THESIS

THE REFINEMENT BY SUPERPOSITION APPROACH TO HP-ADAPTIVITY FOR FINITE ELEMENT SIMULATIONS IN COMPUTATIONAL ELECTROMAGNETICS

Submitted by

Jeremiah Corrado

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2022

Master's Committee:

Advisor: Branislav M. Notaros

Milan M. Ilic
Michael Kirby

ABSTRACT

THE REFINEMENT BY SUPERPOSITION APPROACH TO HP-ADAPTIVITY FOR FINITE ELEMENT SIMULATIONS IN COMPUTATIONAL ELECTROMAGNETICS

The Finite Element Method (FEM) is a versatile numerical tool for simulating the behavior of Partial Differential Equations (PDEs) over geometric models with arbitrary shapes and material parameters. Its applications are widespread as PDEs are used to model the behavior of almost all complex physical systems. Common PDEs include Schrödinger's equation which governs the evolution of quantum systems; the Navier-Stokes equations, which describe the behavior of fluids; and Maxwell's Equations, which are a macroscopic description of all Electric and Magnetic Phenomena. When leveraged against Maxwell's Equations, FEM allows engineers and scientists to rapidly design a wide range of Radio-Frequency (RF) devices such as antennas, RF filters, waveguides, and many others, all of which are important to the development of communications networks, sensing networks, and computing infrastructure.

One open question within FEM research is how to maximize simulation efficiency (i.e., what is the best strategy to maximize the ratio of simulation accuracy to computational resource usage). A typical approach is to use a multi-step process beginning with a coarse (or low resolution) discretization of the geometric model in question, which uses a small number of computational resources. This model is then iteratively refined in an intelligent manner, only introducing more entropy where it is most needed to improve solution accuracy. After several iterations, this approach will have achieved a desirable balance between resource usage and solution accuracy. This work focuses on the development and testing of a simple and ergonomic implementation of h- and p-refinements for FEM. When used in tandem, these two refinement strategies are amenable to the above procedure and efficiently produce accurate results on challenging Computational Electromagnetics problems.

ACKNOWLEDGEMENTS

I would like to thank the CSU Department of Electrical and Computer Engineering for being my educational home for the last six years through my undergraduate and graduate degrees. I am particularly grateful to everyone involved in the CSU Electromagnetics lab for being a pivotal part of my education and the completion of this work. My understanding of and contribution to this field would not have been possible without the guidance of my adviser, Branislav Notaros, and the close collaboration with my lab-mate, Jake Harmon. I would also like to thank my family and friends for their ceaseless support and encouragement over the last two years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Computation modeling tools are an essential part of any modern engineering effort. The systems that engineers are interested in designing have reached a level of complexity that far exceeds the reach of direct mathematical analysis. As such, many numerical tools have been developed to predict the behavior of complex physical systems allowing engineers to make design decisions without the need to construct physical prototypes or perform difficult physical measurements.

Without these tools, much of our modern digital infrastructure (CPUs, memory, high-speed communication) and physical infrastructure (bridges, dams, large buildings) would be impossible or prohibitively expensive to develop. The same is true for analyzing and designing mechanical, chemical, biological, and even quantum systems. As such, the advent of computational modeling tools has been an important area of research, and the improvement of these methods continues to be a vital part many engineering disciplines.

Of the known methodologies, the Finite Element Method (FEM) is one of the most popular due to its versatility and computational scalability. As its name suggests, the premise of the method is to solve a simulation problem by discretizing a model of a physical system into a finite set of elements; each of which supports a finite-accuracy basis set. The dynamics of interest–in the form of a partial differential equation (PDE)–are analyzed individually on each element. The problem is posed such that a given boundary condition is satisfied and an inter-element continuity condition is satisfied. The solution to this problem is expressed in terms of a weighted sum of all basis-functions in the mesh. Generally speaking, this process converts a spatially continuous PDE problem into a straightforward linear-algebra problem whose solution is a good approximation of the full PDE solution.

One can improve the accuracy of the numerical solution by increasing the resolution of the discretization through refinement. We consider two refinement modalities in this work: h-refinement, which introduces smaller elements into the mesh (improving its spatial resolution), and p-refinement,

which increases the polynomial expansion order over existing elements (improving resolution in solution space). The combination of these two techniques is known as hp-adaptivity. This work investigates a non-standard approach to implementing hp-adaptivity, called Refinement-by-Superposition (RBS), wherein h-refinements introduce new overlay elements rather than replacing existing elements. This subtle change massively reduces the complexity of continuity condition enforcement, allowing for the development of simpler FEM software tools.

RBS also reduces the difficulty of implementing anisotropic (or directional) h-refinement. Anisotropic hp-adaptivity can yield significant boosts in efficiency for a variety of FEM simulations when compared with a purely isotropic refinement paradigm. Anisotropy is shown to be particularly useful in the presence of challenging geometric features that can consume many computational resources.

FEM is easily parallelizable due to the element-localized nature of most of the computations involved; however, the compute resources needed to run FEM simulations in a massively parallelized fashion are generally expensive to use. As such, any alteration or extension to the method which can improve efficiency is immediately useful to a wide array of software applications. For many applications, it is also beneficial to be able to run smaller simulations on personal computing devices and avoid the use of external compute hardware entirely. This motivates the development of efficiency-improving FEM techniques, like the one discussed in this work.

The following chapters explore the RBS approach to anisotropic hp-adaptivity from three perspectives: analysis of the fundamental theory, application to adaptive refinement, and open-source software implementation. Chapter 2, taken from the author's work [1], builds on the more fundamental theory discussed in previous hp-adaptivity research [2, 3] by summarizing the RBS approach then introducing anisotropic hp-adaptivity. The efficiency benefits of the method are demonstrated on a challenging benchmark problem. Chapter 3 then introduces an adaptive refinement strategy that can efficiently leverage the implementation discussed in Chapter 2 without pre-defined knowledge of the given problem. Such a strategy must exist for RBS to be valuable in real-world applications. Here, we use an error estimation procedure to drive adaptive refinements

which is based on a more traditional Refinement by Replacement (RBR) exploration of the aforementioned benchmark problem [4]. Lastly, Chapter 4 presents an open-source implementation of the code used to generate the results in the previous two chapters. We discuss the library's design and structure, along with some examples of how it may be used as a simulation tool or to drive further FEM research. Overall, these chapters aim to provide a deep understanding of RBS based anisotropic hp-adaptivity both theoretically and practically.

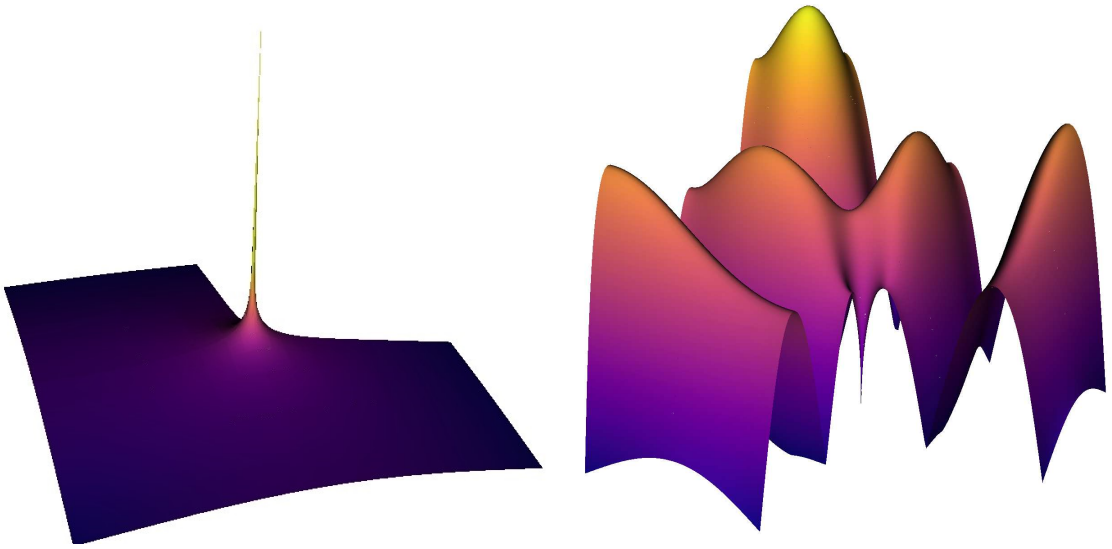
Chapter 2

Anisotropic hp-Refinement Using the Refinement by Superposition Method

2.1 Introduction

Fully anisotropic hp-refinement over quadrilateral and hexahedral discretizations, an under-explored paradigm in Computational Electromagnetics (CEM) and Finite Element Methods (FEM), facilitates a significant enhancement in the tuning of discretizations for accurate *and* efficient simulations. However, given the difficulty of implementing isotropic h-adaptivity over quadrilateral or hexahedral cells under the constraints of a Continuous Galerkin Formulation, little work has been done to extend popular methodologies to support anisotropic h-adaptivity. That is not to say that the necessary theoretical foundations are absent [5, 6], rather, the requisite implementation complexity has impeded adoption of these techniques. The implementation difficulties can be avoided entirely by employing triangular or tetrahedral discretizations [7] or a discontinuous Galerkin Formulation [8]; however, non-rectangular cells are sub optimal for the linear independence of unit vectors in many CEM applications, and discontinuous Galerkin formulations do not maximize per-DoF efficiency. Therefore, we explore a far less burdensome approach to anisotropic hp-adaptivity with quadrilateral cells under a Continous Galerkin formulation by leveraging a Refinement-by-Superposition (RBS) method.

Previous work has demonstrated the RBS method's ability to produce state-of-the-art exponential convergence on the 2-D Maxwell eigenvalue problem, even in the presence of singular or non-smooth behavior [3], of the form depicted in Fig.2.1. We note that without isolation of the irregular solution behavior through suitable h-refinements, p-refinement alone is insufficient to achieve exponential convergence. Thus, any computational method that aims to converge quickly



(a) 1st Eigenfunction **(b)** 9th Eigenfunction

Figure 2.1: Two solutions to the Maxwell Eigenvalue Problem on a unit size L-Shaped waveguide. Both solutions are shown as the magnitude of the Electric Field and exhibit sharp behavior around the re-entrant corner.

regardless of non-idealities in the solution behavior must also implement some form of local h-adaptivity in concert with p-refinement [9–11].

The distinguishing feature of the RBS method as an approach to h-refinement is its simple formulation and ease of implementation. This stands in stark contrast to more mainstream h-adaptivity methods, such as the constrained nodes approach, which necessitate sophisticated algorithms to treat hanging-nodes. As described in [12], the RBS approach enforces continuity requirements by construction and is also able to handle arbitrary degrees of mesh irregularity (edges or faces can have any number of hanging nodes) without any special treatment.

Additionally, RBS not only supports anisotropic h-refinement without significant theoretical alterations, but there are also very few practical difficulties associated with its implementation. This is not the case for other, more traditional approaches to h-refinements over quadrilateral or hexahedral cells. Although excellent results can be achieved for applications in CEM with isotropic RBS hp-adaptivity [3], we aim to show the further benefits of introducing anisotropy in both h and p.

The remainder of this paper is organized as follows. Section II gives a short overview of the fully isotropic formulation of the RBS method. This is only intended to set the stage for the introduction of anisotropic adaptivity. More detailed descriptions of RBS can be found in [3, 12, 13]. Section III explores some practical considerations and implementation details for introducing anisotropy to the RBS method. A specific framework is discussed for the 2-D case; however, it generalizes trivially to 3-D. Finally, Section IV presents an experimental analysis of the method using the 2-D Maxwell Eigenvalue problem as a benchmark.The previously mentioned mesh with singular and non-smooth eigenpairs is used to show the efficacy of the method in situations where hp-refinement is required to achieve exponential convergence. The results show that anisotropic hp-refinements can achieve the same accuracy as fully isotropic refinements with significantly fewer Degrees of Freedom (DoFs), illustrating the usefulness of the method to other challenging problems in CEM.

2.2 Overview of the Refinement-by-Superposition Method

The RBS method is an alternative approach to *h*-refinement that is well suited for a wide variety of computational methods. We focus here on discretizations that employ H(curl)- or H(div)-conforming hierarchical basis over quadrilateral or hexahedral cells (such as those in [14]). We also reference the shape-function classification and coordinate system described in [2].

In contrast to the more typical Refinement-by-Replacement (RBR) method where *h*-refinements decompose a cell into a set of smaller cells, RBS superimposes a set of “child cells" over a “parent cell" without removing the parent cell from the mesh. This subtle change obviates the need for special treatment of hanging nodes as local h-refinements introduce new nodes on a separate refinement layer and leave the parent edges completely intact.

Alternatively, the direct treatment of hanging nodes calls for more complex enforcement of continuity. A common RBR-based strategy is to introduce constrained nodes into the mesh. These nodes add DoFs to the system which are not actually free (they do not contribute entropy to the system) but are carefully constructed such that boundary conditions are satisfied. As such, local

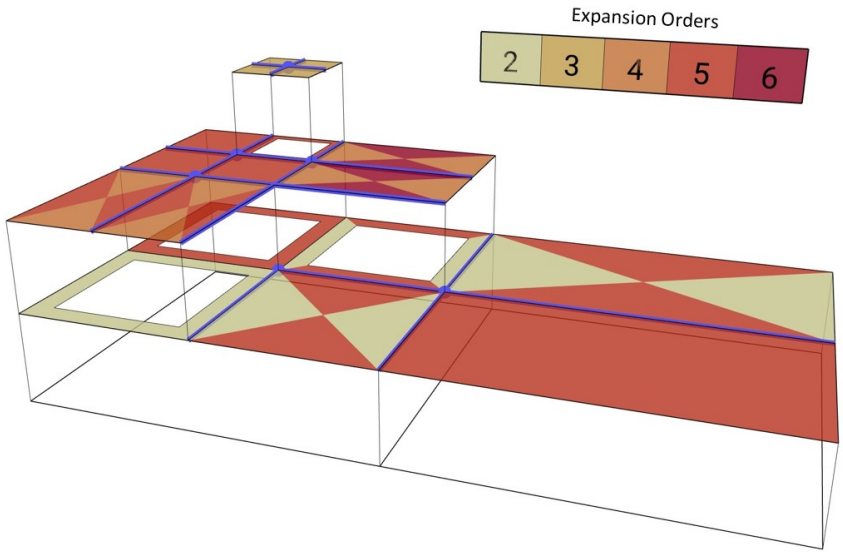


Figure 2.2: Example of a mixed-order 2-D Mesh constructed using the RBS framework. Nodes and Edges with active shape-functions are highlighted in Purple (those shared across refinement layers are highlighted on both layers). Cells that have active Edge-Type or Node-Type shape functions with inactive Cell-Type functions are colored around the edges and transparent in the center. Cells without active shape functions are completely transparent.

h-refinements are allowed, but per-DoF efficiency is reduced and implementation complexity is high.

The RBS method makes a different trade-off between implementation difficulty and resource requirements. By leaving the parent cell in the system, and allowing its DoFs to remain active as necessary, continuity is enforced by construction. The nature of this approach therefore reduces the implementation complexity at the expense of reduced matrix sparsity due to the increased interactions between refinement layers. These denser systems can require more memory and compute to solve; however, the lower implementation complexity often outweighs such a cost.

The following algorithm enforces continuity between neighboring cells and linear independence between refinement layers (additional details and illustrations related to this process in the context of isotropic refinements may be found in [3]):

1. Iterate over each cell and enumerate all possible shape functions based on the cell's expansion orders, then associate them with the relevant geometric components (cell, face, node, or edge)

- For $H(\text{curl})$, associate the shape functions based on the non-zero tangential components
- For $H(\text{div})$, associate the shape functions based on the non-zero normal components

2. Iterate over each non-cell geometric component (faces, nodes, and edges)
    - For each shape function associated with that component, search for matching shape functions on neighboring cell(s)
    - If a match is found, designate it as a new DoF in the connected system
3. Iterate over each non-cell geometric component a second time
    - If the component has descendants with active DoFs, deactivate its DoFs (ex: if an edge has two direct descendant edges which possess sets of matched shape functions, deactivate the edges DoFs)
4. Iterate over each cell
    - If the cell has descendants, leave its cell-type shape functions inactive; otherwise, add them to the connected system as new DoFs

The above procedure is summarized more succinctly by the following remarks:

- Whenever possible, only the DoFs associated with the most-*h*-refined geometric components within an ancestry tree should be active
- If a node, edge, or face does not have an equally *h*-refined neighbor with which to match shape functions, then the responsibility falls on the nearest ancestor component (with a valid neighbor) to enforce continuity

Fig. 2.2 shows an example of an RBS mesh with the active geometric components annotated. This serves as visual representation of the above algorithm. Some anisotropic h- and p-refinements are also included to demonstrate that they are fully supported by the RBS framework. (For this figure and others, the vertical spacing simply represents the addition of refinement layers and has no physical significance to the geometry of the mesh.)

2.3 Introduction of Anisotropic *h*-refinements to RBS: Practical Considerations and Implementation Details

2.3.1 Implementation Goals

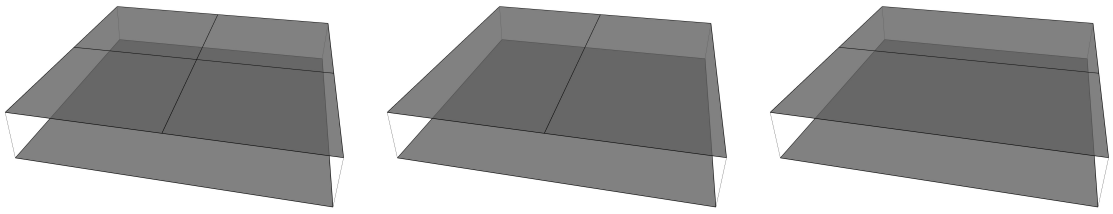
An anisotropic h-refinement, i.e., a directional h-refinement, is advantageous where the intensity of non-smooth or singular behavior is also directionally dependent or is confined to one small region of a cell. In such cases, isotropic h-refinement may introduce more DoFs than necessary to improve solution accuracy. Anisotropic h-adaptivity can therefore contribute to improved efficiency or even faster convergence rates by introducing new unknowns in a more frugal manner [8]. Likewise, anisotropic p-adaptivity permits increasing a cell's polynomial expansion order in only one direction to drive more efficient improvements to the solution accuracy.

The methodology described in section II imposes few limitations on the shape or size of the superimposed cells. Previous works (such as [3]) have limited h-refinements to a simple 4-cell isotropic superposition; however, any set of child cells is permissible so long as the following conditions are met:

1. The child cells cover the entirety of the parent cell
2. The child cells do not extend into neighboring cells or beyond the boundary of the mesh
3. No internal hanging nodes are introduced among the child cells S.T. internal continuity is enforced naturally

For example, a 9-Cell isotropic superposition is equally as valid as a 4-Cell superposition, as the parent cell uses the same mechanism to enforce continuity in either case.

One can imagine taking advantage of the wide variety of anisotropic *h*-refinements allowed by the RBS method; however, it is useful to impose a few practical limitations to generate a simple implementation with maximum usability. In other words, given so few limitations on the shape, size, number, and orientation of the child cells, it is difficult to construct a simple *h*-refinement scheme that is easily targeted by adaptive methods (such as those described in [4]) or by human



(a) T-Type **(b)** U-Type **(c)** V-Type

Figure 2.3: The three types of *h*-refinement used in our 2-D anisotropic h-refinement implementation; all illustrated as a refinement of a single square cell

users while capturing all possible *h*-refinements. A practical goal for any anisotropic h-refinement implementation is to strike a balance between feature-richness and simplicity.

2.3.2 Implementation Details for 2-D FEM

For a specific demonstration of the procedure, we study an anisotropic RBS implementation designed for 2-D FEM which aims to illustrate the benefits of anisotropy while minimizing implementation overhead. This framework generalizes to 3-D trivially. The three types of *h*-refinements considered here (named T-, U-, and V-Type) are shown in Fig. 2.3. T-Type refinement is identical to the isotropic *h*-refinement used in [3]. It is also equivalent to the successive application of a U-Type refinement and two V-Type refinements (or the inverse); however it is implemented directly for the sake of simplicity.

The two anisotropic *h*-refinements, U-Type and V-Type, superimpose only two new cells over the parent cell improving the u-directed or v-directed resolutions respectively, where the coordinates u and v correspond to those of the reference cell, while leaving the opposite direction unaffected. We also designate here that horizontal edges (those superimposed with a new node during a U-Type refinement) are called u-directed edges, and vertical edges (those superimposed with a new node during a V-Type refinement) are called v-directed edges.

Within this framework, it is useful for each cell to keep track of its "*h*-refinement level" in the u and v directions. In other words, each cell must know how many U-Type and V-Type refinements

were applied over the history of its construction from the base layer. When a cell is h-refined, the refinement levels of the child cells are a function of the parent's refinement levels and the refinement type:

- T-type:

$$(u_{child}, v_{child}) = (u_{parent} + 1, v_{parent} + 1)$$

- U-type:

$$(u_{child}, v_{child}) = (u_{parent} + 1, v_{parent})$$

- V-type:

$$(u_{child}, v_{child}) = (u_{parent}, v_{parent} + 1)$$

With this information linked to each cell, it becomes straightforward to determine whether two adjacent cells can match Edge-Type shape functions or if four adjacent cells can match node-type shape functions. This is explored first by example, using the mesh configurations in Fig. 2.4, then a more explicit formulation is given.

The mesh in Fig.2.4(a) has no edges shared between refinement layers, so shape function matching is relatively simple and resembles the isotropic case. Here, the children of Cell 0 and Cell 1 have h-refinement levels of (1,1) and (0,1) respectively. Cells 3 and 6 share a v-directed edge, and their v-directed h-refinement levels match, therefore they can match Edge-Type shape functions. The same is true for Cells 5 and 7. As explained in Section II, these edges are more h-refined than their parent edge, and they can both support shape-functions, therefore Edge-Type functions on Cells 0 and 1 are unnecessary to enforce continuity and are left inactive.

An important feature of the U- and V-Type h-refinements is that only 4 new edges are constructed along the border of the parent cell (as opposed to the 8 generated by a T-Type refinement). The other two edges are shared with the parent cell, which can introduce some ambiguity with shape function matching. In Fig.2.4(b), the central edge is shared by Cells 0, 1, and 3. As such, the Edge-Type shape functions on Cell 1 can match with those on Cells 0 or 3 which have h-refinement

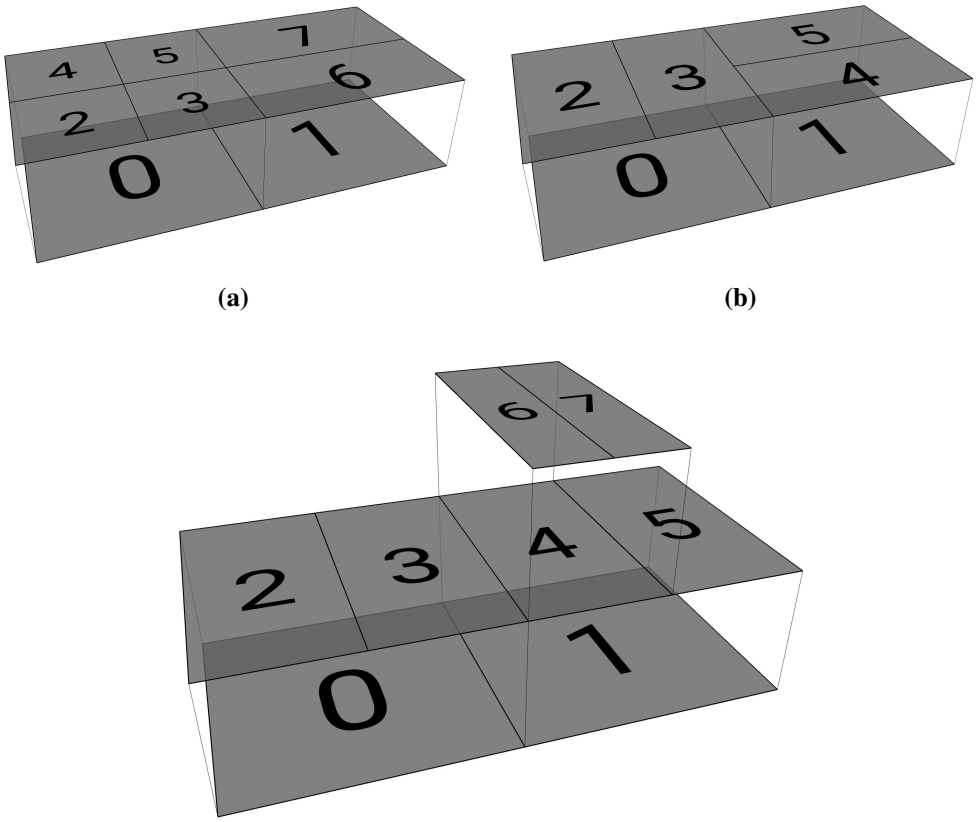


Figure 2.4: Examples of Edge-Type shape function matching cases introduced with anisotropic h-refinement. These cases all focus on continuity enforcement over the v-directed edge shared by Cells 0 and 1. (a) A mesh with a single isotropic h-refinement and one V-Type refinement. (b) A Mesh with one U-Type and one V-Type h-refinement. Outside the context of RBS, this would introduce a hanging-node along the central edge. (c) A 3-Layer Mesh with only U-Type h-refinements illustrating the necessity for a cell-ranking system.

levels of (0, 0) and (1, 0) respectively. In keeping with the RBS concept that the most h-refined cells should be responsible for maintaining continuity whenever possible, Cells 3 and 1 will match Edge-Type shape functions and those on Cell 0 will remain inactive.

This logic extends further when multiple cells of the same relevant h-refinement level share a single edge. For example in Fig.2.4(c), Cells 0 and 3 share many possible matching combinations with Cells 1, 4, and 6, which all have v-directed h-refinement levels of zero. To select a single matching pair, the choices on both sides of the edge are ranked by their u-directed refinement level, and the highest-ranked cells on either side are chosen. As such, shape-functions on Cells 3 and 6 are used to enforce continuity over the central edge.

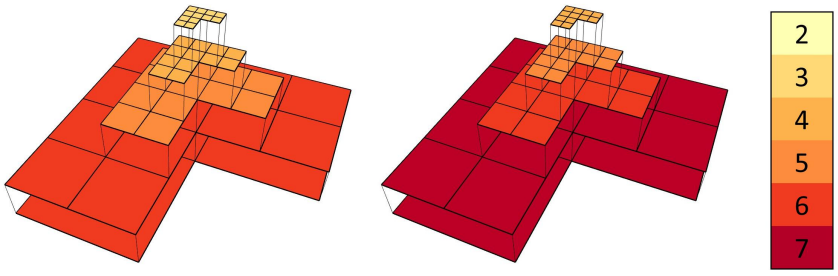
The logic in the above examples is captured by the following remarks:

- All edges in the mesh must maintain two lists of adjacent cells—one for each side—which are ranked first by the cells refinement-level associated with the edges own direction, then by its refinement level associated with the opposite direction

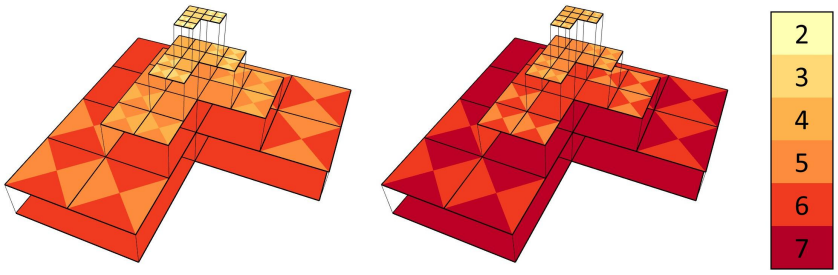
- The highest-ranked cell on each side of the edge is used to construct a valid pair if and only if the edge itself is needed to enforce continuity (which is determined by the broader RBS procedure given in Section 2.2)

An extension of this logic to include four cells is used to match node-type shape functions. In such cases, cells are ranked by the four relevant edges in the same manner, and each edge must "agree" with the neighboring two edges on which cells to select.

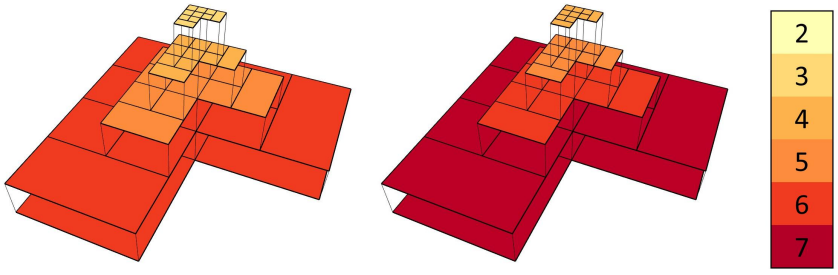
No major changes to the isotropic RBS method, described in Section 2.2, are required. The algorithm used to match shape functions on neighboring cells and enforce continuity requirements remains essentially the same, except for the implementation of Step 2, which is amended to include the slightly more complex matching procedure described above.



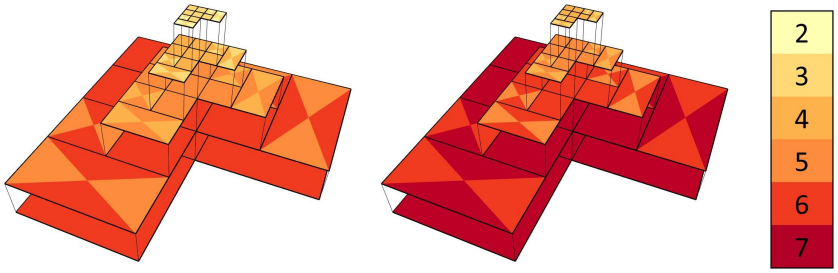
(a) Isotropic-*hp*



(b) Isotropic-*h* Anisotropic-*p*



(c) Anisotropic-*h* Isotropic-*p*



(d) Anisotropic-*hp*

Figure 2.5: Illustration of the *a priori* refinement strategies on a fiver-layer mesh (k = 4). Figures in the left column show the initial mesh and Figures in the right column show the same mesh after one refinement iteration. In all cases, the refinement (from left to right) increments u- and v-directed expansion orders by 1 on all cells. The color scales on the right designate expansion orders. (*p*-anisotropy is shown on the dual-colored cells S.T. the expansion order associated with a given edge is represented by the color touching that edge.)

2.4 Numerical Results

2.4.1 Benchmark Problem Description

We now establish the advantages of introducing hp-anisotropy to the RBS method by solving the Maxwell Eigenvalue problem on an L-shaped waveguide terminated by Dirichlet boundary conditions proposed by [15]. Additionally, we only consider TE propagation modes by asserting that the solution is purely transversal. The Maxwell Eigenvalue problem is formulated as follows: Find $U = \{\mathbf{u}_{hp},\, \lambda_{hp}\} \in B_{hp} \times \mathbb{R}$ such that

$$a(\mathbf{u}_{hp},\, \boldsymbol{\phi}_{hp}) = \lambda_{hp} m(\mathbf{u}_{hp},\, \boldsymbol{\phi}_{hp}) \quad \forall \boldsymbol{\phi}_{hp} \in B_{hp} \tag{2.1}$$

for $B_{hp} \subset H(\text{curl}; \Omega)$, where $m(\mathbf{u}_{hp},\, \boldsymbol{\phi}_{hp}) = \langle \mathbf{u}_{hp},\, \boldsymbol{\phi}_{hp} \rangle$, $a(\mathbf{u}_{hp},\, \boldsymbol{\phi}_{hp}) = \langle \nabla_t \times \mathbf{u}_{hp},\, \nabla_t \times \boldsymbol{\phi}_{hp} \rangle$

The convergence behavior is evaluated for the four combinations of (An) Isotropic-h and (An) Isotropic-p refinement. The solutions express singular and non-smooth behavior making it an ideal benchmark for evaluating the effectiveness of hp-methods. The two eigenfunctions in question are the first—shown in Fig. 2.1(a)—which contains a singularity along the re-entrant corner, and the ninth—shown in Fig. 2.1(b)—which has non-smooth behavior on the re-entrant corner and more complex behavior elsewhere.

2.4.2 Refinement Strategies

A set of four *a priori* refinement strategies are used to illustrate the difference in convergence behavior with and without anisotropy. An example of each is shown in Fig. 2.5. Although the first and ninth eigenfunctions are quite different, they share a key feature of sharp behavior around the re-entrant corner, and thus the same strategies are sufficient to produce exponential convergence on both eigenpairs.

The first strategy, shown in Fig. 2.5(a), is fully isotropic and is identical to the strategy used in [3] to illustrate the method's capacity for exponential convergence. Here, a given number of layers: k, describes the number of refinement layers that were added to the base discretization.

Each new layer is generated with a T-Type refinement of the three cells surrounding the re-entrant corner. To construct the starting discretization, the cells on the first layer are assigned an expansion order of k + 2 (ensuring a minimum order of 3 on the top layer). Subsequent layers have an expansion order 1 less than the previous layer.

Anisotropic p-refinements are introduced by reducing the expansion order of a few select cells by 1 in only one direction. The specific pattern of anisotropic expansion orders is shown in Fig. 2.5(b). The targeted cells and expansion directions were chosen based on the areas of the mesh where the u-directed electric field is much more intense than the v-directed electric field or vice versa.

Anisotropic h-refinements are introduced by replacing the outer two T-Type refinements with a U/V-Type refinement followed by a V/U-Type refinement of the resultant cell closest to the re-entrant corner. As shown in Fig. 2.5(c), the mesh remains courser over the regions that are farther from the re-entrant corner. As such, new DoFs and smaller-scale cells are only introduced around the re-entrant corner where they are most needed to capture the sharp solution behavior.

Finally, a strategy with anisotropic hp-refinements is shown in Fig. 2.5(d). This is simply a combination of the previous two strategies and constitutes the DoF savings from both.

2.4.3 Results and Discussion

Convergence behavior is evaluated as follows: for each of the strategies shown in Fig. 2.5, an initial mesh is generated based on a given value of k. Then, accuracy data is collected by repeatedly p-refining each cell in the mesh and solving the Maxwell eigenvalue problem described above. The relative error at each refinement iteration is computed as the absolute difference between the solution eigenvalue and an accurate numerical benchmark [4]. Results for the first eigenvalue are shown in Fig. 2.6 and results for the ninth are shown in Fig. 2.7. Both figures include results for k=4 and k=8. A maximum expansion order of 14 is employed across experiments, meaning that the k=4 plots contain more refinement iterations (as those experiments start with a lower maximum expansion order).

In all cases, the relative error is nearly identical across the four refinement strategies for any given iteration; however, the number of DoFs varies significantly. Universally, the fully anisotropic refinement pattern performs the most efficiently (requiring fewest NDoFs to achieve a given accuracy), then the Anisotropic-h Isotropic-p pattern, then the Isotropic-h Anisotropic-p pattern, and finally the fully isotropic pattern is the least efficient. Additionally, it is clear that the anisotropic h-refinements constitute a larger improvement in efficiency than the anisotropic p-refinements.

The efficiency improvements obtained by each of the three anisotropic strategies are given numerically in Tables 1-4, with the most significant improvements from each iteration highlighted in bold. Results are given for k=4 and k=8 on both eigenpairs. The values shown are the ratio of the anisotropic strategies per-DoF efficiency to the isotropic per-DoF efficiency for the same iteration. In other words, these values express how many times more efficient the anisotropic solution was than the isotropic solution for some refinement iteration. Or more explicitly, the efficiency gains are expressed by the following equation, where η denotes per-DoF efficiency, ε denotes the accuracy (or inverse of the relative error), and N denotes the NDoFs:

$$\frac{\eta}{\eta_{iso}} = \left(\frac{\varepsilon}{\varepsilon_{iso}}\right) * \left(\frac{N_{iso}}{N}\right) \tag{2.2}$$

For most iterations, the achieved accuracy is approximately equal across strategies, meaning that the above equation is well approximated by the ratio of the number of DoFs. Thus, for the experiments shown here, efficiency gains are equivalent to the DoF savings introduced by the anisotropic strategy.

Across all experiments, anisotropic h-refinements present the most significant gains in efficiency, while anisotropic p-refinements present a smaller, but still useful, boost in efficiency. Notably, the efficieny gains associated with the first refinement iterations for the 9th eigenvalue exhibit some erratic behavior (which can also be deduced from Fig. 2.7). In all other cases, the fully anisotropic refinement strategy yielded the largest gains in efficiency, closely followed by the anisotropic-h isotropic-p strategy, then followed by the isotropic-h anisotropic-p strategy. These insights align with the data shown in Figures 2.6 and 2.7, and indicate that the combined appli-

Table 2.1: Efficiency Gains of anisotropic strategies at each refinement iteration for ***k*=4** on the **1st eigenvalue**

Refinement Iteration	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Isotropic-h Anisotropic-p	1.1463	1.1251	1.1062	1.0906	1.0793	1.0704	1.0633	1.0575	1.0526
Anisotropic-h Isotropic-p	1.2931	1.2866	1.2825	1.2794	1.2769	1.2750	1.2733	1.2720	1.2709
Anisotropic-hp	**1.4462**	**1.4195**	**1.3951**	**1.3757**	**1.3611**	**1.3497**	**1.3405**	**1.3330**	**1.3267**

Table 2.2: Efficiency Gains of anisotropic strategies at each refinement iteration for ***k*=8** on the **1st eigenvalue**

Refinement Iteration	**1**	**2**	**3**	**4**	**5**
Isotropic-h Anisotropic-p	1.0881	1.0855	1.0774	1.0697	1.0633
Anisotropic-h Isotropic-p	1.2760	1.2901	1.2903	1.2886	1.2872
Anisotropic-hp	**1.3673**	**1.3824**	**1.3743**	**1.3643**	**1.3559**

cation of anisotropic h- and p-refinements is a highly effective strategy to increase the per-DoF efficiency of an FEM simulation.

2.5 Conclusion

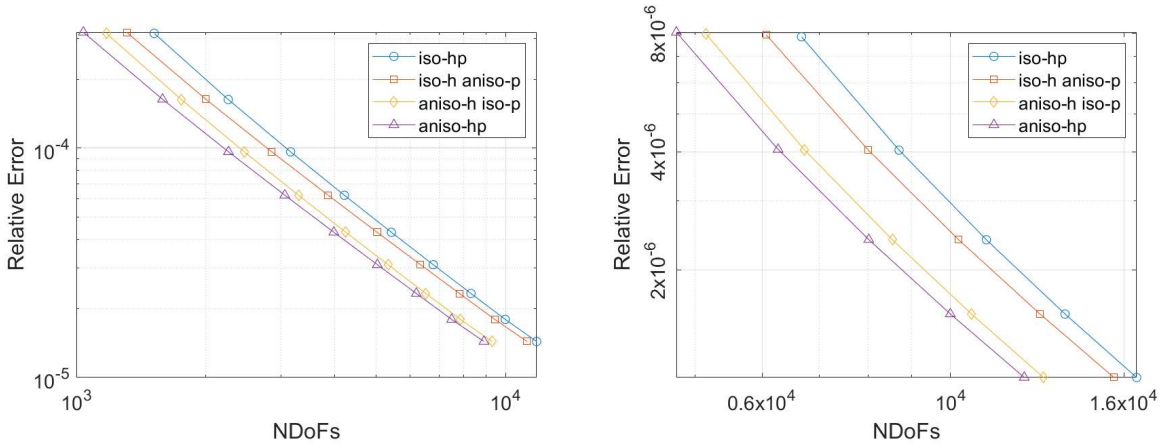
We have demonstrated an extension of the capabilities of the Refinement-by-Superposition approach to full hp-adaptivity by including anisotropic h- and p-refinements. Using the 2-D Maxwell eigenvalue problem over an L-shaped waveguide as a benchmark; we showed that anisotropic refinements, particularly in h, present a significant advantage in computational efficiency. This is consistent with a theoretical understanding, as anisotropic h-refinements permit the construction

Table 2.3: Efficiency Gains of anisotropic strategies at each refinement iteration for ***k*=4** on the **9th eigenvalue**

Refinement Iteration	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Isotropic-h Anisotropic-p	1.3281	1.1547	1.1061	1.0897	1.0788	1.0701	1.0631	1.0573	1.0525
Anisotropic-h Isotropic-p	1.2006	1.2612	1.2840	1.2781	1.2768	1.2748	1.2733	1.2719	1.2708
Anisotropic-hp	**2.1333**	**1.4272**	**1.3971**	**1.3732**	**1.3604**	**1.3492**	**1.3402**	**1.3327**	**1.3265**

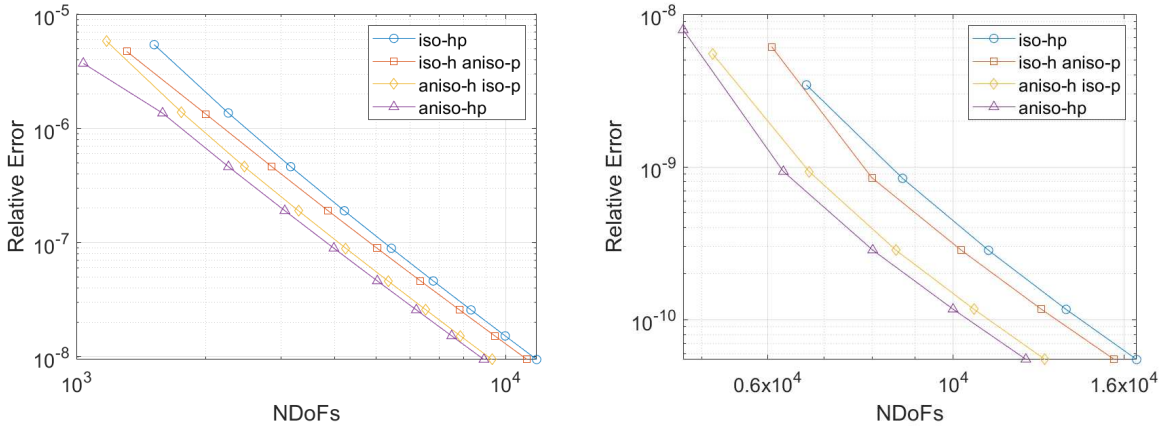
Table 2.4: Efficiency Gains of anisotropic strategies at each refinement iteration for k=8 on the **9th eigenvalue**

Refinement Iteration	**1**	**2**	**3**	**4**	**5**
Isotropic-h Anisotropic-p	0.6199	1.0788	1.0752	1.0683	1.0627
Anisotropic-h Isotropic-p	**0.8081**	1.1713	1.2875	1.2862	1.2854
Anisotropic-hp	0.6122	**1.2517**	**1.3698**	**1.3600**	**1.3526**



(a) k = 4 **(b)** k = 8

Figure 2.6: Convergence behavior of four refinement strategies shown on a log-log scale for the 1st eigenvalue. Results are given for k=4 and k=8.



(a) k = 4 **(b)** k = 8

Figure 2.7: Convergence behavior of four refinement strategies shown on a log-log scale for the 9th eigenvalue. Results are given for k=4 and k=8.

of mesh configurations that capture small-scale behavior without introducing redundant DoFs into the system. Due to the challenging nature of the benchmark problem, we are confident that these efficiency improvements will be broadly applicable to other difficult problems in CEM.

This work also serves to further illustrate the benefits of the RBS method's low implementation complexity. Other approaches to h-refinement on quadrilateral or hexahedral discretizations involve complex frameworks for boundary condition enforcement which do not easily lend themselves to the implementation of anisotropy. The RBS method, by contrast, imposes few limitations on the configuration of the superimposed cells, and thus supports h-anisotropy essentially for free. A conservative implementation of h-anisotropy for the 2-D case was described in detail and can readily be expanded to include more refinement types and can also be generalized for 3D FEM.

Chapter 3

Adaptive Refinement for Anisotropic hp-Refinement

3.1 Introduction

The *a priori* strategy discussed in Chapter 2 is not generally useful for practical applications in CEM. That is, the strategy shown, with its various combinations of isotropic and anisotropic refinement, was specifically designed for the re-entrant corner mesh and would not generalize to other scenarios. Thus, we seek an adaptive strategy capable of effectively leveraging anisotropic hp-adaptivity in a wide variety of situations to take full advantage of the RBS method. Here we describe an iterative refinement strategy, which progressively transforms a low-quality discretization into a highly efficient discretization. The notion of efficiency here is the same as in Chapter 2: achieving maximum solution accuracy with minimum resource utilization. Efficiency can be conceptualized as seeking the maximum accuracy for a limited set of computational resources or the minimum resource utilization for a desired accuracy.

This chapter discusses one such adaptive refinement strategy and evaluates its effectiveness on the benchmark problem defined in Section 2.4.3. We use the same benchmark here, as its eigenfunctions require hp-adaptivity for efficient solutions, and to show that the adaptive algorithm arrives at a similar refinement strategy without any prior knowledge of the problem.

The algorithm described here is comprised of two overarching steps: error estimation and refinement selection. Specifically, we use error estimation to select a subset of elements for refinement. Our strategy is based on the dual weighted residual (DWR) error estimate procedure. Then, a group of heuristics is applied to the mesh to select which refinement modality would be most profitable. Convergence rate analysis and edge-error discontinuity analysis allow us to select between h-, p- and hp-refinement. Then, the analysis of edge error terms allows us to select a refinement direction; namely, a selection is made between the isotropic and various anisotropic variants of our hp-refinements.

3.2 Dual Weighted Residual Error Estimation

The Maxwell Eigenvalue Problem described in Section 2.4.1 can be expressed in the following dual weighted residual (DWR) form [16]:

$$e_{\lambda_{hp}}(1 - \tfrac{1}{2}M) = a(\mathbf{u}_{hp}, \mathbf{u} - \varphi_{hp}) - \lambda_{hp} m(\mathbf{u}_{hp}, \mathbf{u} - \varphi_{hp}) \quad \forall \varphi_{hp} \in B_{hp} \tag{3.1}$$

This equation expresses the error in the eigenvalue accuracy λ_{hp} in terms of an approximate solution to the primal problem $\mathbf{u}_{hp}$ and an exact solution to the dual problem $\mathbf{u}$. Because this exact solution is not available, a higher order solution is used in its place as an approximation. This gives an accurate estimate of the error. We compute the higher order solution by applying a global isotropic p-refinement of magnitude 1 to the entire mesh and computing a solution.

The normalization term $M = m(\mathbf{u} - \mathbf{u}_{hp}, \mathbf{u} - \mathbf{u}_{hp})$ ensures that the error estimate is equal to the absolute difference between the primal eigenvalue and the dual eigenvalue $e_{\lambda_{hp}} = |\lambda_{hp} - \lambda|$. The validity of this equality relies on a pre-normalization of the primal and dual solutions in terms of the L2 norm:

$$\langle \mathbf{u}, \mathbf{u} \rangle = 1 \quad \text{and} \quad \langle \mathbf{u}_{hp}, \mathbf{u}_{hp} \rangle = 1 \tag{3.2}$$

To extract the maximum amount of useful information from the error contribution estimates, φ is chosen to be $\mathbf{\Pi}_{hp}^{curl}\mathbf{u}$, which is the curl conforming projection of the dual solution onto the primal solution space [16]. The subtraction of this information from the dual solution leaves only higher order information in the error estimate. As such, terms that would cancel in the global assembly of the error contributions (due to Galerkin orthogonality) are excluded preemptively.

More granular data can be extracted from the DWR procedure by separating the stiffness-matrix integral, shown below:

$$a(\mathbf{u}, \boldsymbol{\phi}) = \langle \nabla_t \times \mathbf{u}, \nabla_t \times \boldsymbol{\phi} \rangle = \int_{\Omega_K} \nabla_t \times \mathbf{u} \cdot \nabla_t \times \boldsymbol{\phi} \quad d\Omega_K \tag{3.3}$$

into its by-parts form. In this expanded form, the surface term is computed separately from the edge terms, yielding more spatially distinct information [16]:

$$a(\mathbf{u}, \boldsymbol{\phi}) = \int_{\Omega_K} (\nabla \times \nabla_t \times \mathbf{u}) \cdot \boldsymbol{\phi} \quad d\Omega_K - \int_{\partial\Omega_K} (\nabla_t \times \mathbf{u}) \times \boldsymbol{\phi} \cdot \hat{n} \quad dS_K \tag{3.4}$$

The edge-terms will be stored and used later for discontinuity analysis in Section 3.3.2 and refinement direction selection in Section 3.3.3. The full-valued solution of $a(\mathbf{u}, \boldsymbol{\phi})$ is used For the purpose of evaluating Equations 3.1 and 3.5.

Error contributions are accumulated in terms of dual DoFs. The following equation shows the interactions of the ith dual DoF with all primal DoFs; however, in practice, only the overlapping integrals must be computed. Additionally, we note that the higher-order portions of the dual solution constitute the only non-zero error terms, as the others are filtered out by the curl conforming projection operator.

$$\mathbf{e}_{\mathbf{u}i} = \sum_{\boldsymbol{\nu}_{hp} \in \mathbf{u}_{hp}} a(\boldsymbol{\nu}_{hp}, (\mathbf{u} - \boldsymbol{\Pi}_{hp}^{curl}\mathbf{u})_i) - \lambda_{hp} m(\boldsymbol{\nu}_{hp}, (\mathbf{u} - \boldsymbol{\Pi}_{hp}^{curl}\mathbf{u})_i) \tag{3.5}$$

The per-DoF error contributions are then accumulated into separate lists of element DoFs and edge DoFs [1]. Specifically, for each element in the mesh, the error terms associated with its local Element-Type DoFs are summed up to generate an error coefficient for the solution on the element's surface. Then, for each edge in the mesh, the error terms associated with its Edge-Type DoFs are summed up to produce an error coefficient for the solution which spans the two adjacent elements (these coefficients are not to be confused with the by-parts edge-error terms from Equation 3.4).

A group of elements are then selected for refinement by choosing the top contributors from both lists. Specifically, any element with an error coefficient greater than the mean element coefficient is

[1] This separation is made to facilitate more effective h-refinement decisions. In experimental analyses, the presence of singular or sharp behavior often induces relatively large error terms on nearby Edge-Type DoFs, but does not necessarily do the same for Element-Type DoFs. As such, generating an error coefficient for all DoFs (Element-Type and Edge-Type) over an element can cause large edge-error terms to be ignored, leading to a general under h-refinement of the mesh.

marked for refinement. Any edge with a coefficient greater than the mean edge coefficient marks both of its active elements for refinement. Now, with these lists of elements, we must choose between h- and p-refinement and make direction selections. The following section discusses the heuristics used to make these decisions.

3.3 Refinement Classification Heuristics

3.3.1 Convergence Trend Analysis

The primary hp-decision heuristic is based on the theoretical difference in convergence rates associated with h- and p-refinements. This strategy is underpinned by the inequality shown in the following equation [11]:

$$|e_i| \leq C h_i^{min(k-1,p_i)} p_i^{-(k-1)} ||\mathbf{u}||_{H^k(\Omega)} \tag{3.6}$$

where e_i is the error associated with a given element i, C is a solution specific constant, k represents the local regularity of the solution, h_i is the diameter of the element, and p_i is the polynomial expansion order on the element. If the regularity is large enough, the error magnitude is dominated by an exponential relationship with the expansion order (as shown in equation 3.7). When the regularity is too small–or the solution is non-smooth–the error magnitude is dominated by an algebraic relationship with the expansion order (equation 3.8).

$$|e_i| \leq C \frac{h_i^{p_i}}{p_i^{k-1}} ||\mathbf{u}||_{H^k(\Omega)} \tag{3.7}$$

$$|e_i| \leq C \frac{h_i^{k-1}}{p_i^{k-1}} ||\mathbf{u}||_{H^k(\Omega)} \tag{3.8}$$

Thus, in smooth regions of a solution, one can expect the local error to shrink exponentially with respect to p_i, indicating that p-refinement would be appropriate to efficiently move towards a more accurate solution. Alternatively, if the solution is non-smooth, the accuracy will converge

algebraically, indicating that h-refinement must be employed to sequester the non-smooth behavior before p-refinements can be employed profitably.

This fact offers a simple heuristic to identify whether h- or p-refinement would better address a large error contribution associated with an element. As such, over the course of refinement, each element maintains a record of it's error contributions (and the local expansion order associated with that contribution), as does each edge. If at least three historical records are available for an element or edge, the following procedure is executed:

1. Separately fit the (p_i, $|e_i|$) samples to an exponential function and an algebraic function
2. Compute the sum of the squared residuals for both functional fits ($\hat{e}_{(exp)}$ and $\hat{e}_{(alg)}$):
   - $$\hat{\beta_{exp}} = \sum_i (|e_i| - \hat{e}_{(exp)}(p_i))^2 \tag{3.9}$$
   - $$\hat{\beta_{alg}} = \sum_i (|e_i| - \hat{e}_{(alg)}(p_i))^2 \tag{3.10}$$
3. If the exponential residual sum β_{exp} is smaller than the algebraic residual sum β_{alg} (meaning that the convergence behavior is better described by an exponential relationship), then the element is marked for p-refinement, otherwise it is marked for h-refinement

It is important to note that this process is executed for both lists of elements marked for refinement. Thus, the refinements indicated by this procedure are combined whenever an element and one or more of its edges has been marked for refinement. This may mean that an element is h-refined due to the slow convergence of the error terms associated with its Element-Type functions, but is also p-refined due to the fast convergence of its Edge-Type basis functions. This also means that higher magnitude p-refinements can be executed, however, we limit the magnitude to 2 in order to prevent over-refinement.

In order to ensure that the functional fits are relevant to the current state of the mesh, each stack of historical convergence data is limited to 5 entries. This way, if the refinement process begins

in the pre-asymptotic region, the initially erroneous convergence information will be forgotten during later iterations. Additionally, after an h-refinement, the most recent data point is inherited by the child elements, giving them some indication of the local convergence behavior without over-influencing future refinement decisions.

3.3.2 Edge Error Discontinuity Analysis

During the first few iterations of an adaptive refinement procedure, elements will not be able to complete the decision procedure defined in section 3.3.1, as they will not have a sufficient number of error estimates to form functional fits. Specifically, at least three data points are needed to differentiate the algebraic functional fits from exponential. This problem also arises when new elements are added to the mesh via h-refinement, and have not yet accumulated sufficient local convergence data. As such, we need a secondary decision procedure to make hp-refinement decisions when Convergence Trend Analysis cannot be completed.

One alternate heuristic is the analysis of discontinuities in the solution error among neighboring elements. Here, we analyze the relevant edges by computing the magnitude in the difference between the edge error terms on both adjacent elements (these are the edge-error terms generated by the integration-by-parts terms in the DWR procedure). The relevant quantity is described by the following equation where K^- and K^+ represent the relevant edges on the adjacent elements:

$$\delta_{edge} = \left|\left| \int_{\partial\Omega_{K^-}} (\nabla_t \times \mathbf{u}_{hp}) \times (\mathbf{u} - \varphi_{hp}) \cdot \hat{n} \quad dS_{K^-} \right| - \left| \int_{\partial\Omega_{K^+}} (\nabla_t \times \mathbf{u}_{hp}) \times (\mathbf{u} - \varphi_{hp}) \cdot \hat{n} \quad dS_{K^+} \right| \right| \tag{3.11}$$

When the magnitude in the difference between the neighboring error terms is small, this indicates that the discretization is well balanced. In other words, because both neighboring elements "agree" about the accuracy of the solution in the region of interest, we can assume that there is no small-scale or discontinuous behavior that the current discretization fails to capture. As such, p-refinement will be employed on the elements associated with the edge in question. Alternatively,

when neighboring error terms "disagree" about the solution accuracy on a given edge, this indicates that there is some discontinuity or small-scale behavior near the edge. As such, an h-refinement is likely to be more profitable for improving solution accuracy.

Like the previous heuristic, the element-wise and edge-wise analyses may indicate multiple refinements on the same element. These refinements add constructively. For example, a U-type and V-type h-refinement indicated on the same element will add to produce a T-type h-refinement. Furthermore, p-refinements will add constructively, but will be limited to a magnitude of 2.

3.3.3 Directional Selection Using Edge Error Terms

After a group of elements has been marked for refinement and their refinement modalities have been selected, a separate procedure is used to decide on a direction. Specifically, for h-refinements, we would like to decide between the three refinement options: T-, U-, and V-Type, as well as some more advanced composites of these types. For p-refinements, we would like to decide between an isotropic refinement of degree one (increment the expansion order by 1 in $both$ directions) and the degree-one anisotropic p-refinements in $either$ direction (increment the expansion order by 1 in only one direction).

This procedure relies on the edge-error terms computed during the DWR procedure. The vector of edge error coefficients are ordered in terms of the edge indices shown in figure 3.1. These coefficients are then multiplied by a directionality matrix. The matrices for h- and p-refinement are given in equations 3.12 and 3.13 respectively. The rows of these matrices are all magnitude-4 kernels which aim to establish a notion of direcitonality in the edge-error vector. In both cases, the refinement direction associated with the largest coefficient in R is applied to the element.

There are multiple kernels associated with the isotropic refinement variant within both matrices. The first, and most obvious, has a large magnitude when the edge-error coefficients are nearly equal across all edges. The other isotropic kernels produce large coefficients when two adjacent edges have large error magnitudes. In these cases, isotropic refinement is still preferable as there is no clear separation in direction. The next two kernels identify cases where two opposing edges

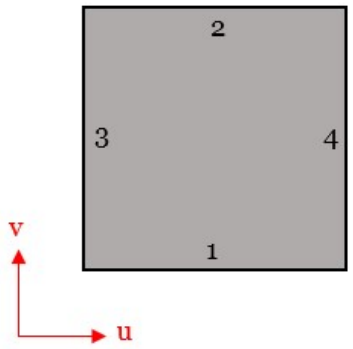


Figure 3.1: Edge index layout. Edge error terms correspond to this index ordering

have significantly larger error coefficients, indicating that anisotropic refinements are best suited to improve accuracy without introducing unneeded information.

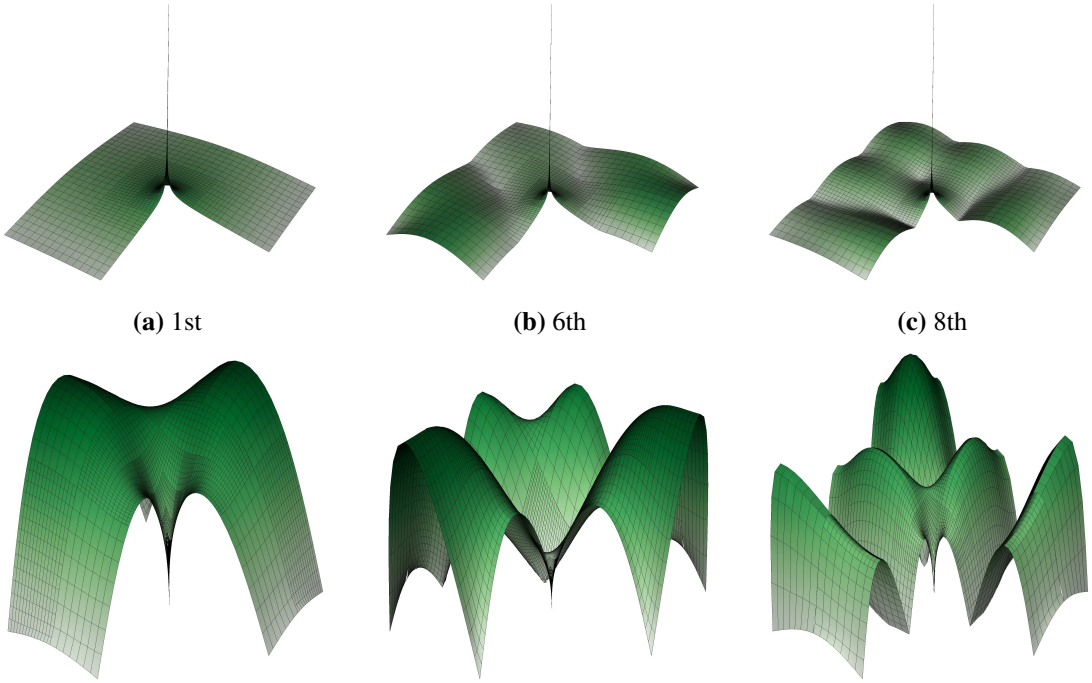
$$R = \begin{bmatrix} W_T \\ W_T \\ W_T \\ W_T \\ W_T \\ W_U \\ W_V \\ W_{V(bottom)} \\ W_{V(top)} \\ W_{U(left)} \\ W_{U(right)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 0 & 0 & 2 \\ 2 & 0 & 2 & 0 \\ 0 & 2 & 0 & 2 \\ 0 & 2 & 2 & 0 \\ 2 & 2 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 2.5 & 0 & 0.75 & 0.75 \\ 0 & 2.5 & 0.75 & 0.75 \\ 0.75 & 0.75 & 2.5 & 0 \\ 0.75 & 0.75 & 0 & 2.5 \end{bmatrix} \times \begin{bmatrix} ee_1 \\ ee_2 \\ ee_3 \\ ee_4 \end{bmatrix} \tag{3.12}$$

$$R = \begin{bmatrix} W_{(1,1)} \\ W_{(1,1)} \\ W_{(1,1)} \\ W_{(1,1)} \\ W_{(1,1)} \\ W_{(1,0)} \\ W_{(0,1)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 0 & 0 & 2 \\ 2 & 0 & 2 & 0 \\ 0 & 2 & 0 & 2 \\ 0 & 2 & 2 & 0 \\ 2 & 2 & 0 & 0 \\ 0 & 0 & 2 & 2 \end{bmatrix} \times \begin{bmatrix} ee_1 \\ ee_2 \\ ee_3 \\ ee_4 \end{bmatrix} \tag{3.13}$$

The h-refinement matrix has four additional refinement kernels associated with composite h-refinements. These aim to efficiently address scenarios where most of the error appears to be concentrated around one edge. An effective strategy is to successively apply anisotropic refinements to the element and of one of its child elements. For example, the refinement type **V(bottom)** is invoked when the bottom edge has an overpowering error coefficient. This will apply a V-Type refinement to the element, then a U-Type refinement to the resultant child element along the bottom edge. This way, the scale of the bottom edge has been reduced, but the scale of the top edge is left unchanged, leaving its continuity with neighboring elements intact.

3.4 Numerical Results

To asses the effectiveness of the refinement algorithm described in the previous sections, we apply it to the re-entrant corner waveguide benchmark described in Chapter 2. Convergence behavior is evaluated for six eigenfunctions; three of which are singular about the re-entrant corner and posses varying degrees of complexity elsewhere in the mesh, the other three are sharp about the re-entrant corner and present much higher energy variation elsewhere in the mesh. The electric field magnitudes for these eigenfunctions are shown in Figure 3.2. We also asses the value of anisotropic hp-adaptivity by executing the same experiment with and without anisotropic refinements enabled. For the isotropic version of the algorithm, the same exact DWR and hp-decision procedures are used, but the direction selection algorithm always defaults to the isotropic option.



(a) 1st **(b)** 6th **(c)** 8th

(d) 2nd **(e)** 5th **(f)** 9th

Figure 3.2: Eigenfunctions of interest for the re-entrant corner waveguide benchmark problem. The Electric Field Magnitudes are shown for the singular (1st, 6th, 8th) and Sharp (2nd, 5th, 9th) functions. All elements are plotted with an 8x8 grid of points which is shown superimposed on the fields

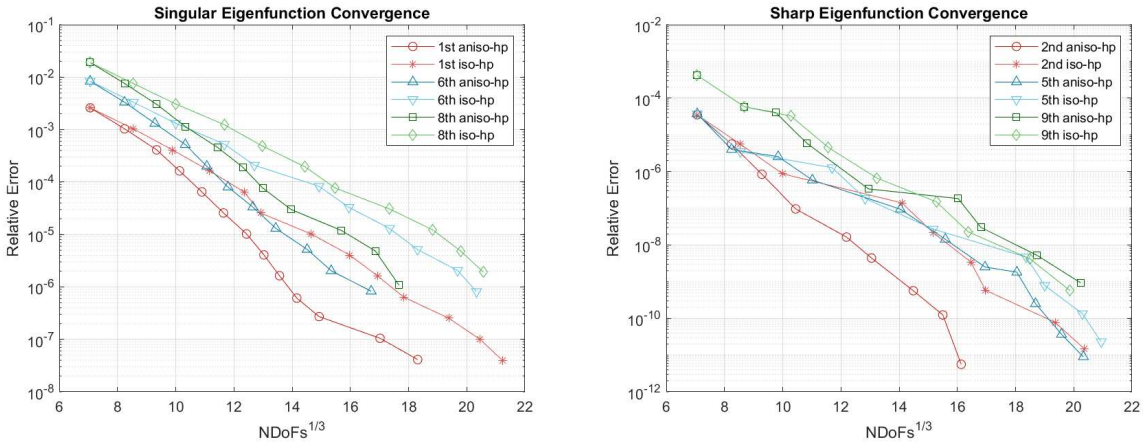


Figure 3.3: Comparison of convergence rates for the isotropic and anisotropic adaptive refinement algorithms. Behavior for the Singular and Sharp eigenfunctions are shown on a cubed-root–log scale

Figure 3.3 shows the solution convergence behavior for both classes of eigenfunction. Results are plotted on a cube-root–log scale where exponential convergence rates are indicated by straight lines. Each data point is associated with one refinement iteration and shows the relative error of the eigenvalue with respect to the number of degrees of freedom used in that iteration.

It is clear that exponential convergence is achieved by the isotropic and anisotropic versions of the adaptive algorithm. Figure 3.3 also indicates that anisotropic refinements can provide faster convergence rates and segnificant improvements in efficiency when the algorithm is applied to singular eigenfunctions. For all three singular cases and for the 2nd eigenpair, there is a significant reduction in the number of DoFs needed to achieve a given accuarcy, which expands with each iteration.

The same separation is not achieved for the 5th and 9th eigenpairs, indicating that further tuning of the algorithm is likely needed for cases where sharp and smooth behavior exist in close vicinity. Regardless of the small difference in isotropic and anisotorpic results, both algorithms achieve exponential convergence on these difficult eigenfunctions, clearly exhibiting the general viability of the methodology for other challenging problems in CEM.

Figure 3.4 shows the final refinement iteration meshes for the singular eigenpair experiments. The anisotropic and isotropic version are displayed side-by-side for comparison. Referencing

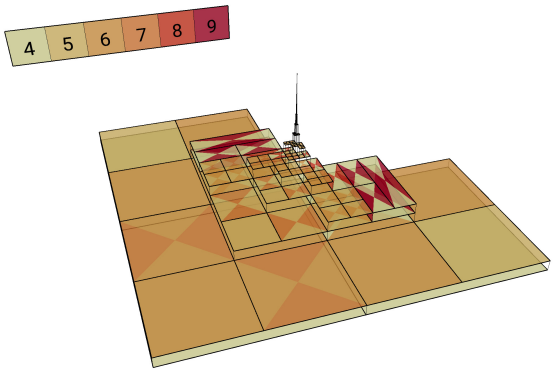
Figure 2.5, it is clear that the isotropic mesh arrangements are reminiscent of the *a priori* strategy, while the anisotropic strategies show less uniform refinement decisions, and tend to introduce more h-refinement. It is also clear that these algorithms rely heavily on anisotropic p-refinements which present significant NDoF savings. For example in 3.4f some elements have an expansion order of 10 in one direction and 4 in the other. An equivalent isotropic refinement would require the element to support 5 additional degrees of polynomials in the other direction which are not necessarily beneficial to solution accuracy.

3.5 Conclusion

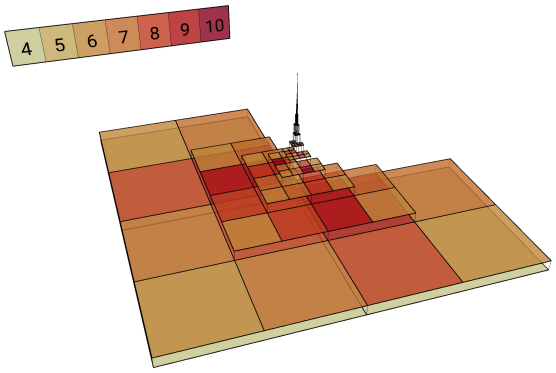
This chapter summarized the implementation details and theoretical considerations of an adaptive refinement algorithm which is able to leverage unconstrained anisotropic hp-adaptivity against challenging 2D FEM problems. For certain eigenfunctions, the algorithm not only achieved improved efficiency–as described in Chapter 2–but also achieved improved rates of convergence using anisotropic refinements. This gives significant credence to the value of RBS to practical CEM applications, particularly those where h-adaptivity is needed to address singular or discontinuous solution behavior.

It is also evident that there is further room for improvement in this arena, as the algorithm made some erroneously large refinement steps when applied to the 5th (Fig. 3.2e) and 9th (Fig. 3.2f) eigenfunctions. A more careful tuning of the somewhat arbitrary thresholds and directionality kernels used in the algorithm would likely overcome these issues without imposing a performance penalty on the singular cases. It may also be beneficial to train a less structured model (a neural-network, or the like) to translate the DWR and convergence-history data into a list of optimal refinements.

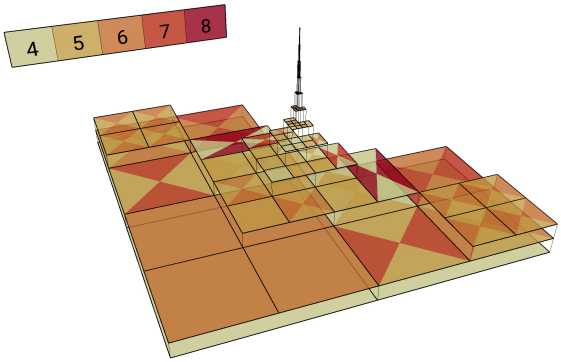
Regardless of its shortcomings, the algorithm described here is a solid stepping stone for more advanced iterative refinement algorithms that can leverage the relatively unconstrained mesh structure permitted by an RBS implementation of FEM.



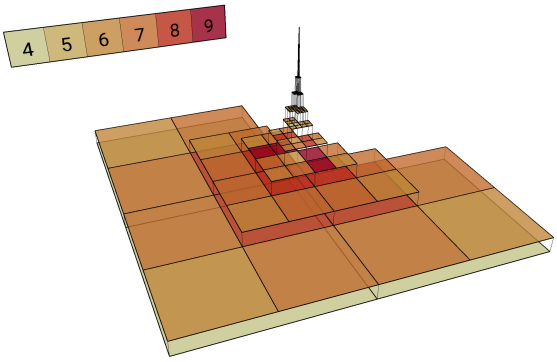
(a) 1st Eigenpair | Anisotropic Refinement



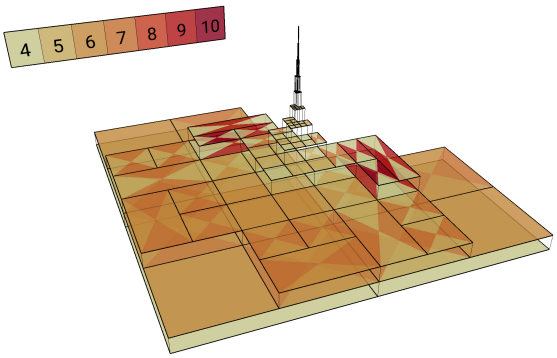
(b) 1st Eigenpair | Isotropic Refinement



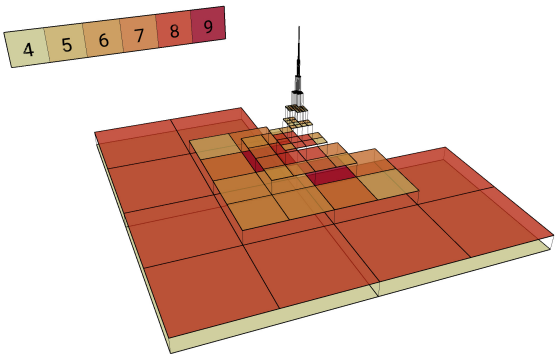
(c) 6th Eigenpair | Anisotropic Refinement



(d) 6th Eigenpair | Isotropic Refinement



(e) 8th Eigenpair | Anisotropic Refinement



(f) 8th Eigenpair | Isotropic Refinement

Figure 3.4: Comparison of final mesh states for Isotropic and Anisotropic adaptive refinement algorithms on singular eigenfunctions. The key in the top left of each figure represents the polynomial expansion orders. Multi-colored elements have been anisotropically p-refined

Chapter 4

The FEM_2D Open Source Library

4.1 Introduction

A portion of this research involved the implementation of an FEM software library which supports the hp-adaptivity model described in Chapter 2. An open source version of the library is available on Github[2] and crates.io [3] [17, 18]. This code was open-sourced for several reasons:

- As a direct proof-of-concept for the methodology
- To inspire and facilitate further research in this area
- To serve as the basis of useful simulation tools

The library is still in its infancy and lacks some of the feature-richness of more developed FEM libraries such as Deall.II; however, its hp-refinement model is fundamentally different, setting it apart from the existing FEM implementations. RBS itself prioritizes simplicity, and the design of fem_2d aims to capitalize on this fact by focusing on clean abstraction layers and easily understandable APIs. RBS also allows the code to be small–weighing in at about 7000 lines–making it straightforward to explore, understand, and contribute to.

4.1.1 Installation and the Rust Language

The fem_2d library was implemented in the Rust Programming Language. Although Rust is less popular than C++ (or other languages common to computational physics), it was chosen for its combination of high-performance, memory safety, and modern features such as Trait-based Generics, Closures, and JavaScript style iterators. The importance of these features is demonstrated in the example code within the following sections.

[2]https://github.com/jeremiah-corrado/fem_2d

[3]https://crates.io/crates/fem_2d

FEM_2D is easily added to any Rust project by including the following line the the project's TOML file under dependencies: `fem_2d = "0.1.0"`. This instructs the Rust compiler to download and link version 1.0 of the library (along with its dependencies).

4.1.2 Basic Usage

FEM_2D's clean and expressive nature are clearly exhibited in Figure 4.1. This example demonstrates a simple implementation of the Maxwell Eigenvalue Problem over a waveguide mesh while summarizing a few of the libraries key data structures and functions.

```
1  use fem_2d::prelude::*;
2
3  fn main() {
4      // Load the mesh from a file
5      let mut mesh = Mesh::from_file("./waveguide.json").unwrap();
6
7      // Apply some hp-Refinements
8      mesh.global_p_refinement(PRef::from(5, 3)).unwrap();
9      mesh.h_refine_elems(vec![2, 4, 8], HRef::u())).unwrap();
10
11     // Create a Domain from the refined mesh
12     let domain = Domain::from_mesh(mesh);
13
14     // Execute Galerkin sampling with an 8x8 grid of Guass-Quad points
15     let eigenvalue_problem = domain.galerkin_sample_gep_parallel::<
16             KOLShapeFn,     // basis function
17             CurlCurl,      // stiffness matrix integral
18             L2Inner,      // mass matrix integral
19     >(Some(8));
20
21     // Solve the eigenvalue problem, looking for solutions near 1.0
22     let eigenpair = nalgebra_solve_gep(eigenvalue_problem, 1.0).unwrap();
23     println!("Found Eigenvalue: {:.15}", eigenpair.value);
24 }
```

Figure 4.1: Solving the Maxwell Eigenvalue Problem using fem_2d

One of fem_2d's most important features is its generic API over basis functions and integrals shown on lines 16-18 of Figure 4.1. Here, **KOLShapeFn** defines a space of basis functions (Denoted by the letter B in previous Chapters). It can be swapped for any other Type that imple-

ments the libraries $ShapeFn$ Trait. **CurlCurl** and **L2Inner** are the integral rules associated with the Maxwell Eigenvalue Problem as described in Equation 2.1; both of which implement the libraries $Integral$ Trait. This generic API gives users a straightforward path to leverage the libraries functionality against other eigenproblems by creating custom implementations of $Integral$ and $ShapeFn$.

One might also notice the invocation of the `.unwrap()` method after some of the function calls. Each of these functions returns a `Result<T, E>` type, where T is the return type upon success, and E is an error type. This pattern indicates that an error may arise during the function's execution. For example, on line 5, the static method `Mesh::from_file` may return the Err variant of Result if the specified file is not found by the operating system. Because `.unwrap()` is appended to the end of the function call, the program will stop executing if an Error is returned.

```
1  fn try_solve_group(gep: &GEP, targets: Vec<f64>) -> Vec<EigenPair> {
2      targets
3          .iter()
4          .filter_map(
5              |target_eigenvalue| match slepc_solve_gep(
6                  gep.clone(),
7                  *target_eigenvalue,
8              ) {
9                  Ok(eigenpair) => {
10                     println!(
11                         "Found Eigenvalue: {:.15} for target: {:.2}",
12                         eigenpair.value, target_eigenvalue
13                     );
14                     Some(eigenpair)
15                 },
16                 Err(e) => {
17                     println!(
18                         "Failed to converge: `{}` for target: {:.2}",
19                         e, target_eigenvalue
20                     );
21                     None
22                 },
23             }
24         ).collect()
25 }
26
```

Figure 4.2: Gracefully failing to converge using the Result return type and the filter_map Iterator

More generally, the `Result<T, E>` Type gives the programmer a pathway to explicitly handle errors as they see fit. It is arguably more ergonomic and readable than a try-catch block, and incurs no additional run-time overhead. For example, if one needed to compute solutions for multiple target eigenvalues on the same eigenproblem, but wasn't sure that all of them would converge, they could implement something like the function shown in Figure 4.2. Here the Result returned by `slepc_solve_gep` is deconstructed using the **`match`** control flow operator. When the Ok variant is encountered–indicating that the eigensolver converged successfully–the solution is collected, otherwise if the Err variant is encountered, the error message is printed and the program moves on.

4.2 The Mesh Structure and hp-Adaptivity API

The Mesh data structure is largely responsible for fem_2d's unique functionality, and thus warrants some more in-depth discussion. This data structure embodies two distinct concepts: the physical characteristics the mesh and its transient refinement state. The library makes a distinction between these two concepts by separating the "finite element" into two separate data structures: the **Element** structure, which represents physical information (like geometric dimensions and material properties), and the **Elem**, which is used to keep track of the refinement state.

Upon Mesh construction, a group of **Elements** are constructed to capture the geometric layout of the given mode. All **Elements** are defined by four points in real space and store the material properties for that region of the model. Each is assigned an **Elem** which is defined over the same region in parametric space. The **Elem** is also assigned a polynomial expansion order of 1 in both directions and is designated as the root node in a tree of **Elem**s. In other words, the Mesh begins in the least-refined state possible for the given geometry.

An h-refinement is executed by appending 2 or more **Elem**s to a leaf in an Elem-Tree. All h-refinement methods return an Err when a non-leaf or non-existent **Elem** is specified for refinement. For the purpose of performance and simplicity, the **Elem**s are actually stored in a grow-able vector

```rust
let mut mesh = Mesh::from_file("./some_mesh.json").unwrap();

// isotopically h-refine all elems
mesh.global_h_refinement(HRef::t()).unwrap();

// anisotropically p-refine all elems
    //(+2 in the u-direction, +4 in the v-direction)
mesh.global_p_refinement(PRef::from(2, 4)).unwrap();

// anisotropically h-refine all elems connected
    // to some target_node (in the v-direction)
let target_node_id = 5;
mesh.h_refine_with_filter(|elem| {
    if elem.nodes.contains(&target_node_id) {
        Some(HRef::v())
    } else {
        None
    }
}).unwrap();

// positively p-refine all elems on the border of the mesh,
    // and negatively p-refine all other elems
mesh.execute_p_refinements(
    mesh.elems.iter().filter_map(|elem| {
        if elem.edges.iter().any(|edge_id| {
            mesh.edges[*edge_id].is_boundary()
        }) {
            Some(PRef::from(1, 1))
        } else {
            Some(Pref::from(-1, -1))
        }
    }).collect()
).unwrap();

```

Figure 4.3: Simple examples of hp-refinement using fem_2d's Mesh API

rather than a tree-like data structure. In other words, their ancestors and descendants are referenced internally as lists of indices instead of direct pointers.

A p-refinement is executed by modifying the polynomial expansion order of an **Elem** anywhere in an Elem-Tree. Similar to h-refinements, these methods return an Err if the specified **Elem** does not exist. It is important to note that p-refinements may be ineffectual if the DoFs on the designated **Elem** are inactive due to a combination of local and surrounding h-refinements.

A variety of hp-refinement methods are available for a wide range of scenarios in order to make the implementation of refinement algorithms as straightforward and unimpeded as possible. A few examples of these methods are shown in Figure 4.3. The entire list of refinement methods is available in the libraries documentation [4].

To provide a more realistic example of how this API may be used in practice, Figure 4.4 gives an implementation of the anisotropic hp-refinement strategy shown in Figure 2.5d. This is a finely tuned *a priori* strategy which relies on some pre-defied knowledge of the mesh layout (represented as a group of constants at the top the code example). It is only applicable to the specific mesh discussed in previous chapters, but gives a concrete example of a useful hp-refinement algorithm.

4.3 Conclusion

The fem_2d library is a small but powerful FEM code which is designed to be extended into other problem domains. It is also intended to be accessible enough for open-source developers or other researchers to work on adding new features and improving performance. Some low hanging fruit are: the implementation of curvilinear elements, vectorization of the integration code (via SIMD instructions or CUDA Kernels), and some intelligent pre-allocation procedure in the sparse matrix data structure. Additionally, the existence of a well documented 2D FEM code that supports RBS is intended to encourage further research in this area. This may include the development of an open-source 3D FEM code that supports RBS and anisotropic hp-refinement, or the development of more advanced adaptive refinement procedures.

[4] https://docs.rs/fem_2d/0.1.0/fem_2d/domain/mesh/struct.Mesh.html

```rust
const CENTRAL_NODE_ID: usize = 4;

const LEFT_ELEMENT_ID: usize = 2;
const CENT_ELEMENT_ID: usize = 0
const RGHT_ELEMENT_ID: usize = 1;

fn build_initial_mesh(mesh: &mut Mesh, k: usize) {
  let mut poly_order = k + 2;

  for i in 0..k {
    // build new layer around the central node
    mesh.h_refine_with_filter(|elem| {
      if elem.nodes.contains(&CENTRAL_NODE_ID) {
        Some(match elem.element.id {
          LEFT_ELEMENT_ID => HRef::U(Some(0)),
          CENT_ELEMENT_ID => HRef::T,
          RGHT_ELEMENT_ID => HRef::V(Some(0)),
          _ => unreachable!()
        }
      } else {
        None
      }
    }).unwrap();

    // set expansion orders
    mesh.set_expansions_with_filter(|elem| {
      let max_h_level = cmp::max(elem.h_levels.u, elem.h_levels.v);

      if max_h_level == (k+1) {
        if !elem.nodes.contains(&CENTRAL_NODE_ID) {
          Some(match elem.element.id {
            LEFT_ELEMENT_ID => [poly_order - 1, poly_order],
            CENT_ELEMENT_ID => [poly_order, poly_order],
            RGHT_ELEMENT_ID => [poly_order, poly_order - 1],
            _ => unreachable!()
          })
        } else {
          Some([poly_order, poly_order])
        }
      } else {
        None
      }
    }).unwrap();

    poly_order -= 1;
  }
}

```

Figure 4.4: Example implementation of an *a priori* anisotropic refinement strategy

Bibliography

[1] Jeremiah Corrado, Jake Harmon, and Branislav Notaros. A refinement-by-superposition approach to fully anisotropic hp-refinement for improved efficiency in cem, Oct 2021.

[2] M. M. Ilic, A. Z. Ilic, and B. M. Notaros. Efficient large-domain 2-D FEM solution of arbitrary waveguides using p-refinement on generalized quadrilaterals. *IEEE Transactions on Microwave Theory and Techniques*, 53(4):1377–1383, 2005.

[3] J. J. Harmon, J. Corrado, and B. M. Notaroš. A refinement-by-superposition hp-method for h(curl)- and h(div)-conforming discretizations. *TechRxiv*, Jun 2021.

[4] J. J. Harmon and B. M. Notaroš. Adaptive hp-refinement for time-harmonic Maxwell eigenvalue problems: Method and benchmarks. *In Review*, 2021.

[5] W Rachowicz. An anisotropic h-type mesh-refinement strategy. *Computer Methods in Applied Mechanics and Engineering*, 109(1):169–181, 1993.

[6] Dominik Schötzau, Christoph Schwab, and Rolf Stenberg. Mixed hp-fem on anisotropic meshes ii: Hanging nodes and tensor products of boundary layer meshes. *Numerische Mathematik*, 83(4):667–697, 1999.

[7] Vít Dolejší, Georg May, Filip Roskovec, and Pavel Solin. Anisotropic hp-mesh optimization technique based on the continuous mesh and error models. *Computers and mathematics with applications (1987)*, 74(1):45–63, 2017.

[8] Stefano Giani and Mohammed Seaid. Multi-hp adaptive discontinuous galerkin methods for simplified pn approximations of 3d radiative transfer in non-gray media. *Applied Numerical Mathematics*, 150:252–273, 2020.

[9] W. Gui and I. Babuška. The h, p and h-p versions of the finite element method in 1 dimension, part i. *Numerische Mathematik*, 49(6):577–612, Nov 1986.

[10] W. Gui and I. Babuška. The h, p and h-p versions of the finite element method in 1 dimension, part ii. *Numerische Mathematik*, 49(6):613–657, Nov 1986.

[11] W. Gui and I. Babuška. The h, p and h-p versions of the finite element method in 1 dimension, part iii. *Numerische Mathematik*, 49(6):659–683, Nov 1986.

[12] N. Zander, T. Bog, S. Kollmannsberger, D. Schillinger, and E. Rank. Multi-level hp-adaptivity: High-order mesh adaptivity without the difficulties of constraining hanging nodes. *Computational Mechanics*, 55(3):499–517, 2015.

[13] N. Zander, T. Bog, M. Elhaddad, F. Frischmann S. Kollmannsberger, and E. Rank. The multi-level hp-method for three-dimensional problems: Dynamically changing high-order mesh refinement with arbitrary hanging nodes. *Computer Methods in Applied Mechanics and Engineering*, 310:252–277, 2016.

[14] M. M. Kostić and B. M. Kolundžija. Maximally orthogonalized higher order bases over generalized wires, quadrilaterals, and hexahedra. *IEEE Transactions on Antennas and Propagation*, 61(6):3135–3148, 2013.

[15] M. Dauge. Benchmark computations for Maxwell equations for the approximation of highly singular solutions. https://perso.univ-rennes1.fr/monique.dauge/benchmax.html.

[16] Jake J. Harmon and Branislav M. Notaroš. Adaptive hp-refinement for 2-d maxwell eigenvalue problems: Method and benchmarks. *IEEE Transactions on Antennas and Propagation*, pages 1–1, 2022.

[17] Jeremiah Corrado. Fem 2d. https://github.com/jeremiah-corrado/fem_2d, 2022.

[18] Jeremiah Corrado, Jake Harmon, Branislav Notaros, and Milan M. Ilic. Fem 2d: A rust package for 2d finite element method computations with extensive support for hp-refinement. Feb 2022.