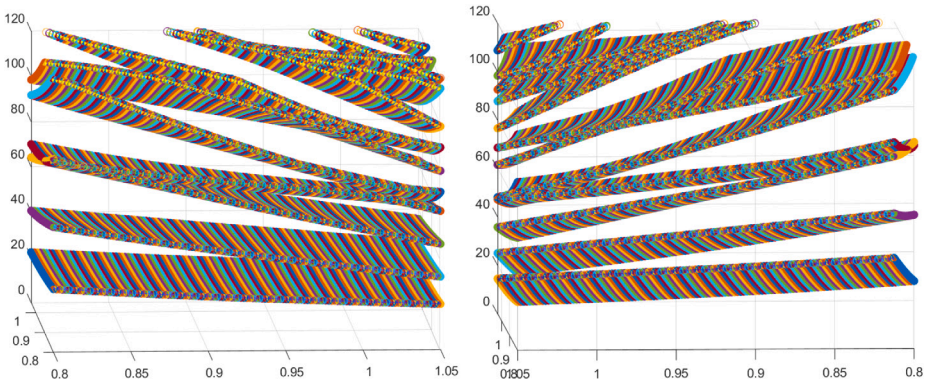
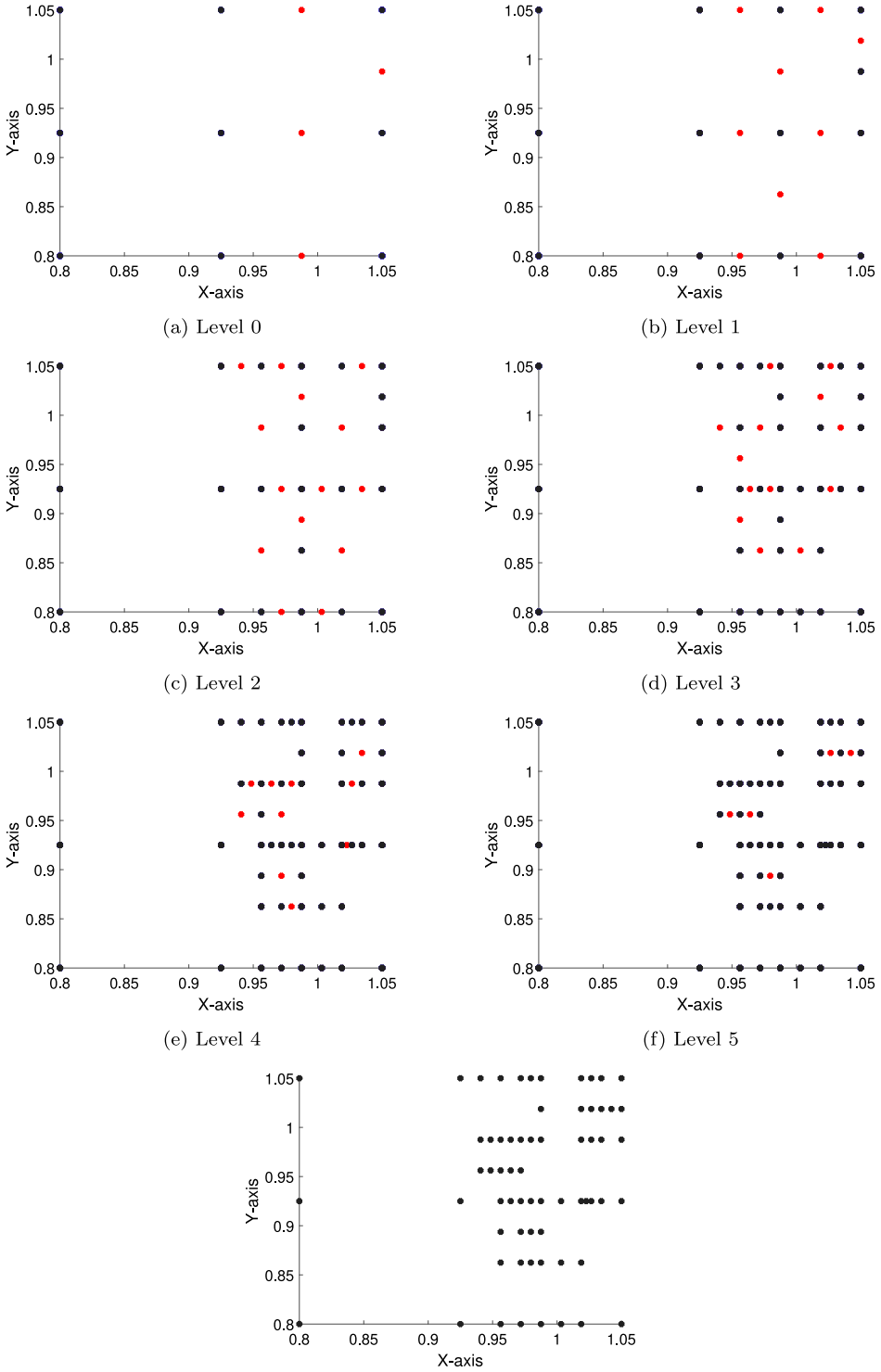


**Fig. 7.** Reference solution for the 2D example; All hypersurfaces.

The computation of the reference solution relies on the uniform tensor product grid of $\mathcal{M}$ containing 129 points (see Fig. 7 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.

Given the initial grid $P^{(0)}$ being the $3 \times 3$ 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 Fig. 8. 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 Fig. 9(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 Fig. 9(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.




(a) Level 0 (b) Level 1

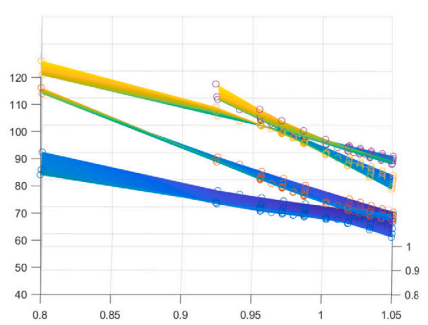
(c) Level 2 (d) Level 3

(e) Level 4 (f) Level 5

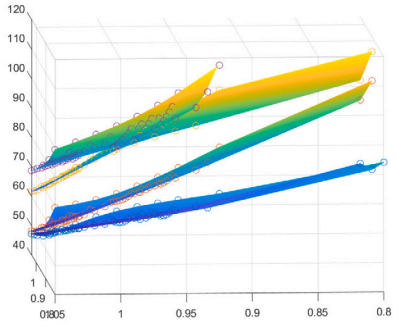
(g) Level 6

**Fig. 8.** Evolution of grid in the 2D example.

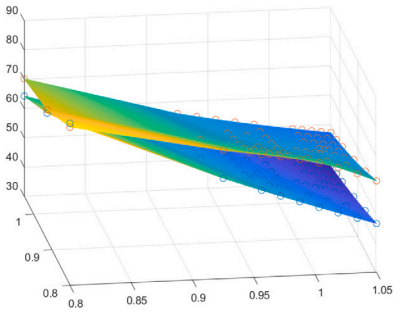
Table 2 summarizes the level-by-level information. We can observe that, from level 5 on, five points are added to the grid, even though all the eigenpairs were correctly matched. These points serve the purpose of allowing the algorithm to fully certify



(a) Hypersurfaces 4,7; Side 1



(b) Hypersurfaces 4,7; Side 2



(c) Hypersurfaces 3,4

**Fig. 9.** Final solution for the 2D example; Detailed view.

**Table 2**
Error table for the 2D example.

| Error By Level | | | | | | |
|---|---|---|---|---|---|---|
| Level | Total no. of points | No. of wrongly matched points | No. of subintervals | No. of uncertified subintervals | No. of uniform grid points | No. of full sparse grid points |
| 0 | 9 | 7 | 12 | 4 | 9 | 9 |
| 1 | 13 | 2 | 10 | 9 | 25 | 21 |
| 2 | 22 | 1 | 22 | 11 | 81 | 49 |
| 3 | 36 | 4 | 36 | 10 | 289 | 113 |
| 4 | 49 | 2 | 39 | 4 | 1089 | 257 |
| 5 | 59 | 0 | 34 | 2 | 4225 | 577 |
| 6 | 64 | 0 | 20 | 0 | 16 641 | 1281 |

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

Next, we compare the performance of our algorithm against two different approaches: using a full sparse grid and a full uniform grid. In such cases, the number of grid points added at each level is fixed. The number of such points is given in Tables 1 and 2. One can see that the size of the full uniform grid is equal to that of the full sparse grid in 1D. However, it is about 13 times larger in 2D at the end of the refinement process, that is, when the smallest subinterval needed to resolve the matching problem is introduced to the grid. Such a difference in size is even more striking in the case of our adaptive sparse grid. In particular, the adaptive grid is about 20 times smaller than the full sparse grid, and 260 times smaller than the full uniform grid.

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`.

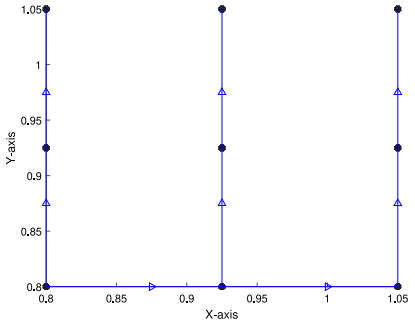



**Fig. 10.** Initial path.

As an example, we display the initial path corresponding to level 0 in Fig. 10. We generate this path by applying the `minspantree` function on the graph $G$ =`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.

## 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 [34], non-intrusive and Galerkin methods [35] and perturbation methods [36–38]. 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.

## Data availability

No data was used for the research described in the article.

## References

[1] A. Quarteroni, A. Manzoni, F. Negri, Reduced basis methods for partial differential equations: an introduction, vol. 92, Springer, 2015.
[2] J. Hesthaven, G. Rozza, B. Stamm, Certified reduced basis methods for parametrized partial differential equations, in: SpringerBriefs in Mathematics, Springer, Bilbao, 2016, p. xiii+131, BCAM SpringerBriefs.
[3] L. Machiels, Y. Maday, I. Oliveira, A. Patera, D. Rovas, Output bounds for reduced-basis approximations of symmetric positive definite eigenvalue problems, Comptes R. Acad. Sci. Series I - Math. 331 (2) (2000) 153–158.
[4] G. Pau, Reduced-basis method for band structure calculations, Phys. Rev. E 76 (2007) 046704.
[5] S. Vallaghé, P. Huynh, D. Knezevic, L. Nguyen, A. Patera, Component-based reduced basis for parametrized symmetric eigenproblems, Adv. Model. Simul. Eng. Sci. 2 (1) (2015) 1–30.
[6] D. Huynh, D. Knezevic, A. Patera, A static condensation reduced basis element method: approximation and a posteriori error estimation, ESAIM Math. Model. Numer. Anal. 47 (1) (2013) 213–251.
[7] T. Horger, B. Wohlmuth, T. Dickopf, Simultaneous reduced basis approximation of parameterized elliptic eigenvalue problems, ESAIM: M2AN 51 (2) (2017) 443–465.
[8] I. Fumagalli, A. Manzoni, N. Parolini, M. Verani, Reduced basis approximation and a posteriori error estimates for parametrized elliptic eigenvalue problems, ESAIM: M2AN 50 (6) (2016) 1857–1885.
[9] H. Hakula, V. Kaarnioja, M. Laaksonen, Approximate methods for stochastic eigenvalue problems, Appl. Math. Comput. 267 (2015) 664–681.
[10] L. Grubišić, M. Saarikangas, H. Hakula, Stochastic collocation method for computing eigenspaces of parameter-dependent operators, Numer. Math. 153 (1) (2023) 85–110.
[11] R. Andreev, C. Schwab, Sparse tensor approximation of parametric eigenvalue problems, in: Numerical Analysis of Multiscale Problems, Springer Berlin Heidelberg, 2012, pp. 203–241.

[12] J. Dölz, D. Ebert, On uncertainty quantification of eigenvalues and eigenspaces with higher multiplicity, SIAM J. Numer. Anal. 62 (1) (2024) 422–451.
[13] H. Hakula, M. Laaksonen, Multiparametric shell eigenvalue problems, Comput. Methods Appl. Mech. Engrg. 343 (2019) 721–745.
[14] N. Perkins, C. Mote, Comments on curve veering in eigenvalue problems, J. Sound Vib. 106 (3) (1986) 451–463.
[15] F. Nobile, D. Pradovera, Non-intrusive double-greedy parametric model reduction by interpolation of frequency-domain rational surrogates, ESAIM: M2AN 55 (5) (2021) 1895–1920.
[16] F. Bonizzoni, F. Nobile, I. Perugia, Convergence analysis of padé approximations for Helmholtz frequency response problems, ESAIM Math. Model. Numer. Anal. 52 (4) (2018) 1261–1284.
[17] F. Bonizzoni, F. Nobile, I. Perugia, D. Pradovera, Fast least-squares padé approximation of problems with normal operators and meromorphic structure, Math. Comp. 89 (2020) 1229–1257.
[18] F. Bonizzoni, F. Nobile, I. Perugia, D. Pradovera, Least-squares padé approximation of parametric and stochastic Helmholtz maps, Adv. Comput. Math. 46 (46) (2020).
[19] F. Bonizzoni, D. Pradovera, M. Ruggeri, Rational-approximation-based model order reduction of Helmholtz frequency response problems with adaptive finite element snapshots, Math. Eng. 5 (4) (2023).
[20] Y. Yue, L. Feng, P. Benner, Reduced-order modelling of parametric systems via interpolation of heterogeneous surrogates., Adv. Model. and Simul. in Eng. Sci. 6 (10) (2019).
[21] N. Georg, W. Ackermann, J. Corno, S. Schöps, Uncertainty quantification for Maxwell's eigenproblem based on isogeometric analysis and mode tracking, Comput. Methods Appl. Mech. Engrg. 350 (2019) 228–244.
[22] A. Ziegler, N. Georg, W. Ackermann, S. Schöps, Mode recognition by shape morphing for Maxwell's eigenvalue problem in cavities, IEEE Trans. Antennas Propagat. 71 (5) (2023) 4315–4325.
[23] D. Yang, V. Ajjarapu, Critical eigenvalues tracing for power system analysis via continuation of invariant subspaces and projected Arnoldi method, in: 2007 IEEE Power Engineering Society General Meeting, 2007, pp. 1–9.
[24] P. Jorkowski, R. Schuhmann, Mode tracking for parametrized eigenvalue problems in computational electromagnetics, in: 2018 International Applied Computational Electromagnetics Society Symposium, ACES, 2018, pp. 1–2.
[25] D. Boffi, Finite element approximation of eigenvalue problems, Acta Numer. 19 (2010) 1–120.
[26] I. Babuska, J. Osborn, Eigenvalue problems, in "handbook of numerical analysis", vol. II, 1991, pp. 645–785, PG Ciarlet and JL Lions Editions, Elsevier science Publishers BV (North-Holland).
[27] H.W. Kuhn, The Hungarian method for the assignment problem, Nav. Res. Logist. Q. 2 (1–2) (1955) 83–97.
[28] F. Alsayyari, Z. Perkó, D. Lathouwers, J. Kloosterman, A nonintrusive reduced order modelling approach using proper orthogonal decomposition and locally adaptive sparse grids, J. Comput. Phys. 399 (2019) 108912.
[29] H.-J. Bungartz, M. Griebel, Sparse grids, Acta Numer. 13 (2004) 147–269.
[30] H. Loberman, A. Weinberger, Formal procedures for connecting terminals with a minimum total wire length, J. ACM 4 (4) (1957) 428–437.
[31] J.B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, Proc. Amer. Math. Soc. 7 (1) (1956) 48–50.
[32] The MathWorks, Inc., Partial differential equation toolbox, 2022, Natick, Massachusetts, United State, URL https://www.mathworks.com/help/pde/.
[33] C. Piazzola, L. Tamellini, Algorithm 1040: The Sparse Grids Matlab Kit - A Matlab Implementation of Sparse Grids for High-Dimensional Function Approximation and Uncertainty Quantification, vol. 50, Association for Computing Machinery, New York, NY, USA, 2024, p. 22.
[34] C. Robert, G. Casella, Monte Carlo statistical methods, Springer Location New York, NY, 2004.
[35] O. Le Maître, O.M. Knio, Spectral methods for uncertainty quantification: with applications to computational fluid dynamics, Springer Science & Business Media, 2010.
[36] F. Bonizzoni, F. Nobile, Regularity and sparse approximation of the recursive first moment equations for the lognormal Darcy problem, Comput. Math. Appl. 80 (12) (2020) 2925–2947.
[37] F. Bonizzoni, F. Nobile, D. Kressner, Tensor train approximation of moment equations for elliptic equations with lognormal coefficient, Comput. Methods Appl. Mech. Engrg. 308 (2016) 349–376.
[38] F. Bonizzoni, F. Nobile, Perturbation analysis for the Darcy problem with log-normal permeability, SIAM/ASA J. Uncertain. Quant. 2 (1) (2014) 223–244.