Solving Elliptic Eigenproblems with Adaptive Multimesh hp-FEM

Stefano Giani[a,1,*], Pavel Solin[b,1]

[a]*Department of Engineering, University of Durham, South Road, Durham DH1 3LE, UK*
[b]*Department of Mathematics and Statistics, University of Nevada, Reno, USA.*

Abstract

This paper proposes a novel adaptive higher-order finite element (hp-FEM) method for solving elliptic eigenvalue problems, where n eigenpairs are calculated simultaneously, but on individual higher-order finite element meshes. The meshes are automatically hp-refined independently of each other, with the goal to use an optimal mesh sequence for each eigenfunction. The method and the adaptive algorithm are described in detail. Numerical examples clearly demonstrate the superiority of the novel method over the standard approach where all eigenfunctions are approximated on the same finite element mesh.

Keywords: Eigenproblem, automatic adaptivity, higher-order finite element method, multimesh hp-FEM

1. Introduction

Eigenvalue problems arise in many important areas of science and engineering, including stability [2], acoustics [10], fluid-solid interaction [1], etc. Therefore, finding efficient numerical methods for their solution is of paramount importance.

The most common numerical methods used to approximate eigenvalue problems in PDE form are finite element methods (FEMs) due to their flexibility and reliability. Unfortunately, eigenvalue problems can be very difficult and plain FEMs may not be enough to compute good approximations of spectra. Over the years, many authors have studied adaptive FE methods for eigenvalue problems. There is now quite a substantial literature on a posteriori error estimation for eigenvalue problems [4, 9, 18].

Conventional adaptive FE algorithms solve for n eigenpairs simultaneously, calling a generalized eigensolver after each mesh refinement step [7, 5]. The calls to the generalized eigensolver on the sequence of meshes are all independent making it difficult to track eigenpairs through the sequence of meshes.

*Corresponding author

Email addresses: `stefano.giani@durham.ac.uk` (Stefano Giani), `solin@unr.edu` (Pavel Solin)

Generalized eigensolvers can change the order of eigenpairs, and for repeated eigenvalues, they can even return an arbitrary linear combination of eigenfunctions from the corresponding eigenspace for each call. But the most significant limitation of this approach is the fact that n different functions are approximated on the same finite element mesh. Namely, local refinements required to efficiently approximate one eigenfunction may not be needed at all for the remaining $n-1$ eigenfunctions. In the end, none of the eigenfunctions is approximated in an optimal way, because overall an unnecessarily large number of degrees of freedom is used.

The present work addresses this problem by approximating each eigenfunction on an individual higher-order finite element mesh which moreover is locally refined independently of each other, thus minimizing the number of degrees of freedom per eigenfunction. This is made possible by using the results in [14] where the problems associated with repeated calls to the generalized eigensolver were circumvented by only calling the generalized eigensolver once at the beginning of the computation, and then employing automatic *hp*-adaptivity combined with a Newton-like iterative method to pursue a single eigenpair on a sequence of locally refined *hp*-meshes.

The multimesh *hp*-FEM method was first introduced by Solin et al. in the context of multiphysics PDE problems [13, 3, 11]. These problems involve multiple physical fields which often exhibit vastly different behaviours. For instance, where one field is smooth (temperature), another one can have a singular gradient (displacement), singular values (electric or magnetic field, stress), or a boundary layer (flow velocity). Naturally, approximating all fields on the same finite element mesh would lead to a strongly over-refined mesh, for the same reasons as when using a single mesh to approximate n different eigenfunctions. Therefore, the multimesh *hp*-FEM approximates each physical field on an individual higher-order finite element mesh. These meshes are automatically adapted independently of each other, with the goal to approximate each physical field using as few degrees of freedom as possible. These meshes have a common, often extremely coarse, ancestor mesh called *master mesh* [13]. In most cases, the master mesh is not used for the actual discretization, but it helps to efficiently navigate a tree of locally refined meshes during automatic adaptivity. The corresponding algorithms are available in the open-source higher-order finite element library Hermes [16].

The methods presented in this work are developed starting from the results and methods in [14, 15]. The main contribution in those papers is the iterative solvers able to track multiple eigenpairs through the sequence of meshes created by the hp-adaptivity. This reduces the number of iterations in the iterative solvers and increases the convergence speed especially in case of multiple eigenvalues. The two solvers presented in [14, 15] are based on Picard's method and Newton's method that in the present work are modified to incorporate multimesh. The main difference between [14] and [15] is that in the former paper symmetric eigenvalue problems are studied and in the latter non-symmetric eigenvalue problems are considered. The limitation of the methods in both [14, 15] is that only one sequence of adaptively refined meshes can be used to approximate multiple eigenpairs. This is what prevents the methods to achieve even better converge rates in terms of DOF. This limitation is overcome in the present work.

2. Model Problem

To illustrate the advantages of the multimesh approach for solving eigenvalue problems, a well-known elliptic model problem is used:

$$-\Delta u = \lambda u \text{ in } \Omega, \quad u = 0 \text{ on } \partial\Omega \ . \tag{1}$$

This problem in a bounded polygonal domain Ω has a positive and discrete spectrum. We denote the eigenvalues of (1) by

$$0 < \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \lambda_4 \leq \lambda_5 \leq \dots \ ,$$

with possible repetition in case of repeated eigenvalues.

Nomenclature

$\mathcal{T}^c_{j,n}$	coarse mesh for eigenvalue j of refinement n
$\mathcal{S}^c_{j,n}(\mathcal{T}^c_{j,n})$	coarse FE space on $\mathcal{T}^c_{j,n}$
$\mathcal{T}_{j,n}$	mesh for eigenvalue j of refinement n
$\mathcal{S}_{j,n}(\mathcal{T}_{j,n})$	FE space on $\mathcal{T}_{j,n}$
$u_{j,n}$	eigenfunction computed on $\mathcal{S}_{j,n}(\mathcal{T}_{j,n})$
$\lambda_{j,n}$	eigenvalue computed on $\mathcal{S}_{j,n}(\mathcal{T}_{j,n})$
$P^{j,n}_{i,m}$	extension operator from $\mathcal{S}_{i,m}(\mathcal{T}_{i,m})$ onto $\mathcal{S}_{j,n}(\mathcal{T}_{j,n})$.
$R^{j,n}_{i,m}$	restriction operator from $\mathcal{S}_{i,m}(\mathcal{T}_{i,m})$ onto $\mathcal{S}_{j,n}(\mathcal{T}_{j,n})$.
$\mathcal{T}_n$	union mesh of all meshes $\mathcal{T}_{j,n}$ for all j
$\mathcal{S}_n(\mathcal{T}_n)$	union FE space of all spaces $\mathcal{S}_{j,n}(\mathcal{T}_{j,n})$ for all j
$P^{n}_{i,m}$	extension operator from $\mathcal{S}_{i,m}(\mathcal{T}_{i,m})$ onto $\mathcal{S}_n(\mathcal{T}_n)$.
$R^{i,m}_{n}$	restriction operator from $\mathcal{S}_n(\mathcal{T}_n)$ onto $\mathcal{S}_{i,m}(\mathcal{T}_{i,m})$.
$\mathbf{A}_{j,n}$	stiffness matrix for the space $\mathcal{S}_{j,n}(\mathcal{T}_{j,n})$
$\mathbf{A}_{n}$	stiffness matrix for the space $\mathcal{S}_n(\mathcal{T}_n)$
$\mathbf{B}_{j,n}$	mass matrix for the space $\mathcal{S}_{j,n}(\mathcal{T}_{j,n})$
$\mathbf{B}_{n}$	mass matrix for the space $\mathcal{S}_n(\mathcal{T}_n)$
$\mathbf{u}^m_{j,n}$	vector of approximated eigenfunction $u_{j,n}$ of iteration m
$\lambda^m_{j,n}$	approximation of $\lambda_{j,n}$ of iteration m

3. Theoretical Framework

Applying the standard FEM framework to Problem (1), we denote by $L^2(\Omega)$ the usual space of square-integrable real-valued functions equipped with the standard norm

$$\|f\|_0 \ := \ \left(\int_\Omega |f|^2\right)^{\frac{1}{2}} \ , \tag{2}$$

and by $H^1(\Omega)$ the usual space of functions in $L^2(\Omega)$ with square-integrable weak first partial derivatives. We denote by $H^1_0(\Omega)$ the subspace of $H^1(\Omega)$ of functions with zero trace on the boundary $\partial\Omega$. The weak formulation of Problem (1) reads:

Find n eigenpairs of the form $(\lambda_j, u_j) \in \mathbb{R} \times H^1_0(\Omega)$ such that

$$\left.\begin{array}{lcl} a(u_j, v) & = & \lambda_j \; b(u_j, v), \quad \text{for all} \quad v \in H^1_0(\Omega) \\ \|u_j\|_0 & = & 1 \end{array}\right\} \tag{3}$$

where

$$a(u, v) := \int_\Omega \nabla u \cdot \nabla v, \tag{4}$$

and

$$b(u, v) := \int_\Omega uv. \tag{5}$$

In this work, all meshes on Ω are assumed to be irregular with an arbitrary-level of hanging nodes [12]. For brevity, only 2D domains meshed with triangular elements are considered, but all the results also apply to 3D problems and to other types of finite elements such as quadrilaterals, prisms, bricks, etc.

Meshes on Ω are denoted by $\mathcal{T}$ with subscripts and occasionally superscripts. Higher-order finite element spaces constructed on a mesh $\mathcal{T}$ are denoted by $\mathcal{S}(\mathcal{T}) \subset H^1_0(\Omega)$, also with subscripts and occasionally superscripts. Automatic *hp*-adaptivity is used extensively in this work, therefore different orders of polynomials may be used on different elements in the same mesh. Adaptivity creates a sequence of meshes and finite element spaces, and to identify uniquely each mesh and finite element space in the sequence, a subscript n is used. Also, the key advantage in using multimesh techniques is that each eigenpair is computed on a different mesh with its own finite element space. Considering the eigenpair (λ_j, u_j) of Problem (1) and index j, the finite element space used to approximate it is denoted by $\mathcal{S}_{j,n}(\mathcal{T}_{j,n})$, where the subscript n refers to the position of the finite element space in the sequence created by the adaptivity process. Similarly, the computed eigenpair is denoted by $(\lambda_{j,n}, u_{j,n})$. Therefore, the discrete version of (3) reads:

Find n eigenpairs of the form $(\lambda_{j,n}, u_{j,n}) \in \mathbb{R} \times \mathcal{S}_{j,n}(\mathcal{T}_{j,n})$ such that

$$\left.\begin{array}{lcl} a(u_{jn}, v_n) & = & \lambda_{j,n} \; b(u_{j,n}, v_n), \quad \text{for all} \quad v_n \in \mathcal{S}_{j,n}(\mathcal{T}_{j,n}) \\ \|u_{j,n}\|_0 & = & 1 \end{array}\right\} \tag{6}$$

In the rest of the work, we apply the multimesh *hp*-FEM to compute several eigenpairs simultaneously but on different meshes. In some of the steps in the algorithms below, we need the union finite element spaces: Given a set of finite element spaces $\mathcal{S}_{j,n}(\mathcal{T}_{j,n})$ approximating different eigenpairs and from the same adaptive step n, the union finite element space $\mathcal{S}_n(\mathcal{T}_n)$ is constructed in such a way that each $\mathcal{S}_{j,n}(\mathcal{T}_{j,n})$ in the set is contained in it. This also implies that the mesh $\mathcal{T}_n$ is the union mesh of all meshes $\mathcal{T}_{j,n}$ in the set, i.e. $\mathcal{T}_n$ can be constructed starting from any of the $\mathcal{T}_{j,n}$ meshes by proper refinement. The union finite element space $\mathcal{S}_n(\mathcal{T}_n)$ can always be constructed because all locally refined meshes in the set originate from a common coarse *master mesh* [13].

4. Adaptive Multimesh *hp*-FEM Algorithm

In this section the *hp*-adaptive multimesh algorithm, Algorithm 3, is introduced. Due to the iterative nature of the algorithm, iterative solvers are preferable. In [15, 14], iterative solvers based on Picard's and Newton's method for eigenvalue problems are presented. These methods are adapted in the following to be used with the multimesh technique.

4.1. Picard's Method

The Picard's method is a simple iterative method based on solving a linear system multiple times with the right-hand side computed in the previous iteration. When applied to eigenvalue problems, the nature of the method is not changed at all, just the right-hand side is now depending on the eigenvalue. Problem (6) can be reformulated in matrix form as:

Find n eigenpairs of the form $(\lambda_{j,n}, \mathbf{u}_{j,n}) \in \mathbb{R} \times \mathbb{R}^N$, where N is the dimension of $\mathcal{S}_{j,n}(\mathcal{T}_{j,n})$, such that

$$\left.\begin{array}{lcl} \mathbf{A}_{j,n}\mathbf{u}_{j,n} & = & \lambda_{j,n}\mathbf{B}_{j,n}\mathbf{u}_{j,n} \ , \\ \mathbf{u}^t_{j,n}\mathbf{B}_{j,n}\mathbf{u}_{j,n} & = & 1 \end{array}\right\} \tag{7}$$

where the entries of the matrices $\mathbf{A}_{j,n}$ and $\mathbf{B}_{j,n}$ are

$$(\mathbf{A}_{j,n})_{k,p} := a(\phi_p, \phi_k) \ , \quad (\mathbf{B}_{j,n})_{k,p} := b(\phi_p, \phi_k) \ ,$$

where ϕ_i are the basis functions spanning $\mathcal{S}_{j,n}(\mathcal{T}_{j,n})$.

In [14], we presented the implementation of the Picard's method solving (7).

One of the main drawbacks of method in [14] is the fact that it always tends to convergence to the eigenpair with the smallest eigenvalue. This is annoying in the multimesh context because our aim is to be able to control which eigenpair is computed on each mesh. This problem is solved introducing an orthogonalization step to force the algorithm to converge to the correct eigenpair.

The Picard's method with orthogonalization, see Algorithm 1, takes as arguments the matrices $\mathbf{A}_{j,n}$ and $\mathbf{B}_{j,n}$ of (7), the matrices $\mathbf{A}_n$ and $\mathbf{B}_n$ constructed using the union finite element space $\mathcal{S}_n(\mathcal{T}_n)$ from all spaces $\mathcal{S}_{i,n}(\mathcal{T}_{i,n})$ with $i = 1, \ldots, j$, an initial guess $\tilde{u}_{j,n}$ for the eigenfunction, the tolerances AbsTol and Tol and it also takes the $j-1$ approximate eigenfunctions $u_{1,n}, \ldots, u_{j-1,n}$. Then it returns the approximate eigenpair $(\lambda_{j,n}, u_{j,n})$. Because we use this iterative method on a sequence of adaptively refined meshes, we normally set as initial guess the projection on the refined mesh of the eigenfunction of interest $u_{j,n-1}$ computed on the previous mesh in the adapted sequence,.

Algorithm 1 Multimesh Picard's method with orthogonalization

```
$(\lambda_{j,n}, u_{j,n})$ := MultimeshPicardOrtho($\mathbf{A}_{j,n}, \mathbf{B}_{j,n}, \mathbf{A}_n, \mathbf{B}_n, \tilde{u}_{j,n}, \{u_{i,n}\}_{i=1}^{j-1}$, Tol, AbsTol)
$\mathbf{u}_{j,n}^1 := \tilde{u}_{j,n}$
$\lambda_{j,n}^1 := \frac{(\mathbf{u}_{j,n}^1)^t \mathbf{A}_{j,n} \mathbf{u}_{j,n}^1}{(\mathbf{u}_{j,n}^1)^t \mathbf{B}_{j,n} \mathbf{u}_{j,n}^1}$
$m = 1$
repeat
    $\mathbf{u}_{j,n}^{m+1} := \mathbf{A}_{j,n}^{-1} \lambda_{j,n}^m \mathbf{B}_{j,n} \mathbf{u}_{j,n}^m$
    for $i = 1$ to $j-1$ do
        $\mathbf{u}_{j,n}^{m+1} := \mathbf{u}_{j,n}^{m+1} - R_n^{j,n}((P_{i,n}^n u_{i,n}^t \mathbf{B}_n P_{j,n}^n \mathbf{u}_{j,n}^{m+1}) P_{i,n}^n u_{i,n})$
                                                        ▷ Orthogonalization
    end for
    $\mathbf{u}_{j,n}^{m+1} = \frac{\mathbf{u}_{j,n}^{m+1}}{((\mathbf{u}_{j,n}^{m+1})^t \mathbf{B}_{j,n} \mathbf{u}_{j,n}^{m+1})^{1/2}}$       ▷ Normalization
    $\lambda_{j,n}^{m+1} := \frac{(\mathbf{u}_{j,n}^{m+1})^t \mathbf{A}_{j,n} \mathbf{u}_{j,n}^{m+1}}{(\mathbf{u}_{j,n}^{m+1})^t \mathbf{B}_{j,n} \mathbf{u}_{j,n}^{m+1}}$
    $m := m + 1$
until $\frac{\|\mathbf{u}_{j,n}^m - \mathbf{u}_{j,n}^{m-1}\|_1}{\|\mathbf{u}_{j,n}^{m-1}\|_1} <$ Tol or $|\lambda_{j,n}^m - \lambda_{j,n}^{m-1}| <$ AbsTol
$u_{j,n} := \mathbf{u}_{j,n}^m$
$\lambda_{j,n} := \lambda_{j,n}^m$
```

The orthogonalization step in Algorithm 1 differs from the one introduced in [14] because here the multimesh framework has to be taken into account. The computed eigenfunctions $u_{i,n}$ are computed on different meshes, therefore to compare them appropriate restriction and projection operators have to be used. The orthogonalization takes place in the union finite element space $\mathcal{S}_n(\mathcal{T}_n)$ which is finer than any of the spaces where $u_{i,n}$ belong. The projection of any eigenfunction $u_{i,n}$ onto $\mathcal{S}_n(\mathcal{T}_n)$ can be done without introducing any approximation error. The prolongation operator from a space $\mathcal{S}_{i,n}(\mathcal{T}_{i,n})$ onto $\mathcal{S}_n(\mathcal{T}_n)$ is denoted by $P_{i,n}^n$. Similarly, the restriction operator from $\mathcal{S}_n(\mathcal{T}_n)$ to the space $\mathcal{S}_{i,n}(\mathcal{T}_{i,n})$ is denoted by $R_n^{i,n}$.

4.2. Newton's Method

The second iterative method used in this work is based on the Newton's method presented in [14].

As in the case of the Picard's method, also the Newton's method needs an orthogonalization step compatible with the multimesh approach, see Algorithm 2.

Algorithm 2 Multimesh Newton's method with orthogonalization

```
$(\lambda_{j,n}, u_{j,n}) := \text{MultimeshNewtonOrtho}(\mathbf{A}_{j,n}, \mathbf{B}_{j,n}, \mathbf{A}_n, \mathbf{B}_n, \tilde{u}_{j,n}, , \{u_{i,n}\}_{i=1}^{j-1}, \text{Tol}, \text{AbsTol})$
$\mathbf{u}_{j,n}^1 := \tilde{u}_{j,n}$
$\lambda_{j,n}^1 := \frac{(\mathbf{u}_{j,n}^1)^t \mathbf{A}_{j,n} \mathbf{u}_{j,n}^1}{(\mathbf{u}_{j,n}^1)^t \mathbf{B}_{j,n} \mathbf{u}_{j,n}^1}$
$m = 1$
**repeat**
    Solve $J_f(\mathbf{u}_{j,n}^m, \lambda_{j,n}^m) \cdot \tilde{h} = -f(\mathbf{u}_{j,n}^m, \lambda_{j,n}^m)$
    $\mathbf{u}_{j,n}^{m+1} := \mathbf{u}_{j,n}^m + h$
    $\lambda_{j,n}^{m+1} := \lambda_{j,n}^m + \delta$
    **for** $i = 1$ to $j - 1$ **do**
        $\mathbf{u}_{j,n}^{m+1} := \mathbf{u}_{j,n}^{m+1} - R_n^{j,n}((P_{i,n}^n u_{i,n}^t \mathbf{B}_n P_{j,n}^n \mathbf{u}_{j,n}^{m+1}) P_{i,n}^n u_{i,n})$
                                                        ▷ Orthogonalization
    **end for**
    $\mathbf{u}_{j,n}^{m+1} = \frac{\mathbf{u}_{j,n}^{m+1}}{((\mathbf{u}_{j,n}^{m+1})^t \mathbf{B}_{j,n} \mathbf{u}_{j,n}^{m+1})^{1/2}}$          ▷ Normalization
    $\lambda_{j,n}^{m+1} := \frac{(\mathbf{u}_{j,n}^{m+1})^t \mathbf{A}_{j,n} \mathbf{u}_{j,n}^{m+1}}{(\mathbf{u}_{j,n}^{m+1})^t \mathbf{B}_{j,n} \mathbf{u}_{j,n}^{m+1}}$
    $m := m + 1$
**until** $\frac{\|\mathbf{u}_{j,n}^m - \mathbf{u}_{j,n}^{m-1}\|_1}{\|\mathbf{u}_{j,n}^{m-1}\|_1} < \text{Tol}$ **or** $|\lambda_{j,n}^m - \lambda_{j,n}^{m-1}| < \text{AbsTol}$
$u_{j,n} := \mathbf{u}_{j,n}^m$
$\lambda_{j,n} := \lambda_{j,n}^m$
```

4.3. Automatic Adaptivity

In this section we discuss the *hp*-adaptive multimesh algorithm, Algorithm 3. The purpose of this algorithm is to compute a selection of eigenpairs using multimesh *hp*-FEM combined with automatic *hp*-adaptivity, meaning that each eigenpair has its own sequence of adaptively refined *hp*-meshes. To start the algorithm, the indices of the eigenpairs of interest are inserted in the set $\mathcal{I}$ and the initial finite element space $\mathcal{S}_0^c(\mathcal{T}_0^c)$ is passed to the algorithm. The space $\mathcal{S}_0^c(\mathcal{T}_0^c)$ is the starting space for all eigenpairs. Algorithm 3 computes all eigenpairs up to the level of accuracy defined in Accuracy. Because different eigenpairs can require different numbers of adaptive steps to reach the specified accuracy, the algorithm removes the indices of the eigenpairs one-by-one as soon as they are approximated well enough, see line 27 in Algorithm 3. The algorithm stops only when $\mathcal{I}$ is empty, see line 32 in Algorithm 3. In this way no mesh is adapted more than necessary. As explained in Sections 4.1 and 4.2, the solvers need initial guesses for the eigenpairs to compute. In line 2 in Algorithm 3, a generalized eigensolver is called to produce initial guesses for the solvers called in lines 17 and 19. The generalised eigensolver used in the simulations is the eigenvalue solver in PySparse [6]. This solver is not designed for multiple meshes, but at the beginning of Algorithm 3 all eigenpairs are computed on $\mathcal{S}_0^c(\mathcal{T}_0^c)$, therefore a standard eigensolver is more than enough. The finite element spaces are refined using an algorithm from [17] that is an analogy to embedded higher-order ODE methods, see line 23-25. Such algorithm constructs an approximation pair of eigenfunctions with different orders of accuracy and uses their difference as an a-posteriori error estimator to guide *hp*-adaptivity.

As described in detail in [17], automatic *hp*-adaptivity is much more efficient than standard *h*-adaptivity. But on the other hand, it requires much more information about the error. An elementwise-constant error estimate is not enough to guide it. One needs an error estimate which provides information about the *shape of the error* in every element. This is exactly what the difference between eigenfunctions calculated with different orders of accuracy is used for. In every element the adaptivity algorithm considers a number of *competitive hp-refinements*, and chooses the one which, roughly speaking, minimizes the projection error while also not adding too many new degrees of freedom.

Algorithm 3 Adaptive Multimesh *hp*-FEM Algorithm

```
1:  Create the set $\mathcal{I}$ containing the indices of the eigenvalues to target and denote by $\mathcal{I}_{\max}$ the maximum value in $\mathcal{I}$.
2:  Let $\mathcal{T}_0^c$ be an initial coarse mesh and $\mathcal{S}_0^c(\mathcal{T}_0^c)$ the corresponding finite element space. A generalized eigensolver is called one time only, to obtain solution pairs $(\lambda_{j,0}^c, u_{j,0}^c)$, for $j = 1, \dots, \mathcal{I}_{\max}$, on the initial coarse mesh $\mathcal{T}_0^c$.
3:  Copies $\mathcal{S}_{j,0}^c(\mathcal{T}_{j,0}^c)$ of $\mathcal{S}_0^c(\mathcal{T}_0^c)$ are made for each value of $j$ from 1 to $\mathcal{I}_{\max}$.
4:  Set $k := 0$
5:  repeat
6:      Construct fine meshes $\mathcal{T}_{j,k}$ and spaces $\mathcal{S}_{j,k}(\mathcal{T}_{j,k})$, for $j = 1, \dots, \mathcal{I}_{\max}$, by refining all elements in space $\mathcal{S}_{j,k}^c(\mathcal{T}_{j,k}^c)$ and increasing their polynomial degrees by one.
7:      Assemble $\mathbf{A}_k$ and $\mathbf{B}_k$, the stiffness and mass matrices on the union mesh $\mathcal{T}_k$ and union space $\mathcal{S}_k(\mathcal{T}_k)$, $\mathcal{S}_k(\mathcal{T}_k)$ is the union of all $\mathcal{S}_{j,k}(\mathcal{T}_{j,k})$, for $j = 1, \dots, \mathcal{I}_{\max}$, respectively.
8:      for $j$ from 1 to $\mathcal{I}_{\max}$ do
9:          if $k == 0$ then
10:             Project the approximation $u_{j,0}^c$ onto the space $\mathcal{S}_{j,0}(\mathcal{T}_{j,0})$. The projection is denoted by $\tilde{u}_{j,0} := P_c^{j,0} u_{j,0}^c$. Since the finite element spaces $\mathcal{S}_{j,0}^c(\mathcal{T}_{j,0}^c)$ and $\mathcal{S}_{j,0}(\mathcal{T}_{j,0})$ are embedded, there is no projection error.
11:         else
12:             Project the approximation $u_{j,k-1}$ onto the space $\mathcal{S}_{j,k}(\mathcal{T}_{j,k})$. The projection is denoted by $\tilde{u}_{j,k} := P_{j,k-1}^{j,k} u_{j,k-1}$.
13:         end if
14:         Assemble $\mathbf{A}_{j,k}$ and $\mathbf{B}_{j,k}$, the stiffness and mass matrices on the spaces $\mathcal{S}_{j,k}(\mathcal{T}_{j,k})$, respectively.
15:         Calculate an initial guess $\tilde{\lambda}_{j,k}$ for the eigenvalue on the space $\mathcal{S}_{j,k}(\mathcal{T}_{j,k})$ using the relation
                $$\tilde{\lambda}_{j,k} = \frac{(\tilde{u}_{j,k})^T \mathbf{A}_{j,k} \tilde{u}_{j,k}}{(\tilde{u}_{j,k})^T \mathbf{B}_{j,k} \tilde{u}_{j,k}}.$$
            The pair $(\tilde{\lambda}_{j,k}, \tilde{u}_{j,k})$ is **not a solution** to the generalized eigenproblem on the space $\mathcal{S}_{j,k}(\mathcal{T}_{j,k})$, but it is used as an initial guess.
```

```
16:         if Picard's method is selected then
17:             $(\lambda_{j,k}, u_{j,k}) := \text{MultimeshPicardOrtho}(\mathbf{A}_{j,k}, \mathbf{B}_{j,k}, \mathbf{A}_k, \mathbf{B}_k, \tilde{u}_{j,k}, \{u_{i,k}\}_{i=1}^{j-1}, \text{Tol}, \text{AbsTol})$
18:         else
19:             $(\lambda_{j,k}, u_{j,k}) := \text{MultimeshNewtonOrtho}(\mathbf{A}_{j,k}, \mathbf{B}_{j,k}, \mathbf{A}_k, \mathbf{B}_k, \tilde{u}_{j,k}, \{u_{i,k}\}_{i=1}^{j-1}, \text{Tol}, \text{AbsTol})$
20:         end if
21:         if $j \in \mathcal{I}$ then
22:             Project the approximation $u_{j,k}$ back to the coarse space $\mathcal{S}^c_{j,k}(\mathcal{T}^c_{j,k})$
    to obtain $P^{j,k}_c u_{j,k}$.
23:             Calculate the a-posteriori error estimate $e_{j,k}$,

                    $$e_{j,k} = u_{j,k} - P^{j,k}_c u_{j,k}.$$

    Note: $e_{j,k}$ is a function, not a number.
24:             if $\|e_{j,k}\|_0 >$ Accuracy then
25:                 Use $e_{j,k}$ to guide one step of automatic hp-adaptivity [17]
    that yields a new coarse $\mathcal{S}^c_{j,k+1}(\mathcal{T}^c_{j,k+1})$.
26:             else
27:                 Remove $j$ from $\mathcal{I}$.
28:             end if
29:         end if
30:     end for
31:     Update $k := k + 1$
32: until $\mathcal{I}$ is empty.
```

5. Numerical Results

In this section we compare the adaptive multimesh *hp*-FEM method, Algorithm 3, with a standard adaptive *hp*-FEM method, Algorithm 6, where all eigenfunctions are approximated on the same locally refined automatically adapted mesh.

The structure of Algorithm 6 is similar to the structure of Algorithm 3 except for two important features: there is no multimesh, all eigenpairs are computed on the same adapted mesh, and the mesh is adapted using information from the a-posteriori error estimates from all considered eigenpairs. As in Algorithm 3, both Picard's and Newton's method solvers are available in Algorithm 6, see Algorithm 4 and Algorithm 5 respectively. Both methods use the orthogonality technique to keep the eigenpairs well separated.

Algorithm 4 Picard's method with orthogonalization

```
$(\lambda_{j,n}, u_{j,n}) := \text{PicardOrtho}(\mathbf{A}_n, \mathbf{B}_n, \tilde{u}_{j,n}, \{u_{i,n}\}_{i=1}^{j-1}, \text{Tol}, \text{AbsTol})$
$\mathbf{u}_{j,n}^1 := \tilde{u}_{j,n}$
$\lambda_{j,n}^1 := \dfrac{(\mathbf{u}_{j,n}^1)^t \mathbf{A}_n \mathbf{u}_{j,n}^1}{(\mathbf{u}_{j,n}^1)^t \mathbf{B}_n \mathbf{u}_{j,n}^1}$
$m = 1$
**repeat**
    $\mathbf{u}_{j,n}^{m+1} := \mathbf{A}_n^{-1} \lambda_{j,n}^m \mathbf{B}_n \mathbf{u}_{j,n}^m$
    **for** $i = 1$ to $j-1$ **do**
        $\mathbf{u}_{j,n}^{m+1} := \mathbf{u}_{j,n}^{m+1} - (u_{i,n}^t \mathbf{B}_n \mathbf{u}_{j,n}^{m+1}) u_{i,n}$
                                                            ▷ Orthogonalization
    **end for**
    $\mathbf{u}_{j,n}^{m+1} = \dfrac{\mathbf{u}_{j,n}^{m+1}}{((\mathbf{u}_{j,n}^{m+1})^t \mathbf{B}_n \mathbf{u}_{j,n}^{m+1})^{1/2}}$     ▷ Normalization
    $\lambda_{j,n}^{m+1} := \dfrac{(\mathbf{u}_{j,n}^{m+1})^t \mathbf{A}_n \mathbf{u}_{j,n}^{m+1}}{(\mathbf{u}_{j,n}^{m+1})^t \mathbf{B}_n \mathbf{u}_{j,n}^{m+1}}$
    $m := m + 1$
**until** $\frac{\|\mathbf{u}_{j,n}^m - \mathbf{u}_{j,n}^{m-1}\|_1}{\|\mathbf{u}_{j,n}^{m-1}\|_1} < \text{Tol}$ **or** $|\lambda_{j,n}^m - \lambda_{j,n}^{m-1}| < \text{AbsTol}$
$u_{j,n} := \mathbf{u}_{j,n}^m$
$\lambda_{j,n} := \lambda_{j,n}^m$
```

Algorithm 5 Newton's method with orthogonalization

```
$(\lambda_{j,n}, u_{j,n}) := \text{NewtonOrtho}(\mathbf{A}_n, \mathbf{B}_n, \tilde{u}_{j,n}, \{u_{i,n}\}_{i=1}^{j-1}, \text{Tol}, \text{AbsTol})$
$\mathbf{u}_{j,n}^1 := \tilde{u}_{j,n}$
$\lambda_{j,n}^1 := \dfrac{(\mathbf{u}_{j,n}^1)^t \mathbf{A}_n \mathbf{u}_{j,n}^1}{(\mathbf{u}_{j,n}^1)^t \mathbf{B}_n \mathbf{u}_{j,n}^1}$
$m = 1$
repeat
    Solve $J_f(\mathbf{u}_{j,n}^m, \lambda_{j,n}^m) \cdot \tilde{h} = -f(\mathbf{u}_{j,n}^m, \lambda_{j,n}^m)$
    $\mathbf{u}_{j,n}^{m+1} := \mathbf{u}_{j,n}^m + h$
    $\lambda_{j,n}^{m+1} := \lambda_{j,n}^m + \delta$
    for $i = 1$ to $j - 1$ do
        $\mathbf{u}_{j,n}^{m+1} := \mathbf{u}_{j,n}^{m+1} - (u_{i,n}^t \mathbf{B}_n \mathbf{u}_{j,n}^{m+1}) u_{i,n}$
                                                        ▷ Orthogonalization
    end for
    $\mathbf{u}_{j,n}^{m+1} = \dfrac{\mathbf{u}_{j,n}^{m+1}}{((\mathbf{u}_{j,n}^{m+1})^t \mathbf{B}_n \mathbf{u}_{j,n}^{m+1})^{1/2}}$          ▷ Normalization
    $\lambda_{j,n}^{m+1} := \dfrac{(\mathbf{u}_{j,n}^{m+1})^t \mathbf{A}_n \mathbf{u}_{j,n}^{m+1}}{(\mathbf{u}_{j,n}^{m+1})^t \mathbf{B}_n \mathbf{u}_{j,n}^{m+1}}$
    $m := m + 1$
until $\frac{\|\mathbf{u}_{j,n}^m - \mathbf{u}_{j,n}^{m-1}\|_1}{\|\mathbf{u}_{j,n}^{m-1}\|_1} < \text{Tol}$ or $|\lambda_{j,n}^m - \lambda_{j,n}^{m-1}| < \text{AbsTol}$
$u_{j,n} := \mathbf{u}_{j,n}^m$
$\lambda_{j,n} := \lambda_{j,n}^m$
```

Algorithm 6 Standard (Single-Mesh) Adaptive *hp*-FEM Algorithm

```
1: Create the set $\mathcal{I}$ containing the indices of the eigenvalues to target and denote by $\mathcal{I}_{\max}$ the maximum value in $\mathcal{I}$.
2: Let $\mathcal{T}_0^c$ be an initial coarse mesh and $\mathcal{S}_0^c(\mathcal{T}_0^c)$ the corresponding finite element space. A generalized eigensolver is called one time only, to obtain a solution pairs $(\lambda_{j,0}^c, u_{j,0}^c)$, for $j = 1, \ldots, \mathcal{I}_{\max}$, on the initial coarse mesh $\mathcal{T}_0^c$.
3: Set $k := 0$
4: repeat
5:     Construct fine mesh $\mathcal{T}_k$ and space $\mathcal{S}_k(\mathcal{T}_k)$, by refining all elements in space $\mathcal{S}_k^c(\mathcal{T}_k^c)$ and moreover increasing their polynomial degrees by one.
6:     Assemble $\mathbf{A}_k$ and $\mathbf{B}_k$, the stiffness and mass matrices on the mesh $\mathcal{T}_k$ and space $\mathcal{S}_k(\mathcal{T}_k)$, respectively.
7:     for $j$ from 1 to $j_{\max}$ do
8:         if $k == 0$ then
9:             Project the approximation $u_{j,0}^c$ to the space $\mathcal{S}_0(\mathcal{T}_{j,0})$. The projection is denoted by $\tilde{u}_{j,0} := P_c^0 u_{j,0}^c$. Since the finite element spaces $\mathcal{S}_0^c(\mathcal{T}_0^c)$ and $\mathcal{S}_0(\mathcal{T}_0)$ are embedded, there is no projection error.
10:        else
11:            Project the approximation $u_{j,k-1}$ to the space $\mathcal{S}_k(\mathcal{T}_k)$. The projection is denoted by $\tilde{u}_{j,k} := P_{k-1}^k u_{j,k-1}$.
12:        end if
13:        Calculate an initial guess $\tilde{\lambda}_{j,k}$ for the eigenvalue on the space $\mathcal{S}_k(\mathcal{T}_k)$ using the relation
               $$\tilde{\lambda}_{j,k} = \frac{(\tilde{u}_{j,k})^T \mathbf{A}_k \tilde{u}_{j,k}}{(\tilde{u}_{j,k})^T \mathbf{B}_k \tilde{u}_{j,k}}.$$
           The pair $(\tilde{\lambda}_{j,k}, \tilde{u}_{j,k})$ is not a solution to the generalized eigenproblem on the space $\mathcal{S}_k(\mathcal{T}_k)$, but it is used as an initial guess.
```

```
14:         if Picard's method is selected then
15:             $(\lambda_{j,k}, u_{j,k}) := \text{PicardOrtho}(\mathbf{A}_k, \mathbf{B}_k, \tilde{u}_{j,k}, \{u_{i,k}\}_{i=1}^{j-1}, \text{Tol}, \text{AbsTol})$
16:         else
17:             $(\lambda_{j,k}, u_{j,k}) := \text{NewtonOrtho}(\mathbf{A}_k, \mathbf{B}_k, \tilde{u}_{j,k}, \{u_{i,k}\}_{i=1}^{j-1}, \text{Tol}, \text{AbsTol})$
18:         end if
19:     end for
20:     for $j \in \mathcal{I}$ do
21:         Project the approximation $u_{j,k}$ back to the coarse space $\mathcal{S}_k^c(\mathcal{T}_k^c)$ to
    obtain $P_c^k u_{j,k}$.
22:         Calculate the a-posteriori error estimate $e_{j,k}$,

                    $e_{j,k} = u_{j,k} - P_c^k u_{j,k}.$

    Note: $e_{j,k}$ is a function, not a number.
23:     end for
24:     Use all $\|e_{j,k}\|_0$, for all $j \in \mathcal{I}$ to automatically perform $hp$-adaptivity [17]
    that yields a new coarse $\mathcal{S}_{k+1}^c(\mathcal{T}_{k+1}^c)$
25:     Update $k := k + 1$
26: until All $\|e_{j,k}\|_0 <$ Accuracy for all $j \in \mathcal{I}$.
```

5.1. L-shaped domain: the simple eigenvalues case

We start considering Problem (1) on a L-shaped domain. This is a very common test case used very often in literature because the re-entering corner introduces additional numerical difficulties. It is well known that eigenfunctions for this problem may lack in regularity, i.e. regularity less than $H^2(\Omega)$. The lack of regularity is addressed with $hp-$adaptivity imposing a larger proportion of $h-$refinement and less $p-$refinement. Concentration of $h-$refinement where the eigenfunctions are less regular, i.e. around the re-entering corner for the L-shaped domain, regains some of the convergence speed lost by the lack of regularity. The spectrum of the L-shaped domain contains eigenfunctions with different regularity and some of them are smooth, i.e. regularity at least in $H^2(\Omega)$. For such eigenfunctions, $p-$refinement is preferable to $h-$refinement. Clearly, this is the dilemma facing standard $hp-$adaptive methods on this problem. If eigenfunctions with different level of regularity have to be computed just adapting one single common mesh, the mesh is not optimal for all of them. Higher p-elements, which are favourable for more regular eigenfunctions, do not help very much with irregular eigenfunctions. On the other hand, $h-$refinement concentrated around the re-entering corner which is necessary for irregular eigenfunctions has very little use for regular eigenfunctions. With multimesh, this dilemma is avoided because each mesh is adapted considering only one eigenfunction.

A good example of what has just been explained, can be seen considering the first two smallest eigenpairs of the L-shaped domain problem, see Figure 1, both of them are simple eigenpairs. The first eigenfunction is an irregular one with a lack of regularity at the re-entering corner. The second one is more regular but still not completely smooth, see [8] for a detailed analysis on the regularity of the eigenfunctions for the L-shaped domain. The difference in the regularity between the first two eigenfunctions is noticed by the adaptive procedure and exploited using different level of refinement for h and p. In Figure 1, the adapted meshes for the two methods for an approximation error of about 0.1% are compared. Comparing Figures 1(c) and 1(d) is clear that in Figure 1(c) the adaptivity focused heavily with $h-$refinement around the re-entering corner where the singularity in the gradient of the solution exists. In contrast, in Figure 1(d) only p-refinement is applied which is more efficient for more regular solutions. The meshes in Figure 1(e,f) are computed by the Single-Mesh method trying to resolve both eigenfunctions, therefore they are the same mesh. The result is a mix of the previous two meshes with heavily $h-$refinement around the re-entering corner and more spread high order elements across the entire domain. Numerically this mesh is not as good as the meshes in Figures 1(c) and 1(d), this can be seen from the convergence curves in Figure 2.

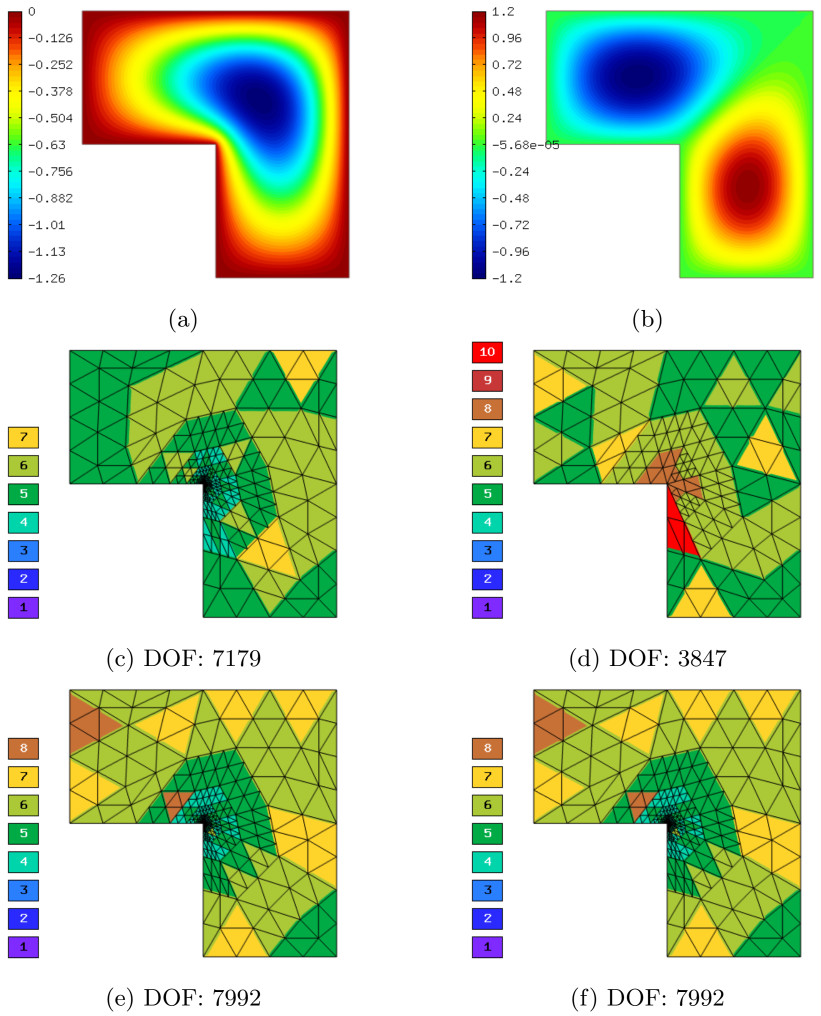


Figure 1: Corresponding eigenfunctions to the first two smallest eigenvalues of the L-shaped domain and relative meshes. The first eigenfunction (a) is non-regular. The second eigenfunction (b) is regular. Adapted meshes for the multimesh and the Single-Mesh methods for an accuracy of about 0.1%. Meshes (c,d) computed using the multimesh method and adapted for each of the two eigenfunctions. Meshes (e,f) computed using the standard method and adapted simultaneously for the two eigenfunctions. For each mesh, the number of DOF is stated.

The differences in convergence rates between the multimesh $hp-$adaptive method, Algorithm 3, and the standard $hp-$adaptive method, Algorithm 6, are presented in Figure 2. For this comparison the Picard's solver has been used for both algorithm with relative tolerance of $1e-8$ and absolute tolerance of $1e-9$. In Figure 2, the error is relative and given in percentage and is computed using the automatic $hp-$adaptive technique in [17]. Algorithm 3 is used to compute simultaneously the first two eigenfunctions. Algorithm 6 is used a first time to also compute simultaneously the first two eigenfunctions, the corresponding curves in Figure 2 are marked as "Single-Mesh". Then, Algorithm 6 is also run two more times to compute each eigenfunction in isolation, the corresponding curves in Figure 2 are marked as "Single-Eigenpair". First, it is interesting to notice that the convergence curves of Algorithm 3 overlap exactly the convergence curves of Algorithm 6 when is used to compute each eigenfunction in isolation. This suggests that the multimesh technique does not deteriorate at all the efficiency of the underlying $hp-$adaptive method and each mesh is adapted as well as possible for each eigenfunction. Secondly, the convergence curves of Algorithm 3 have completely independent number of DOF, this is highlighting the nature of the multimesh technology that allow for several meshes to be adapted independently. In fact, the mesh for the first eigenfunction is adapted 11 times while the mesh for the second eigenfunction is adapted only 6 times to reach the accuracy of at least 0.5%. Thanks to the multimesh freedom, each eigenfunction can be computed only up to the required accuracy without wasting extra computational time. Thirdly, it is worth to notice that the convergence curves of Algorithm 6, when is used to compute both eigenfunctions simultaneously, i.e. "Single-Mesh" curves, are always above the convergence curves of Algorithm 3 showing the higher efficiency in term of number of DOF of the multimesh method. Finally, it is also interesting the difference in convergence speed between the first and the second eigenpair, regardless of the method used for the computation, this is due to the difference in the regularity of the eigenpairs. In general, smoother a function is less DOF are needed for the adaptive process to well approximate it.

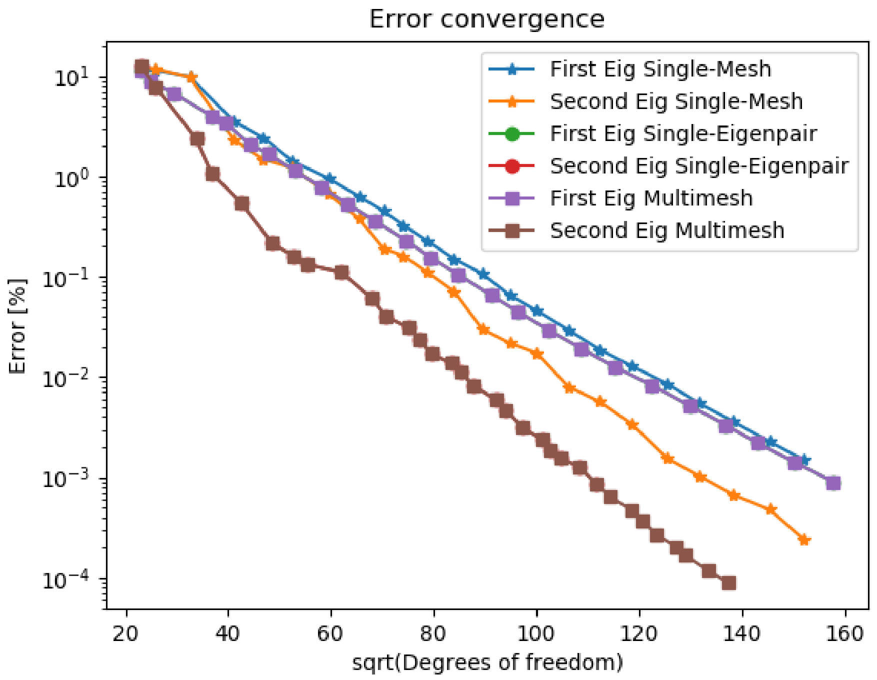


Figure 2: Convergence plot for the first two eigenfunctions of the L-shaped domain.

In Table 1, a subsection of the results presented in Figure 2 are also presented. In tabular form it is easier to appreciate the higher efficiency of the multimesh method in terms of number of DOF. Algorithm 3 reaches the requested accuracy of 0.5% far before the Single-Mesh method.

	Multimesh				Single-Mesh		
	Eigenpair 1		Eigenpair 2				
Step	DOF	Err.(%)	DOF	Err.(%)	DOF	$e_{1,k}$(%)	$e_{2,k}$(%)
1	529	11.397	529	12.435	529	11.397	12.435
2	622	8.876	673	7.738	663	11.402	11.733
3	859	6.785	1142	2.371	1060	9.808	9.715
4	1358	3.969	1352	1.082	1696	3.515	2.298
5	1571	3.385	1804	0.548	2190	2.442	1.482
6	1961	2.080	2360	0.216	2763	1.413	1.214
7	2287	1.681			3539	0.955	0.668
8	2810	1.131			4286	0.631	0.379
9	3384	0.779			4936	0.450	0.190
10	4004	0.529			5491	0.326	0.162
11	4683	0.364					

Table 1: Comparison between Algorithm 3 and Algorithm 6 in terms of DOF and accuracy.

Just for comparison, the L-shaped domain example is solved using only h-refinement which is a popular refinement technique in literature [9, 18] easier to implement than hp-refinement. However, the difference in performances is dramatic. In Figure 3, the convergence plots for the same methods used in Figure 2 allowing only for h-refinement and with constant order of polynomials equal to 3 everywhere in the meshes. Cubic order elements are used because the results using lower-order elements, i.e. linear or quadratic, are so poor that the comparison with hp-refinement would not be meaningful. Figures 2 and 3 are both plotted in semilog scale for easier comparison. It is clear the inefficiency of h-refinement compared to hp-refinement looking at the range of values along the axes, h-refinement achieves much worse accuracy using many more degrees of freedom. For this reason, only hp-refinement is considered in the rest of the paper.

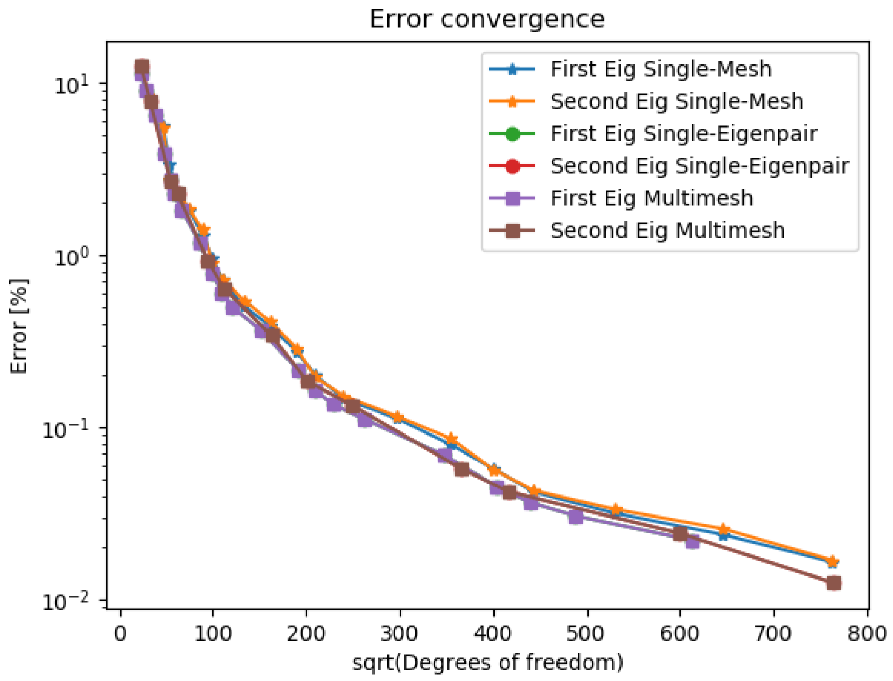
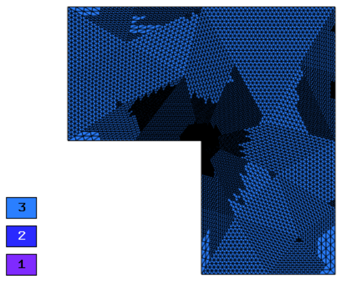
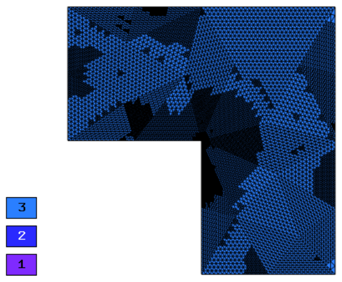


Figure 3: Convergence plot for the first two eigenfunctions of the L-shaped domain using *h*-refinement only.

In Figure 4 the adapted meshes using only *h*-refinement for the multimesh method and approximating the eigenfunctions to an accuracy around 0.1% are presented. The same accuracy is reached using *hp*-refinement on the meshes in Figure 1. The difference in number of degrees of freedom is enormous, confirming the superiority of *hp*-refinement.



(a) DOF: 68956

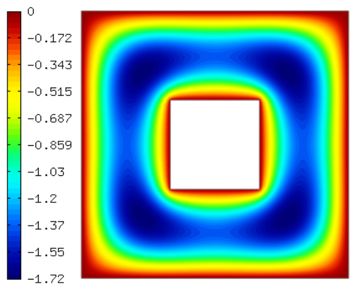


(b) DOF: 62239

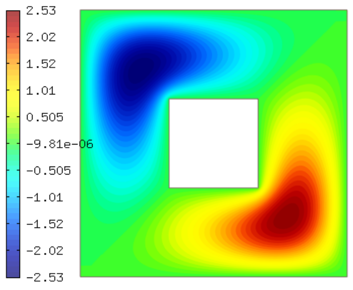
Figure 4: Adapted meshes for the multimesh for an accuracy of about 0.1% using only h-refinement. For each mesh, the number of DOF is stated.

5.2. Square-hole domain: the multiple eigenvalues case

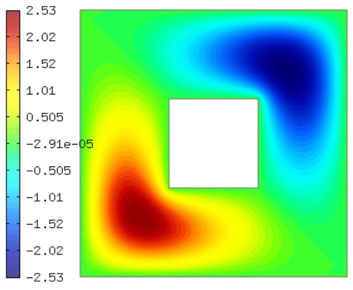
We consider now Problem (1) on a square domain with a centred square hole. Due to the presence of four re-entering corners, also the spectrum of this problem contains non-regular solutions. This case is particularly interesting because the second and the third eigenvalues form a multiple eigenvalue of multiplicity two. For this reason, we are going to consider the first three smallest eigenvalues of the problem, see Figure 5(a-c), all of them are non-regular. In Figure 5(d-f), the meshes computed by the Single-Mesh method trying to resolve all three eigenfunctions at the same time. This forces the method to use the same mesh for all eigenpairs, therefore all the meshes in Figure 5(d-f) are identical.



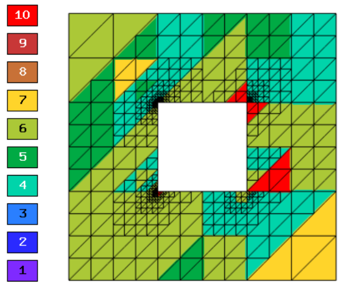
(a)



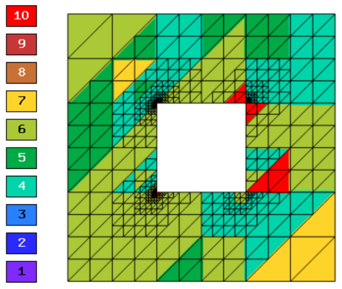
(b)



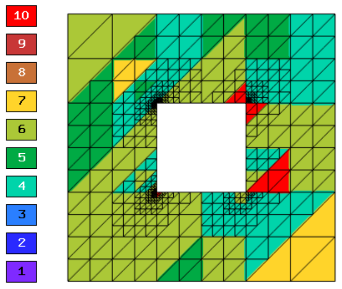
(c)



(d) DOF: 12827



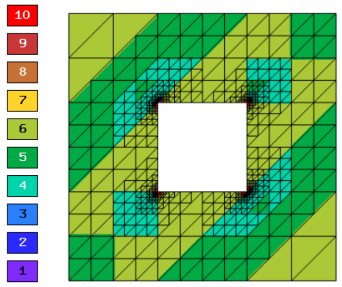
(e) DOF: 12827



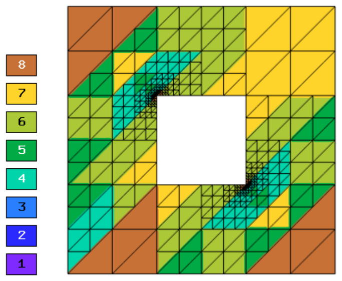
(f) DOF: 12827

Figure 5: First three eigenfunctions of the square-hole domain and final meshes for the Single-Mesh method. The first eigenfunction (a) is a simple one. The second (b) and the third (c) eigenfunctions belong to an eigenvalue of multiplicity two. The meshes for the first eigenfunction (d), for the second (e) and for the third (f). For each mesh, the number of DOF is stated.

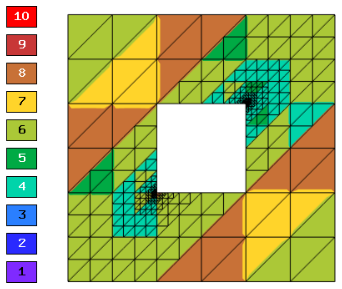
In Figure 6, the final meshes for Algorithm 3 are presented. Since the meshes have been refined independently, different refinement patterns are visible. The first eigenfunction has singularities in the gradient in all four corners, in Figure 6(a) is clearly visible heavily $h-$refinement around all four corners. The second and the third eigenfunctions are related by a rotation-symmetry since they belong to the same eigenspace. Due to the rotation, they have singularities in the gradient in distinct corners. Looking at Figures 6(b) and 6(c) it is clear that the re-entering corners targeted by $h-$refinement correspond to the corners where each eigenfunction has lack of regularity. It is important to notice that DOF numbers in Figure 5 and Figure 6 cannot be compared directly, since the accuracy is different, see Figure 7. Mesh (a) in Figure 6 has 14892 DOF and the accuracy is 0.457%. For comparison, mesh (a) in Figure 5 has only 12827 DOF but the accuracy is 1.06% which is far worse.



(a) DOF: 14892



(b) DOF: 9948



(c) DOF: 8159

Figure 6: Final adapted meshes for the multimesh methods for the first (a), second (b) and third (c) eigenvalues. For each mesh, the number of DOF is stated.

In Figure 7, the convergence rates of the multimesh $hp-$adaptive method, Algorithm 3, and of the standard $hp-$adaptive method, Algorithm 6, are compared. As in the previous case, the curves marked as "Single-Mesh" are computed using Algorithm 6 on all three eigenfunctions simultaneously. The curves marked as "Single-Eigenpair" are computed using Algorithm 6 on each of the three eigenfunctions individually. Again we can see that the curves produced by Algorithm 3 overlap exactly the curves computed by Algorithm 6 on each of the three eigenfunctions individually. Also, the curves computed by Algorithm 6 on all three eigenfunctions simultaneously are always above the curves computed by Algorithm 3. Finally, as in Figure 2, the second and third eigenpairs converge faster than the first one. This is due to the difference in the regularity between the first eigenpair and the other two. The second and the third eigenvectors exhibit only two singularities each at the re-entering corners where the first one has four singularities. There is also a slight difference between the convergence curves for the second and third eigenpair. This discrepancy is not due to a difference in the regularity since these two eigenpairs belong to the same eigenvalue, but to properties of the meshes. The initial mesh is a structured mesh with all the inclined edges oriented in the same direction. The adaptivity process preserves such orientation. It is natural that such structure in the meshes is more favourable to approximate one of the two eigenpairs.

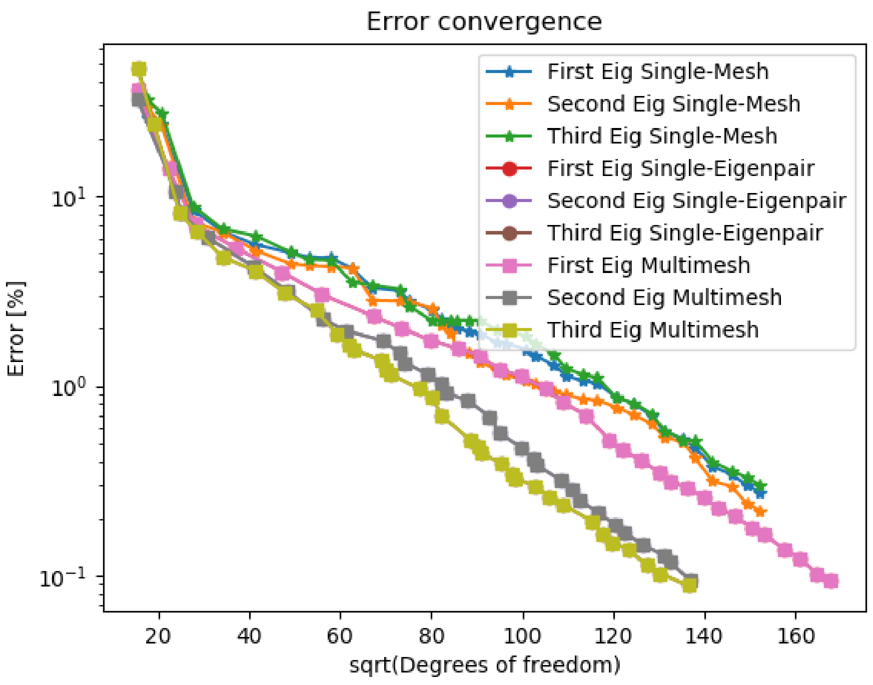


Figure 7: Convergence plot for the first three smallest eigenfunctions of the square-hole domain.

In Table 2, the same results presented in Figure 7 are also presented. In tabular form it is easier to appreciate the higher efficiency of the multimesh method in terms of number of DOF. Algorithm 3 reach the requested accuracy of 0.5% in far less adaptive steps compared to Algorithm 6. Not even after 26 adaptive steps, Algorithm 6 has reached the requested accuracy.

	Multimesh						Standard Reference			
	Eig. 1		Eig. 2		Eig. 3					
Step	DOF	Err.	DOF	Err.	DOF	Err.	DOF	$e_{1,k}$	$e_{2,k}$	$e_{3,k}$
1	240	36.4	240	32.3	240	47.0	240	36.4	32.3	47.0
2	495	14.0	569	10.6	362	24.2	314	26.2	29.6	32.3
3	783	7.12	624	8.11	614	8.13	430	24.1	23.6	27.5
4	1355	5.36	957	6.05	804	6.54	766	8.43	7.31	8.88
5	2230	3.98	1681	4.23	1168	4.78	1166	6.53	6.47	6.73
6	3129	3.04	2355	3.15	1700	4.05	1714	5.55	5.17	6.18
7	4518	2.34	3139	2.26	2293	3.09	2406	5.04	4.41	5.08
8	5382	2.02	3741	1.96	2997	2.50	2808	4.78	4.30	4.64
9	6359	1.74	4830	1.73	3615	1.79	3354	4.72	4.26	4.61
10	7387	1.57	5330	1.48	3828	1.64	3921	4.17	4.19	3.53
11	8247	1.44	5495	1.32	3965	1.55	4500	3.28	2.83	3.40
12	9019	1.22	6292	1.15	4746	1.37	5326	3.17	2.80	3.24
13	9971	1.12	6760	1.04	4917	1.22	5654	2.82	2.77	2.62
14	11042	0.97	6942	0.91	5041	1.15	6446	2.52	2.57	2.21
15	11848	0.82	7732	0.84	5989	0.97	6773	2.25	2.10	2.22
16	12994	0.70	8608	0.69	6415	0.87	7115	2.15	1.91	2.22
17	14194	0.52	9026	0.57	6775	0.70	7321	2.00	1.59	2.22
18	14892	0.46	9948	0.47	7847	0.52	7768	1.96	1.49	2.22
19					8159	0.48	8248	1.88	1.34	2.21
20							8902	1.70	1.21	1.98
21							9322	1.67	1.14	1.97
22							10150	1.56	1.06	1.83
23							10599	1.44	1.03	1.67
24							11443	1.29	0.93	1.47
25							12010	1.13	0.90	1.24
26							12827	1.07	0.85	1.15

Table 2: Comparison between Algorithm 3 and Algorithm 6 in terms of DOF and accuracy. All errors are relative and given in percentage.

6. Conclusions

We presented a novel adaptive multimesh *hp*-FEM method for the solution of elliptic eigenproblems where all eigenfunctions are approximated on individual meshes which moreover are adaptively refined independently of each other. The multimesh approach was compared to a standard approach where all eigenfunctions are approximated on the same mesh. Numerical experiments confirmed that the multimesh approach can lead to a significant reduction of the number of degrees of freedom per eigenfunction.

[1] M.G. Armentano, C. Padra, R. Rodríguez, and M. Scheble. An hp finite element adaptive scheme to solve the laplace model for fluidsolid vibrations. *Computer Methods in Applied Mechanics and Engineering*, 200(1):178–188, 2011.

[2] K. Andrew Cliffe, Edward J. C. Hall, and Paul Houston. Adaptive discontinuous galerkin methods for eigenvalue problems arising in incompressible fluid flows. *SIAM Journal on Scientific Computing*, 31(6):4607–4632, 2010.

[3] L. Dubcova, P. Solin, G. Hansen, and H. Park. Comparison of multimesh hp-FEM to interpolation and projection methods for spatial coupling of reactor thermal and neutron diffusion calculations. *J. Comput. Phys.*, 230:1182–1197, 2011.

[4] R. G. Durn, C. Padra, and R. Rodríguez. A posteriori error estimates for the finite element approximation of eigenvalue problems. *Math. Mod. & Met.in Appl. Sc.*, 13(8):1219–1229, 2003.

[5] C. Engström, S. Giani, and L. Grubišić. Efficient and reliable hp-FEM estimates for quadratic eigenvalue problems and photonic crystal applications. *Computers & Mathematics with Applications*, 72(4):952 – 973, 2016.

[6] R. Geus and P. Arbenz. Pysparse and pyfemax: A python framework for large scale sparse linear algebra.

[7] S. Giani, L. Grubišić, and J. S. Ovall. Benchmark results for testing adaptive finite element eigenvalue procedures part 2 (conforming eigenvector and eigenvalue estimates). *Applied Numerical Mathematics*, 102:1 – 16, 2016.

[8] J. Gopalakrishnan, L. Grubišić, and J.Ovall. Spectral discretization errors in filtered subspace iteration. *Math. Comp.*, 89(321):203–228.

[9] V. Heuveline and R. Rannacher. A posteriori error control for finite element approximations of elliptic eigenvalue problems. *Advances in Computational Mathematics*, 15(1):107–138, 2001.

[10] N Lalor and H.H. Priebsch. The prediction of low- and mid-frequency internal road vehicle noise: a literature survey. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 221(3):245–269, 2007.

[11] D. Pugal, P. Solin, K.J. Kim, and A. Aabloo. Modeling ionic polymer-metal composites with space-time adaptive multimesh hp-FEM. *Commun. Comput. Phys.*, 11:249–270, 2012.

[12] P. Solin, J. Cerveny, and I. Dolezel. Arbitrary-level hanging nodes and automatic adaptivity in the hp-FEM. *Math. Comput. Simul.*, 77:117 – 132, 2008.

[13] P. Solin, J. Cerveny, L. Dubcova, and D. Andrs. Monolithic discretization of linear thermoelasticity problems via adaptive multimesh hp-FEM. *J. Comput. Appl. Math.*, 234:2350 – 2357, 2010.

[14] P. Solin and S. Giani. An iterative adaptive finite element method for elliptic eigenvalue problems. *J. Comput. Appl. Math.*, 236(18):4582 – 4599, 2012.

[15] P. Solin and S. Giani. An iterative adaptive hp-FEM method for non-symmetric elliptic eigenvalue problems. *Computing*, 95(1):183213, 2013.

[16] P. Solin, L. Korous, and P. Kus. Hermes2D, a C++ library for rapid development of adaptive hp-FEM and hp-DG solvers. *J. Comput. Appl. Math.*, 270:152 – 165, 2014.

[17] P. Solin, K. Segeth, and I. Dolezel. *Higher-order finite element methods.* Chapman & Hall, CRC Press, 2003.

[18] T. F. Walsh, G. M. Reese, and U. L. Hetmaniuk. Explicit a posteriori error estimates for eigenvalue analysis of heterogeneous elastic structures. *CMAME*, 196(37):3614–3623, 2007.