RICE UNIVERSITY

Application of Harmonic Coordinates to 2D Interface Problems on Regular Grids

by

Tommy L. Binford, Jr.

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

Doctor of Philosophy

Houston, Texas

November 2011

Contents

List of Tables

Chapter 1

Introduction

In this thesis, I present a method for solving numerically the following elliptic partial differential equation

$$-\nabla \cdot (\alpha(x)\ \nabla u) + \beta(x)\, u = f(x) \quad \text{in } \Omega \tag{1.1}$$

where the α and β are piecewise constant functions with relation to a multitude of interior interfaces that do not align with the nodes of a regular computational grid. The algorithm I develop is based on a finite element method by Owhadi and Zhang (2006) that incorporates fine scale behavior of the coeffieints into the finite element basis functions by way of a special coordinate transformation called harmonic coordinates. Owhadi and Zhang (2006) intentionally represent the support of the basis functions incorrectly resulting in a non-conforming finite element basis and thus diminishing the accuracy of the numerical solution. My implementation corrects the non-conformity of the basis by using an accurate polygon intersection algorithm I developed, which improves the accuracy of the method. The primary focus of this thesis is solving Eq. (1.1) with piecewise constant coefficient functions to provide an alternative finite element spatial discretization suitable for solving the acoustic wave equation on regular grids.

1.1 Motivation and Context

The model problem Eq. (1.1) for this work is motivated by the acoustic wave equation

$$-\nabla \cdot (\alpha(x)\ \nabla p) + \beta(x) \frac{\partial^2 p}{\partial t^2} = f(x,t) \quad \text{in } \Omega \tag{1.2}$$

where $\alpha(x)^{-1}$ is the density, $\beta(x)^{-1}$ is the bulk modulus, $f(x,t)$ is some (generally localized) acoustic source, and $p(x,t)$ is the acoustic pressure in the medium. Accurately solving Eq. (1.1) is a key component of seismic inversion where surface acoustic measurements are used to determine the structure of the upper crust in a noninvasive search for hydrocarbons. Thus, improving solutions to the wave equation will improve the accuracy of seimic inversion.

The regular grid finite difference time-domain (FDTD) method is widely used for the numerical solution of wave propagation problems of this type due to its ease of implementation and efficiency in both memory access and storage. For materials whose mechanical properties vary smoothly with position, FDTD yields an optimal approximation in both space and time (Cohen, 2002). For the simulation of such waves, FDTD is both efficient and accurate. However, in studying composite materials, such as the Earth's crust, mechanical properties like density and bulk modulus change abruptly from region to region. With detailed composite variations, material transitions will occur inside grid cells of the regular grid spatial discretization for FDTD. Where coefficients are discontinuous, the smoothness of the solution is reduced, meaning fewer derivatives exist. The degree of smoothness of a function determines the truncation error of local Taylor approximations. Therefore, wave solutions in discontinuous media are poorly approximated by FDTD since the convergence of FDTD is determined by how well local Taylor approximation can represent the solution within grid cells. Formal analysis of the truncation error for FDTD solutions to the second order wave equation has shown that grid-interface misalignment for discontinuous material transitions contributes a component that is first-order in space (Brown, 1984). Thus, the error due to grid-interface misalgnment will dominate and reduce the overall accuracy of the method.

Errors induced by material transitions within grid cells can be corrected by aligning a conventional grid with the interface. Though alignment is difficult to accomplish for conventional grid methods, the staggered grid schemes used to solve the pressure-velocity formulation of the acoustic wave equation present an additional problem. The staggered grid FDTD method uses one grid for the pressure and an offset grid for the velocity field. One can imagine that aligning the interfaces with one grid will introduce a misalignment with the other grid due to the offset, and this has been quantified by Symes and Vdovina (2009). Therefore, any standard finite difference method will exhibit reduced accuracy when applied to problems with discontinuous coefficient functions due to grid-interface misalignment error.

The variational methods, continuous (cGTD) and discontinuous Galerkin (dGTD) time domain, are similar to FDTD on regular grids because they exploit the same kind of approximation properties of a polynomial basis to establish convergence, which is akin to local Taylor approximation at the low order. The misalignment error discussed above also reduces the accuracy of cGTD (Cohen, 2002, pp. 208-209) and dGTD (Wang, 2009), since convergence depends on how well interpolation by smooth functions within grid cells can represent the solution. Thus, only when the solution is smooth will the approximation error be small for a given regular spatial discretization of a medium with complicated material changes whether that discretization is determined by finite differences or based on finite elements.

Both cGTD and dGTD finite elements may also use irregular or unstructured grids that can be fitted to the complex changes in the material properties. Using fitted grids that follow changes in the material properties for the finite element family of methods avoids some of the approximation error inherent to unfitted grids since the polynomial approximation spaces admit jumps in the derivative of the solution at element boundaries (Strang and Fix, 1973, pg. 14). It has been shown explicitly for the pressure-velocity formulation of the acoustic wave equation that a fitted spatial grid also recovers the expected accuracy for dGTD (Wang, 2009). However, meshes fitted to complex interfaces found in large models, such as those encountered in exploration geophysics, will lead to smaller elements and may require manual grid manipulation. With smaller elements, the computational cost for each time step increases as the number of spatial degrees of freedom grows, gaining accuracy but leaving behind the efficiency of regular grid methods.

A benefit of finite element methods over finite difference is the ease with which the approximation properties can be modified. By changing the basis function used to approximate the solution, the accuracy of the method can be improved. The theoretical framework and construction of the finite element matrices remains essentially the same. Therefore, I choose to focus on finite element methods for the spatial discretization of the wave equation on regular triangular grids.

Applying the finite element method to the spatial component of the wave equation with zero source function leads to a semidiscrete formulation

$$\frac{\partial^2}{\partial t^2}\int_\Omega \beta p_h \, v_j \, dx + \int_\Omega \nabla p_h \alpha \, \nabla v_j \, dx = 0$$

where the functions v_j reside in some suitable discrete space V_h of piecewise polynomials (homogeneous Dirichlet boundary conditions). The estimate p_h for the true pressure field is then separated by using $v_j \in V_h$ to approximate the spatial part and writing

$$p_h\left(x,t\right) = \sum_j \hat{p}_j\left(t\right)\ v_j\left(x\right)$$

where the coefficients now vary in time. Substituting the pressure field p_h into the wave equation Eq. (1.2) leads to a system of ordinary differential equations in the time-varying coefficients $\hat{p}_j$

$$\mathcal{M}\frac{\partial^2 \hat{\mathbf{p}}}{\partial t^2} + \mathcal{S}\hat{\mathbf{p}} = 0$$

where $\mathcal{M}, \mathcal{S}$ are the finite element mass and stiffness matrices defined by the integrals

$$\mathcal{M} = \int_\Omega \beta v_i v_j \, dx \quad \text{and} \quad \mathcal{S} = \int_\Omega \nabla v_i \alpha \nabla v_j \, dx. \tag{1.3}$$

The semidiscrete form reveals that the spatial discretization can be considered independent of the time-stepping used to advance the solution. Since the spatial discretization of the domain creates the grid-interface alignment problem observed in the reflection times of traveling waves, it makes sense to consider improving only the spatial discretization terms. In the case of finite elements on regular grids, improving the spatial approximation means using a different finite dimensional space V_h of basis functions, which are used to construct the mass and stiffness matrices. Changing the basis functions used for the spatial part of the pressure field modifies the entries of the mass and stiffness matrices which determine how coefficients $\hat{\mathbf{p}}(t)$ will react to discontinuities in the medium when time-stepping is implemented.

1.2 Problem Statement

Since the finite element spatial discretization for the wave equation is independent of the time stepping, I focus on establishing an approximation scheme for the elliptic equation

$$-\nabla \cdot (\alpha(x)\ \nabla u) + \beta(x)\, u = f(x) \quad \text{in } \Omega \tag{1.4}$$

with suitable Dirichlet boundary conditions and piecewise constant coefficients. This will provide insight into reducing the error due to grid-interface misalignment because the weak formulation of the elliptic model problem involves the same mass and stiffness matrices as the semi-discrete form of the wave equation.

The approximate solution to Eq. (1.4) is sought on a coarse, structured mesh $\mathcal{T}$ of triangular elements. However, variations on scales smaller than the element size of $\mathcal{T}$ means that some form of homogenization is necessary. I focus on a finite element homogenization method developed by Owhadi and Zhang (2006) that uses the component-wise solution to the following auxiliary problem

$$\begin{aligned} \nabla \cdot \alpha\,(x)\,\mathbf{F} &= 0 \qquad \text{in } \Omega \\ \mathbf{F}(x) &= x \qquad \text{on } \partial\Omega \end{aligned} \tag{1.5}$$

to incorporate fine-scale structures into the coarse-scale basis functions. The solution $\mathbf{F}$ to this auxiliary problem is called a *harmonic map* and qualifies as a coordinate transformation in 2D provided the coefficient function α satisfies uniform ellipticity and boundedness. In all cases, a numerical solution to Eq. (1.5) using standard finite elements is computed on a fine mesh $\mathcal{E}$. A fitted mesh $\mathcal{E}$ is used for piecewise constant coefficient functions where well-defined interfaces exist, such as the case for domains in problems of reflection seismology. This means there are two meshes for any given problem: a fine mesh $\mathcal{E}$ for the auxiliary problem and a coarse mesh $\mathcal{T}$ for the solution to the model problem.

For planar problems such as Eq. (1.4), the solution $u \in \mathcal{H}^1(\Omega)$ and is not smooth enough to justify the standard finite element error estimates. Owhadi and Zhang (2006) show that the composition $u \circ \mathbf{F}^{-1} \in \mathcal{H}^2(\Omega)$, which does satisfy the usual interpolation assumptions for piecewise linear functions and results in an optimal error estimate

$$\|u \circ \mathbf{F}^{-1} - w_h\|_{\mathcal{H}^1(\Omega)} = \mathcal{O}(h),$$

where w_h is an approximation using piecewise linear basis functions. Instead of approximating $u \circ \mathbf{F}^{-1}$, Owhadi and Zhang (2006) incorporate the harmonic map $\mathbf{F}$ into the basis and prove that

$$\|u - u_h\|_{\mathcal{H}^1(\Omega)} = \mathcal{O}(h),$$

where $u_h \in X_h$ and

$$X_h = \left\{v \circ \mathbf{F} : \text{ for all } K \in \mathcal{T},\, v|_K \in \mathbb{P}_1(K)\right\}.$$

Thus, the usual piecewise linear basis functions are composed with $\mathbf{F}$ to form a high-resolution approximation space that recovers optimal order accuracy.

Basis functions for X_h are defined on a fine mesh that is different from the one where $\mathbf{F}$ and the coefficient functions are known. A remedy for this inconvenience is provided by Owhadi and Zhang (2006) in the form of a localized implementation. Instead of forming the composite basis functions using the usual piecewise linear functions on each element K, they form the approximation space

$$Z_h = \left\{v : \text{ for all } K \in \mathcal{T},\, \phi \in \mathbb{P}_1(\mathbf{F}(K)),\, v|_K = \phi \circ \mathbf{F}|_K\right\}.$$

The elements $\mathbf{F}(K)$ are triangles formed by mapping only the vertices of coarse elements K to harmonic coordinates. A consequence of this construction is the localized composite basis functions are non-conforming, meaning that the basis no longer satisfies continuity at element edges in the coarse mesh $\mathcal{T}$. This non-conformity increases the error level and reduces the rate of convergence of the finite element method.

The true domain for each of these composite basis function in Z_h is not a triangular element. Restricting the basis to the coarse elements K inaccurately represents the support of these functions leading to a non-conforming basis. In this thesis, I develop a unique method for reproducing the support of these basis functions by forming mesh-element intersections. Since the functions ϕ described in Z_h are piecewise linear on $\mathbf{F}(K)$, I form the mesh-element intersections of each mapped coarse element $\mathbf{F}(K)$ with elements in the mapped mesh $\mathbf{F}(\mathcal{E})$ to accurately represent the support of these basis functions. I show that these intersections together with the piecewise linear approximation of $\mathbf{F}$ reduce the stiffness matrix assembly to a simple summation on each element, which means no quadrature scheme is required. Further, I exploit these intersection objects, which are each collections of simple polygons (up to hexagons), to establish a quadrature-independent mass matrix assembly algorithm.

The initial cost of solving the harmonic map problem and computing mesh intersections is expected to be amortized over a large number of calculations. In the case of elliptic problems this means a large number of source functions for a given geometry. The power of this method is that the coarse approximation is accurate and needs to be computed once per simulation. Applications to the wave equation are obvious since many time steps are needed for each calculation. Thus, the purpose of this method is to compute coarse operators (mass and stiffness matrices) that produce accurate results and use that coarse approximation to model physical phenomena.

I completely describe these new finite element assembly algorithms in Chapter 3. Additionally, I present an implementation of the original localized method using mesh-element intersections which is distinct from the implementation of Owhadi and Zhang (2006). In Chapters 4, I describe the mesh-element intersection algorithm and software framework developed for and used in this thesis. This framework is exercised in Chapter 5 to demonstrate that the new assembly algorithms result in improved accuracy over the original non-conforming method. Finally, I discuss conclusions and future work in Chapter 6.

Chapter 2

Background and Literature Review

Interface problems present a unique difficulty for the finite element method on regular grids. Although solvability of these problems is established in Hilbert space $\mathcal{H}^1(\Omega)$ by basic theory, the issue of solving elliptic problems numerically is much more complicated when the coefficients are not represented by smooth functions. I introduce the weak formulation for the model problem and briefly discuss these conditions for solvability. I then present the standard finite element method and discuss how the convergence rate is adversely affected on regular grids by coefficients with reduced smoothness. This leads to an exploration of methods, all based on finite elements, that seek to improve accuracy of the numerical solution by incorporating sub-grid coefficient fluctuations into the finite element approximation space.

2.1 Solvability of the Model Problem

Consider a domain $\Omega \subset \mathbb{R}^2$ with a closed boundary $\partial\Omega$. The goal is to establish the conditions under which

$$\begin{aligned} -\nabla \cdot (\alpha(x)\ \nabla u) + \beta(x)\, u = f(x)\,, \qquad & \text{in } \Omega, \\ u = g, \qquad & \text{on } \partial\Omega, \end{aligned} \tag{2.1}$$

has a unique solution when α, β are discontinuous or even $\alpha \in [L^\infty(\Omega)]^{2\times 2}, \beta \in L^\infty(\Omega)$ (bounded, measurable). Interface problems, which are the focus of this thesis, have α, β discontinuous across well-defined boundaries within the domain. However, more general statements about the solvability of the model problem can be made without such a restriction on the smoothness of the coefficients. Before restricting the focus to piecewise constant coefficients, I will discuss what it means to be a solution in the classical and weak sense and how the smoothness of the coefficients must be restricted in each case.

Classical Solutions

One would likely seek solutions u that satisfy the differential equation Eq. (2.1) in the classical sense: replace u with the proposed solution, and apply the differential operator revealing an equivalence between the right- and left-hand sides along with agreement at the domain boundary. However, with $f \in \mathcal{C}^0(\Omega)$ such a solution would need to be at least twice continuously differentiable, that is, $u \in \mathcal{C}^2(\Omega)$. Regularity assumptions on the coefficient functions are similarly strict requiring $\alpha \in [\mathcal{C}^1(\Omega)]^{2\times 2}$, $\beta \in \mathcal{C}^0(\Omega)$. These conditions are enough to ensure $u \in \mathcal{C}^2(\Omega)$ in 1D. Yet, even with $\alpha = 1$ and $\beta = 0$ in 2D, which corresponds to the simple Poisson problem, it is well-known that a classical solution may not exist when $f \in \mathcal{C}^0(\Omega)$ (Haroske and Triebel, 2008). The fact is physical problems can have solutions that are not differentiable in the classical sense motivates the definition of a *generalized* or *weak solution*, also known as a *variational solution* due to variational principle used to establish it.

Weak Solution

The weak formulation arises from the application of the variational principle to the model problem. Multiply both sides on the differential equation by smooth test functions $\varphi \in \mathcal{C}_0^\infty(\Omega)$ that are zero on the boundary $\partial\Omega$ and integrate over the domain Ω

$$-\int_\Omega \nabla \cdot (\alpha\,(x)\;\nabla u)\;\varphi + \int_\Omega \beta\,(x)\,u\,\varphi = \int_\Omega f\,\varphi. \tag{2.2}$$

Applying integration by parts to the second order term, the above expression takes the form

$$\int_\Omega \nabla\varphi \cdot (\alpha\,(x)\;\nabla u) + \int_\Omega \beta\,(x)\,u\,\varphi = \int_\Omega f\,\varphi. \tag{2.3}$$

The boundary term from integrating by parts is zero since $\varphi = 0$ on the boundary by definition. Notice that the functions φ and u as expressed need only be once differentiable. Thus, the weak formulation imposes weaker conditions on the smoothness of the solution than the classical form. Relaxing the smoothness requirement means that the space of acceptable solutions is expanded to include functions that do not satisfy the equation in the classical sense.

This class of acceptable solutions is the Hilbert space $\mathcal{H}^1(\Omega)$ defined as

$$\mathcal{H}^1(\Omega) = \{u : \Omega \to \mathbb{R} : \|u\|_{L^2(\Omega)} + \|\nabla u\|_{L^2(\Omega)} < \infty\},$$

and the L^2-norms are defined as

$$\|u\|^2_{L^2(\Omega)} = \int_\Omega |u|^2, \qquad \|\nabla u\|^2_{L^2(\Omega)} = \int_\Omega \nabla u \cdot \nabla u$$

Any solution $u \in \mathcal{H}^1(\Omega)$ must also satisfy the Dirichlet boundary condition g. The class of functions in $\mathcal{H}^1(\Omega)$ that are zero on the domain boundary are denoted by $\mathcal{H}^1_0(\Omega)$. Boundary conditions are then enforced by seeking solutions u such that $u = w + \tilde{u}$, where $\tilde{u} \in \mathcal{H}^1_0(\Omega)$ solves the differential equation with homogeneous boundary conditions and the trace, or restriction of $w \in \mathcal{H}^1(\Omega)$ in the to the boundary is g. Essentially the solution $\tilde{u}$ is lifted from $\mathcal{H}^1_0(\Omega)$ to satisfy the boundary condition. Thus, the solution $u \in \mathcal{H}^1(\Omega)$ is sought such that $u - w \in \mathcal{H}^1_0(\Omega)$. Any problem with suitably smooth boundary data can be transformed into a homogeneous boundary value problem by this approach. I mention this because many results are presented in terms of homogeneous boundary conditions without such a comment. For the class of functions satisfying the Dirichlet boundary conditions, I adopt the notation $\mathcal{H}^1_E(\Omega) = w + \mathcal{H}^1_0(\Omega)$ with $w \in \mathcal{H}^1(\Omega)$ and $\operatorname{trace}(w) = g$ used by Strang and Fix (1973, pg. 70)

This same relaxed smoothness also applies to the test functions φ since Eq. (2.3) does not require such high regularity as $\mathcal{C}^\infty_0(\Omega)$. The space of infinitely differentiable functions $\mathcal{C}^\infty_0(\Omega)$ is dense in $\mathcal{H}^1_0(\Omega)$, so relaxing the smoothness to $\varphi \in \mathcal{H}^1_0(\Omega)$ is a perfectly acceptable approximation.

Thus, the weak formulation of Eq. (2.1) is: Find $u \in \mathcal{H}^1_E(\Omega)$ such that

$$-\int_\Omega \nabla\varphi \cdot (\alpha\,(x)\;\nabla u) + \int_\Omega \beta\,(x)\,u\,\varphi = \int_\Omega f\,\varphi \tag{2.4}$$

is true for all $\varphi \in \mathcal{H}^1_0(\Omega)$. This is the usual weak formulation that serves as the starting point for the finite element method. The existence and uniqueness of weak solutions under the conditions of the model problem Eq. (2.1) follow directly from the Lax-Milgram theorem. For a detailed treatment and explanation of these concepts see, e.g., Evans (1998), Brenner and Scott (2002), and Ciarlet (2002).

2.2 Finite Element Method

Provided the model problem is well-posed, a weak solution exists and a powerful method for approximating that solution is the finite element method. The idea of the finite element method is to replace the continuum function spaces $\mathcal{H}^1(\Omega)$ with a proper subset of simple functions. This discretization starts with a mesh of the domain.

Let $\mathcal{T}_h$ be a mesh of triangular elements of $\Omega \in \mathbb{R}^2$. Each element $K \in \mathcal{T}$ has size h_K which is the diameter of a ball enclosing the element given by

$$h_K = \max_{p,q \sim K} \operatorname{dist}(p, q)$$

where $\operatorname{dist}(\cdot,\cdot)$ is the Euclidean distance function, and $p \sim K$ means p is a vertex of K. The element size associated with the triangulation $\mathcal{T}$ is the largest ball diameter over the whole mesh

$$h = \max_{K} h_K. \tag{2.5}$$

Error bounds stated here use this definition of h for the mesh size.

Consider the set of piecewise continuous functions

$$P_h^1(\Omega) = \left\{ v \in \mathcal{C}^0(\mathcal{T}_h) : \forall j, v|_{K_j} \in \mathbb{P}_1(K_j) \right\}, \tag{2.6}$$

where $\mathbb{P}_1$ is the space of first-order polynomials. In this case, functions in P_h^1 are linear functions on each element $K \in \mathcal{T}_h$. This space of piecewise functions is $\mathcal{H}^1(\Omega)$-conformal, meaning that P_h^1 is a subset of $\mathcal{H}^1(\Omega)$.

The discretized weak formulation for the model problem stated in terms of the piecewise linear approximation space is: Find $u_h \in P_h^1 \cap \mathcal{H}_E^1(\Omega)$ such that

$$-\int_\Omega \nabla\varphi \cdot (\alpha\,(x)\; \nabla u_h) + \int_\Omega \beta\,(x)\, u_h\, \varphi = \int_\Omega f\, \varphi \tag{2.7}$$

is true for all $\varphi \in P_h^1 \cap \mathcal{H}_0^1(\Omega)$. Provided that the true solution $u \in \mathcal{H}^2(\Omega)$, and $\alpha \in \mathcal{C}^0(\Omega)$ the difference $u - u_h$ measured in the $L^2(\Omega)$-norm behaves as

$$\|u - u_h\|_{L^2(\Omega)} = \mathcal{O}(h^2),$$

where h is the mesh size in Eq. (2.5). As the mesh size $h \to 0$, the difference $u - u_h$ converges quadratically to zero. The true solution loses some regularity when α is piecewise constant rather than continuous. This loss of regularity means that $u \in \mathcal{H}^1(\Omega)$ and the standard error estimate giving a quadratic convergence rate no longer applies. A direct consequence of this result is a slower rate of convergence.

To see this explicitly, consider the following simple 1D example in the context of linear interpolation. A governing factor in the finite element error bounds is the interpolation error of the underlying approximation space P_h^1. The piecewise linear

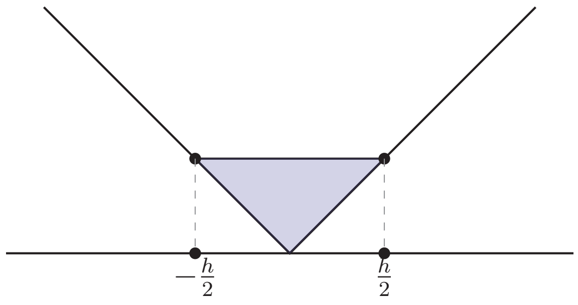


Figure 2.1: With a discontinuity in the derivative, piecewise linear functions do not necessarily provide an optimal approximation. In this figure, the shaded region shows the where a piecewise linear interpolant fails to capture the behavior of this continuous function in the interval $-\frac{h}{2} \leq x \leq \frac{h}{2}$. This results in an $\mathcal{O}(h^{\frac{3}{2}})$ error in the $L^2([-1,1])$-norm.

function

$$v(x) = \begin{cases} -x & -1 \leq x \leq 0 \\ \\ x & 0 \leq x \leq 1 \end{cases}$$

is continuous, and its derivative is bounded in $L^2([-1,1])$. However, the second derivative of v is not in $L^2([-1,1])$ which means $v \in \mathcal{H}^1([-1,1])$ but $v \notin \mathcal{H}^2([-1,1])$. Suppose the interval $[-1,1]$ is subdivided in such a way that the point $x = 0$ is always at the midpoint of a sub-interval, as in Fig. 2.1. A piecewise linear interpolation v_h of v on this grid will exactly match the function for $x \leq -\frac{h}{2}$ and $x \geq \frac{h}{2}$, but the interpolating polynomial in the interval $-\frac{h}{2} \leq x \leq \frac{h}{2}$ is simply $v_h(x) = \frac{h}{2}$. This means the only error contribution arises from the difference $v - v_h$ within the interval $-\frac{h}{2} \leq x \leq \frac{h}{2}$. In the $L^2([-1,-1])$-norm, this error is easily computed, and the rate of convergence estimated by reducing h is

$$\|u - u_h\|_{L^2([-1,1])} = \mathcal{O}(h^{\frac{3}{2}})$$

rather than $\mathcal{O}(h^2)$. Attempting to approximate a function in this way, by allowing kinks within intervals, affects the accuracy of the piecewise linear interpolation and degrades the convergence rate.

Finite elements, like the finite difference method, assume some degree of smoothness within elements. The simple example above shows that when this condition is violated the consequence is slower convergence. This is precisely how grid-interface misalignment affects the accuracy of finite element solutions. The challenge lies in constructing an accurate approximation to the solution u of the model problem when interfaces do not align with the computational grid. With regular grid finite element discretizations, improving the approximation of the solution must involve augmenting the basis so the interpolating function used in the finite element method can capture such sub-grid behavior. One should note that, although $v \in \mathcal{H}^1(\Omega)$, a good approximation can be obtained by aligning the finite element mesh with interfaces. Alignment works because the piecewise linear approximation, which is also in $\mathcal{H}^1(\Omega)$, allows jumps in the derivative provided they occur at element boundaries.

2.3 Local Basis Modifications

A direct approach to improving the accuracy of fine elements is to apply physical constraints to the approximation space at the interface. Methods for constructing such approximations for interface problems are rooted in the finite difference immersed boundary method (IBM) constructed to model flow around a flexible heart valve (Peskin, 1972). The focus of IBM was moving interfaces, but the method inspired the immersed interface method Leveque and Li (1994), which has been used to solve equations such as Eq. (1.4) and the wave equation in pressure-velocity form in media with stationary interfaces. Finite difference stencil coefficients in IIM are modified by incorporating continuity of the solution and the flux at an interface into the local Taylor approximations. Applying these continuity conditions has been shown to dramatically improve the accuracy of the finite difference method Leveque and Li (1994) when a single interface crosses an element.

A direct extension of this methodology is to apply these same continuity conditions to construct finite element basis functions. (Li, 1998) used these conditions to establish the immersed finite element method (IFEM). A major benefit of IFEM is the continuity conditions give usual piecewise linear basis functions when the coefficient function is smooth. Thus, contributions to the finite element matrices are unaffected away from interfaces. Using these these modified basis functions to construct finite element mass and stiffness matrices improves the solution accuracy on regular grids for Eq. (1.4) (Li, 1998), the heat equation (Li and Ito, 2006), and models for electromagnetic ion flow (Kafafy et al., 2005). Although IFEM does provide a means to recover accuracy for simple interface problems, recovering accuracy by this method requires that the computational grid be fine that each element is crossed by only a single interface.

These immersed methods provide no simple means to construct the stencil coefficients or finite element matrices for very complex regions with multiple interfaces crossing single elements, which is often encountered in seismic modeling. With such complex media, a more general means of including sub-grid information in the finite element basis functions is necessary. Regardless, the general idea of incorporating small scale information into a large scale approximation is valuable.

Using local fine scale information to develop coarse scale approximations has been researched extensively in the field of numerical homogenization (Bensoussan et al., 1978). Here the goal to compute effective discrete operators in the case of periodic, randomly varying coefficient functions. Fine-scale cell problems are solved to infer effective, coarse scale representations of the material properties. Homogenization by this approach assumes that there is a separation of scales (Papanicolaou, 1998). That is, only changes at small and large scales are captured with no effects from intermediate scales. However, a detailed study of well-log data shows that there is actually a continuum of scale information present in the sound velocity measurements (Herrmann, 1997). Therefore, the assumptions of scale separation and periodicity mean that this kind of asymptotic homogenization theory is not well-suited for seismic modeling, which is the motivation for this thesis.

The basic methodology of solving individual cell problems was adopted and extended in the development of multi-scale finite elements (MSFEM) (Hou and Wu, 1997; Efendiev and Hou, 2000). Multiscale methods use basis functions that are the solutions of a differential equation of the form

$$\begin{aligned} \nabla\, \alpha\,(x)\,\nabla\phi_j &= 0 \quad \text{in } K, \\ \phi_j &= g_j \quad \text{on } \partial K, \end{aligned} \tag{2.8}$$

on each triangular element K where the index $j = 1, 2, 3$ corresponds to the nodes of K. Most applications choose $g_j = x$ so the solutions satisfy the nodal property $\phi_j\,(x_i) = \delta_{ij}$ and maintain linearity on the boundary of the element. The result of this choice is a set of basis functions that have interior variations guided by the material property α and appear as standard finite element basis functions at the edges of each element K in the mesh. An important consequence of this boundary condition is the functions ϕ_j are restricted from fine scale movement at element edges. This is improved by applying oscillatory boundary conditions.

Under all but the simplest cases of coefficient functions α, Eq. (2.8) must be solved numerically. The resolution of the discrete problem associated with Eq. (2.8) is chosen such that all small scale information of interest contained in K will be incorporated into the basis functions ϕ_j. Therefore, there are two mesh sizes for each element: a fine scale mesh where possibly rapid coefficient variations occur, and a coarse mesh where the approximate solution is represented. Here the finite element mass and stiffness matrices, which are discrete operators, are computed at the coarse grid scale by accurately performing the integrations in Eq. (1.3) using these high resolution basis functions instead of simple piecewise linear functions.

There are two issues that introduce error constructing the basis functions in this manner. Since the local material variations affect the solution globally, using locally constructed MSFEM basis functions can induce boundary layers which increase in the error. This component of error is usually addressed by oversampling; that is, extending the local domain beyond a single element to allow different boundary conditions at element edges (Efendiev and Hou, 2000). The other component of error is associated with the cell resonance phenomenon, which is encountered in numerical homogenization theory (Bensoussan et al., 1978). Cell resonance is observed in problems with many scales when the coarse scale coincides with one of the fine scales. A Petrov-Galerkin MSFEM strategy was successfully used by Hou et al. (2004) to reduce cell resonance for low contrast media $(2 : 1)$.

It was recognized by Kozlov (1980) that the researchers in periodic homogenization had been implicitly applying a special type of coordinate transformation to compute averaged operators. Following the terminology in General Relativity, Kozlov (1980) coined the term *harmonic coordinates* to describe the transformation

$\mathbf{F} = (F_1, F_2, \ldots, F_d)$ whose components satisfy

$$\nabla \cdot \alpha(x)\, \nabla \mathbf{F} = 0 \tag{2.9}$$

component-wise with appropriate boundary conditions. Allaire and Brizzi (2004) follow this approach and use harmonic coordinates element-wise to construct MSFEM basis functions by forming the composition of standard piecewise linear basis functions with the transformation $\mathbf{F}$ that satisfies

$$\nabla\, \alpha(x)\, \nabla F_j = 0 \quad \text{in } K, \tag{2.10}$$

$$F_j = x_j \quad \text{on } \partial K, \tag{2.11}$$

for $j = 1, 2$. Solutions of Eq. (2.10) are computed numerically on a fine mesh in each element K of the mesh. These locally constructed composite basis functions $\psi_j \circ \mathbf{F}$, where ψ_j are piecewise linear on the coarse elements, produce a method that is equivalent to MSFEM. Their method separates the fine scale computation from the coarse basis functions. The benefit of this composition approach is that the method is independent of the order of the coarse basis functions.

There is no assumption of periodicity inherently applied in the construction of MSFEM basis functions and algorithms. However, almost all theoretical work is confined material properties that vary periodically (Hou and Wu, 1997; Hou et al., 1999). One exception is the recently developed MSFEM interface method (Chu et al., 2010), which has the same optimal-order convergence as IFEM and is similarly limited to single-element interface crossings.

2.4 Global Basis Modifications

Let the global harmonic map on the model problem domain Ω be the vector $\mathbf{F} = (F_1, F_2)$ that satisfies

$$\begin{aligned} \nabla \cdot \alpha\left(x\right)\mathbf{F} = 0, \qquad & \text{in } \Omega, \\ \mathbf{F}(x) = x, \qquad & \text{on } \partial\Omega, \end{aligned} \tag{2.12}$$

component-wise. Alessandrini and Nesi (2001, Theorem 4) show that $\mathbf{F}$ is a homeomorphism (continuous, continuous inverse, bijective) in 2D even when $\alpha \in [L^{\infty}(\Omega)]^{2\times 2}$. This result provides the necessary justification for using $\mathbf{F}$ as a coordinate transformation. However, this result does not extend to higher dimensions without further restrictions on the coefficient function α. An important counter-example by Briane et al. (2004, Corollary 1) shows that the Jacobian of $\mathbf{F}$ can change sign when the contrast in the coefficients is high enough. When the Jacobian of a map changes sign this signals that orientation is not preserved and the map is not invertible. Although this high-contrast example is pathological from the standpoint of seismology, it does limit the extension of harmonic maps in 3D since we are no longer guaranteed a coordinate transform by solving Eq. (2.12) with general coefficients. The setting for this thesis is 2D, so the harmonic map $\mathbf{F}$ is guaranteed to be a homeomorphism.

Following the composition rule of Allaire and Brizzi (2004), Owhadi and Zhang

(2006) show that the solution with respect to global harmonic coordinates satisfies a non-divergence form of the model problem. To see this, suppose $\mathbf{F}$ satisfies Eq. (2.12) on the domain Ω. Using this coordinate transform, write $u(x) = \widetilde{u} \circ \mathbf{F}(x)$. Applying the chain rule, the gradient of the composite function u is

$$\nabla_x u = D\mathbf{F}^T(\nabla_y \widetilde{u}) \circ \mathbf{F},$$

where $D\mathbf{F}$ is the Jacobian matrix of the map $\mathbf{F}$ and ∇_x denotes the gradient with respect to coordinates $x = (x_1, x_2)$ and ∇_y denotes the gradient with respect to coordinates $y = (y_1, y_2)$ ($\mathbf{F}$-coordinates). In general, α is a real, 2×2 matrix with each component in $L^\infty(\Omega)$. With this matrix, the second order operator in Eq. (1.1) is obtained by multiplying by the coefficient matrix α and applying the chain rule once more to write

$$\begin{aligned}\nabla_x \cdot \alpha\,(x)\,\nabla_x u = {} & (\nabla_x \cdot \alpha\,\nabla_x \mathbf{F}) \cdot (\nabla_y \widetilde{u}) \circ \mathbf{F} \\ & + \sum_{m,n} \left(D\mathbf{F}\,\alpha\,(x)\,D\mathbf{F}^T\right)_{m,n} \frac{\partial^2 \widetilde{u}}{\partial y_m \partial y_n} \circ \mathbf{F},\end{aligned}$$

where $\nabla_x \cdot \alpha \nabla_x$ is applied to each component of $\mathbf{F}$. Observe that the coefficient is represented by the matrix $\nabla\mathbf{F}\,\alpha\,(x)\,\nabla\mathbf{F}^T$, which is outside the differential operator. Furthermore, $\nabla_x \cdot \nabla_x \mathbf{F} = 0$ leaving only the term involving the second order differential operator. It has been shown that this non-divergence form of the elliptic problem in 2D has unique solution $\widetilde{u} \in \mathcal{H}^2(\Omega)$ provided α satisfies the condition of uniform ellipticity (Maugeri et al., 2000, Proposition 1.5.1). Thus, harmonic coordinates transform the problem in such a way that, even with $L^\infty(\Omega)$ coefficient functions, the solution gains a degree of regularity.

An important consequence of this representation is that the standard error estimates for piecewise linear finite elements can optimally approximate $\widetilde{u}$. Notice that the definition of u in terms of $\widetilde{u}$ and invertibility of $\mathbf{F}$ reveal that $u \circ \mathbf{F}^{-1} \in \mathcal{H}^2(\Omega)$. Therefore one may write an approximation of u in terms of nodal linear functions as

$$u_h \circ \mathbf{F}^{-1}(x) = \sum_{j \in \mathcal{N}} U_j \, \phi_j(x)$$

where ϕ_j are the nodal basis functions of $P_h^1(\Omega)$ and $\mathcal{N}$ are the nodes of $\mathcal{T}$. A direct result is

$$u_h(x) = \sum_{j \in \mathcal{N}} U_j \, \phi_j \circ \mathbf{F}(x)$$

implying that a suitable approximation space for u involves composite basis functions. Owhadi and Zhang (2006) show that one can indeed replace the standard space of linear functions P_h^1 with

$$X_h^1 = \left\{ v \circ \mathbf{F} : \text{ for all } K \in \mathcal{T},\, v|_K \in \mathbb{P}_1(K) \right\}.$$

Further, they prove that for $u_h \in X_h^1$ the optimal order estimate

$$\|u - u_h\|_{\mathcal{H}^1(\Omega)} = \mathcal{O}(h),$$

holds provided the ratio of the largest and smallest eigenvalues of the new coefficient matrix $D\mathbf{F}\alpha\,(x)\,D\mathbf{F}^T$ is bounded.

The application of this method is calculating the solution to the model problem on a coarse mesh $\mathcal{T}$ of Ω. However, the fine-scale basis functions in X_h^1 will require an accurate representation of the harmonic map. Thus, the harmonic map auxiliary problem is computed by the standard finite element method on a high-resolution mesh $\mathcal{E}$ of the domain Ω. The choice of boundary condition in Eq. (2.12) means that $\mathbf{F}$ does not change the domain boundary.

This composition rule, using a piecewise linear approximation of $\mathbf{F}$, leads to basis functions that are piecewise linear at the fine mesh level. Given a fine mesh element $W \in \mathcal{E}$, the approximate harmonic map $\mathbf{F}$ will produce a new triangle $\mathbf{F}(W)$ by mapping the vertices. Essentially, the approximate harmonic map is an affine transformation from Ω to Ω. However, these mapped elements provide the input for the standard piecewise linear basis functions ϕ described in X_h^1. Thus, the composite basis functions are actually defined on a mapped version of the fine mesh. This means the composite basis functions are define on a triangular fine mesh that is not the same as the mesh used to compute $\mathbf{F}$. Furthermore, the coefficient functions are defined on the same mesh as $\mathbf{F}$.

To avoid this inconvenience Owhadi and Zhang (2006) propose an approximation space where the basis functions are localized to each element K

$$Z_h = \{v : \text{ for all } K \in \mathcal{T},\, \phi \in \mathbb{P}_1(\mathbf{F}(K)),\, v|_K = \phi \circ \mathbf{F}|_K\},$$

where $\mathbf{F}(K)$ are triangles formed by mapping only the vertices of coarse elements K to harmonic coordinates and the functions ϕ are linear on the mapped element. Note that the composite basis functions are restricted to the coarse elements $K \in \mathcal{T}$. This induces a non-conformity which adds an additional component of error to the approximation.

These alternative basis functions are more attractive from the standpoint of approximating because the fine mesh $\mathcal{E}$ is common to all components of the scheme. The method presented in the next chapter removes the non-conforming error by accurately representing the basis support.

Chapter 3

Harmonic Coordinate Finite Element Implementation

In this chapter, I describe the harmonic coordinate finite element method (HCFEM) implementation for this thesis. The foundation of the HCFEM is the standard finite element method. Thus, I begin by presenting the basic FEM discretization of the model problem. An algorithm for the the standard FEM assembly completes that discussion.

With the standard FEM algorithm defined, I then describe the construction of the high resolution basis functions. At this point I introduce the harmonic map subproblem that provides the fine scale information. This fine scale information is then incorporated into the basis through composition. With this new basis described, I present one of the principle contributions of this thesis, namely the matrix assembly algorithms using mesh intersections to precisely represent this composite basis. The final section of this chapter is devoted to computing the error between the numerical and true solutions.

3.1 Weak Formulation of the Model Problem

The model problem for this thesis is

$$\begin{aligned} -\nabla \cdot (\alpha(x)\ \nabla u) + \beta(x)\, u = f(x)\,, &\qquad \text{in } \Omega, \\ u = g, &\qquad \text{on } \partial\Omega, \end{aligned} \tag{3.1}$$

where α and β are piecewise constant functions, the domain Ω is rectangular, and $f \in L^2(\Omega)$. In Chapter 2, I presented the weak formulation for problem Eq. (3.1): Find $u \in \mathcal{H}^1_E(\Omega)$ such that

$$-\int_\Omega \nabla\varphi \cdot (\alpha(x)\ \nabla u) + \int_\Omega \beta(x)\, u\, \varphi = \int_\Omega f\, \varphi \tag{3.2}$$

is true for all $\varphi \in \mathcal{H}^1_0(\Omega)$. In this chapter, the discussion turns from one of approximation properties, as in Chapter 2, to that of an explicit definition of the methods used to compute numerical solutions of the model problem. This weak formulation is the starting point for each of the finite element methods presented in this chapter. I will first present an implementation of the standard finite element method, which is used to provide a foundation for the HCFEM.

3.2 Standard Finite Element Discretization

While I described the approximation properties of piecewise linear finite elements in Chapter 2, I did not explain how the method is implemented. Any finite element implementation begins with a mesh or family of meshes. Let $\mathcal{T}_h$ denote a shape regular triangular mesh of the domain Ω. In this thesis, the mesh $\mathcal{T}_h$ is simply a triangulation of the domain Ω without regard to internal interfaces. I will described a mesh such as the one shown in FIGREF as an unfitted mesh.

Assume this representative mesh $\mathcal{T}_h$ is comprised of N_{el} triangular elements K with mesh size h as defined as in Eq. (2.5). A mesh of N_{el} triangular elements is fully described by a set of globally numbered vertices P and a connectivity matrix $C \in \mathbb{N}_0^{3\times N_{\text{el}}}$. With a unique numbering for the vertices P, each triangle $K \in \mathcal{T}_h$ can be described by three non-negative integers $a, b, c \in \mathbb{N}_0$ where $p_a^K, p_b^K, p_c^K \in P$ are the vertices of K. As presented here, the column of the connectivity array C associated with element K holds these three integers a, b, c providing a link between the element and its vertices. I will take it as a foregone conclusion that the vertices P and connectivity matrix C are provided by some mesh generator, as is generally the case.

The space of piecewise linear functions

$$V_h = \{v \in \mathcal{C}^0(\Omega) : \forall K \in \mathcal{T}_h, v|_K \in \mathbb{P}_1\} \tag{3.3}$$

was introduced in Chapter 2. In the definition, $\mathbb{P}_1$ is the class of linear functions. Thus, each member of V_h is a linear function within each element K. This space V_h is $\mathcal{H}^1(\Omega)$-conformal, meaning that any $v \in V_h$ is also a member of $\mathcal{H}^1(\Omega)$.

On any triangle K, a basis for $\mathbb{P}_1$ has the form

$$\phi_{j,K}(x) = a_0^{j,K} + a_1^{j,K} x_1 + a_2^{j,K} x_2, \quad j = 1, 2, 3 \tag{3.4}$$

where $x = (x_1, x_2)$ are the cartesian components of the point $x \in K$. The coefficients $a_k^{j,K}$ are computed by applying the nodal property $\phi_{j,K}(p_k) = \delta_{jk}$ at the vertices $\{p_k^K\}_{k=a,b,c}$ and solving the resulting 3×3 linear systems

$$\begin{bmatrix} 1 & p_{a,x_1}^K & p_{a,x_2}^K \\ 1 & p_{b,x_1}^K & p_{b,x_2}^K \\ 1 & p_{c,x_1}^K & p_{c,x_2}^K \end{bmatrix} \begin{bmatrix} a_0^{j,K} \\ a_1^{j,K} \\ a_2^{j,K} \end{bmatrix} = \begin{bmatrix} | \\ e_j \\ | \end{bmatrix} \tag{3.5}$$

where e_j is the unit vector with 1 in the jth position and zero in all others. Here the non-negative integers a, b, c are the vertex indices from C for element K, and $x = (x_1, x_2)$ denotes the cartesian component of the vertex.

Each of the functions $\phi_{j,K}$ for $j = 1, 2, 3$ are only nonzero within the associated element K. These three basis functions, determined by solving the linear systems described above, are shown on a representative element K in Fig. 3.1.

In terms of the basis functions defined above, the approximation $u_h \in V_h \cap \mathcal{H}_E^1(\Omega)$

to the true solution of the model problem is expressed as

$$u_h(x) = \sum_{K \in \mathcal{T}_h} \sum_{j=1}^{3} U_{C_{j,K}} \phi_{j,K}(x), \tag{3.6}$$

where U is a vector of unknown coefficients at the vertices of the mesh $\mathcal{T}_h$, and C is the connectivity array associated with $\mathcal{T}_h$. The object $V_h \cap \mathcal{H}^1_E(\Omega)$ indicates the class of piecewise linear functions satisfying the boundary conditions. Substituting this expression for u_h into the weak formulation Eq. (3.1) and replacing v_h with the basis functions from the test space V_h, we have the discrete weak formulation: Find $u_h \in V_h \cap \mathcal{H}^1_E(\Omega)$ such that

$$\sum_{K \in \mathcal{T}_h} \sum_{j=1}^{3} U_{C_{j,K}} \left(\int_K \nabla\phi_{i,K} \cdot \alpha(x) \cdot \nabla\phi_{j,K} + \int_K \beta(x)\, \phi_{i,K}\, \phi_{j,K} \right) = \sum_{K \in \mathcal{T}_h} \int_K f\, \phi_{i,K},$$

for all $\phi_{i,K} \in V_h \cap \mathcal{H}^1_0(\Omega)$.

Each integral in the discretized weak formulation above can be identified as the

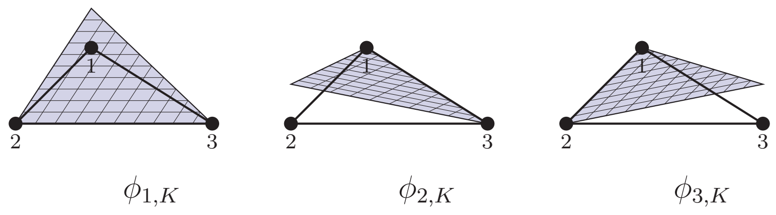


Figure 3.1: The $\mathbb{P}_1$ Lagrange basis functions.

usual finite element stiffness matrix $\mathcal{S}$

$$\mathcal{S}_{C_{i,K},C_{j,K}} \overset{\pm}{=} \int_K \nabla\phi_{i,K} \cdot \alpha\left(x\right) \cdot \nabla\phi_{j,K}\, dx, \tag{3.7}$$

and mass matrix $\mathcal{M}$

$$\mathcal{M}_{C_{i,K},C_{j,K}} \overset{\pm}{=} \int_K \beta\left(x\right)\, \phi_{i,K}\, \phi_{j,K}\, dx, \tag{3.8}$$

where the indices are written in terms of the connectivity array C. Entries in the mass and stiffness matrix may be visited multiple times when the values at $C_{i,K}$ and $C_{j,K}$ repeat. Therefore, I use the symbol $\overset{\pm}{=}$ to represent a cumulative, or running summation. The components of the load vector $\mathcal{F}$ are

$$\mathcal{F}_{C_{i,K}} = \int_K f\, \phi_{i,K}\, dx. \tag{3.9}$$

Therefore, the weak formulation reduces to a linear system to find the coefficient vector U in the expression

$$\sum_{K\in\mathcal{T}_h} \sum_{j=1}^{3} U_{C_{j,K}} \left(\mathcal{S}_{C_{i,K},C_{j,K}} + \mathcal{M}_{C_{i,K},C_{j,K}}\right) = \sum_{K\in\mathcal{T}_h} \mathcal{F}_{C_{i,K}},$$

or more compactly

$$\left(\mathcal{S} + \mathcal{M}\right) U = \mathcal{F}.$$

Choosing $V_h \cap \mathcal{H}_0^1(\Omega)$ restricts the basis functions $\phi_{i,K} = 0$ on the domain boundary and rows of $\mathcal{S}$ and $\mathcal{M}$ associated with boundary degrees of freedom U are all zeros. Likewise, data in the right-hand side $\mathcal{F}$ associated with the same degrees of freedom $U_{C_{i,K}}$ will have zero entries. Dirichlet boundary conditions are easily enforced under this construction by replacing the diagonal entries in $\mathcal{S} + \mathcal{M}$ with 1 and the zero right-hand side data by the boundary function g evaluated at the boundary nodes associated with the appropriate degrees of freedom in U. This one standard method of enforcing Dirichlet boundary conditions in the finite element method.

Evaluating the Standard FEM Integrals

A standard approach for evaluating the inner products is to apply a Gauss quadrature rule. Integrals are transformed to discrete summations where the integrand is evaluated at specially selected points in the domain of integration. Points are computed on a reference triangle $\hat{K}$ defined by the vertices $\hat{p} = \{(0,0),(1,0),(0,1)\}$. Denote the l_q quadrature points by $\{\hat{\xi}_l\}_{l=1}^{l_q}$ with the associated quadrature weights $\{\hat{w}_l\}_{l=1}^{l_q}$.

Since my implementation defines the basis functions on the physical element $K \in \mathcal{T}_h$, I must map the quadrature points to K in order to evaluate the integrals. Mapping the reference quadrature data requires a transformation function that takes points in the reference triangle $\hat{K}$ to elements K. Piecewise linear basis functions are computed on the reference element $\hat{K}$ exactly as they are for any other triangle. These reference basis functions associated with the vertices $\hat{p}$ are

$$\begin{aligned}\widehat{\phi}_1(\hat{\xi}_1, \hat{\xi}_2) &= 1 - \xi_1 - \xi_2,\\ \widehat{\phi}_2(\hat{\xi}_1, \hat{\xi}_2) &= \xi_1,\\ \widehat{\phi}_3(\hat{\xi}_1, \hat{\xi}_2) &= \xi_2,\end{aligned}$$

written in terms of reference coordinates $\hat{\xi} = (\hat{\xi}_1, \hat{\xi}_2)$. Note that these basis functions satisfy the nodal property $\widehat{\phi}_j(\hat{p}_i) = \delta_{ij}$.

A mapping T_K that takes points in $\hat{K}$ and maps them to an element K is easily constructed in terms of these reference basis functions $\widehat{\phi}_j$ by

$$T_K(\hat{\xi}) = \sum_{j=1}^{3} p_j \widehat{\phi}_j(\hat{\xi})$$

where $\hat{\xi}$ is a point in $\hat{K}$. Notice that the mapping has the property that each vertex $\hat{p}_j$ of the reference element $\hat{K}$ is mapped to a unique vertex of the physical element K. Thus, $T_K(\hat{p}_j) = p_j$ for all $j = 1, 2, 3$.

Given the quadrature points $\{\hat{\xi}_l\}_{l=1}^{l_q}$, the mapping T_K produces a set of points $\{\xi_l\}_{l=1}^{l_q}$ in the physical element K. The basis functions $\phi_{j,K}$ and their gradients are evaluated at these transformed points.

Like the quadrature points, the weights $\hat{w}_l$ must be transformed since they are associated with the area of the reference triangle. This transformation is accomplished

Algorithm 3.1 Global Matrix Assembly for Standard FEM

```
Given $\mathcal{T}_h,\ C \in \mathbb{N}_0^{3\times N_{\text{el}}}$
Let $\mathcal{M} = 0,\ \mathcal{S} = 0,\ \mathcal{F} = 0$
**for** $K \in \mathcal{T}_h$ **do**
  **for** $1 \leq ni \leq 3$ **do**;   $i = C_{ni,K}$
    **for** $1 \leq nj \leq 3$ **do**;   $j = C_{nj,K}$
      **for** $1 \leq l \leq l_q$ **do**
        *Map Gauss quadrature points from $\widehat{K}$ to $K$*
        $\xi_l = T_K(\hat{\xi}_l)$
        *Map Gauss quadrature weights*
        $w_l = \hat{w}_l \det DT_K$
        *Evaluate stiffness and mass matrix integrals*
        $\mathcal{S}_{i,j} = \mathcal{S}_{i,j} + \nabla\phi_{ni,K}(\xi_l) \cdot \alpha\,(\xi_l) \cdot \nabla\phi_{nj,K}(\xi_l)\ w_l$
        $\mathcal{M}_{i,j} = \mathcal{M}_{i,j} + \beta(\xi_l)\,\phi_{ni,K}(\xi_l)\ \phi_{nj,K}(\xi_l)\,w_l$
      **end for**
      *Evaluate right-hand side*
      $\mathcal{F}_i = \mathcal{F}_i + f(\xi_l)\,\phi_{i,K}(\xi_l)\,w_l$
    **end for**
  **end for**
**end for**
```

by applying the determinant of the Jacobian matrix DT_K

$$(DT_K)_{i,j} = \frac{\partial T_i}{\partial x_j}$$

which is a constant 2×2 matrix for each triangular element. The weights are individually transformed by $\hat{w}_l \det DT_K = w_l$.

Using these transformed quadrature points ξ_l and weights w_l, the stiffness matrix inner products are evaluated as

$$\begin{aligned}\mathcal{S}_{C_{i,K},C_{j,K}} &\overset{+}{=} \int_K \nabla\phi_{i,K} \cdot \alpha\,(x) \cdot \nabla\phi_{j,K}\,dx,\\ &\overset{+}{\approx} \sum_{l=1}^{l_q} \nabla\phi_{i,K}(\xi_l) \cdot \alpha(\xi_l) \cdot \nabla\phi_{j,K}(\xi_l)\,w_l,\end{aligned}$$

where $\overset{+}{\approx}$ indicates that the quadrature introduces some approximation error. The mass matrix integral is transformed in exactly the same manner. This finite element assembly procedure is shown in Algorithm 3.1.

Alternative Implementations

Although it is often convenient to construct the finite element basis on a reference element, I have chosen above to compute the piecewise linear basis on each element directly. The reason for this divergence is the composition rule is most easily implemented in the absence of the reference map and I wish to maintain the same notation for the basis functions throughout the rest of this thesis. One can look to Ern and Guermond (2004) for a detailed presentation of the finite element algorithm in more traditional form.

3.3 Harmonic Coordinate FEM

In this section, I present two harmonic coordinate finite element methods (HCFEM). The fundamental aspect of the HCFEM approach is the harmonic coordinate transform. To described the methods, I begin by introducing a piecewise linear approximation to the harmonic map. Using this approximation I then describe two methods of constructing composite basis functions. The first approach is the localized Galerkin method that inspired this work (Owhadi and Zhang, 2006). From these non-conforming basis functions, I then explain how to construct a conforming basis. The remainder of the section is devoted to the development of matrix assembly algorithms using both the non-conforming and conforming HCFEM bases.

3.3.1 Approximate Harmonic Coordinates

I begin this section by briefly summarizing the salient details from the discussion of harmonic maps in Chapter 2. The harmonic coordinates subproblem associated with the model problem Eq. (3.1) is

$$\nabla \cdot (\alpha(x)\, \nabla F_i) = 0 \quad \text{in } \Omega \tag{3.10}$$

$$F_i = x_i \quad \text{on } \partial\Omega, \tag{3.11}$$

where α is the same as in Eq. (3.1) and $F_i : \Omega \to \mathbb{R}$ for each $i = 1, 2$. These problems, which yield the components of the harmonic map, are well-posed when the model problem Eq. (3.1) is well-posed. Therefore, the solutions F_1, F_2 exist and are unique.

I will denote the harmonic map as a vector-valued function $\mathbf{F}(x_1, x_2) = (F_1, F_2)$

where it is understood that F_1 and F_2 take the same arguments as $\mathbf{F}$. When it is clear from the context, I will use the more compact notation $x = (x_1, x_2)$ as I have in previous sections.

An important property, discussed in Chapter 2, is the map $\mathbf{F}$ composed of these components F_1 and F_2 defines a coordinate transformation. The Dirichlet boundary conditions for the harmonic map have special significance in that the components do not change the location of points on the boundary $\partial\Omega$ in their respective directions. More explicitly, when given a point $p \in \partial\Omega$, the harmonic map components act on the components of (p_1, p_2) such that $F_1(p_1, p_2) = p_1$ and $F_2(p_1, p_2) = p_2$. That is, $\mathbf{F}(p) = p$ on the boundary. Thus, the harmonic coordinate transformation only modifies the interior of the problem domain.

Since finding an exact solution in closed form is unlikely, the solutions F_1, F_2 are approximated using piecewise linear finite elements. The details can easily be summarized using the notation from the previous section as follows.

The approximation of the harmonic coordinate transform takes place on a triangular mesh $\mathcal{E}_{h'}$ of Ω that will generally satisfy two properties in this thesis. First, the mesh $\mathcal{E}_{h'}$ usually has a size $h' \ll h$ where h is the size of the unfitted mesh $\mathcal{T}_h$. This ensures the harmonic map will sufficiently resolve fine-scale structures in the domain. Secondly, elements in $\mathcal{E}_{h'}$ must align with internal interfaces where the coefficient functions exhibit discontinuities.

Of course, neither of these conditions is absolute. The condition on resolution might be satisfied with a grid size h' very close to h for some problems. Likewise, the second requirement could be relaxed in the case of material properties that vary randomly at scales which are too small to approximate practically. Under those circumstances, one might use a harmonic average within affected elements to obtain an effective constant material property at the finer scale. This will not be an issue in this thesis, where coefficient functions are strictly piecewise constant. Regardless, the triangulation $\mathcal{E}_{h'}$ cannot truly be described as a fine mesh. Instead, I refer to the fitted finite element mesh $\mathcal{E}_{h'}$ associated with the harmonic map $\mathbf{F}$ on Ω as the *auxiliary mesh*. A complementary description for the unfitted mesh $\mathcal{T}_h$ associated with the model problem is *primary mesh* (rather than coarse mesh). These definitions help prevent a confusing situation in cases where the auxiliary mesh has roughly the same element size as the primary mesh. Throughout the rest of the thesis I will drop the subscript h and h' on these meshes in favor of writing $\mathcal{T}$ as any primary mesh of Ω and $\mathcal{E}$ as any auxiliary mesh of the same domain.

Suppose the mesh $\mathcal{E}$ has an associated connectivity array $Q \in \mathbb{N}_0^{3\times N_{\text{el}}}$. Then the piecewise linear approximation to each component of the harmonic map is written as

$$F_1(x) = \sum_{W\in\mathcal{E}} \sum_{j=1}^{3} \hat{F}^1_{Q_{j,W}} \phi_{j,W}(x),$$

$$F_2(x) = \sum_{W\in\mathcal{E}} \sum_{j=1}^{3} \hat{F}^2_{Q_{j,W}} \phi_{j,W}(x),$$

where the arrays $\hat{F}^i$ for $i = 1, 2$ are the solution vectors from solving problems

Eq. (3.10) by the finite element method. These piecewise linear representations have exactly the same form as the approximation to the model problem in Eq. (3.6). Combining the solution vectors so that

$$\hat{\mathbf{F}} = \begin{bmatrix} \hat{F}_1^1 & \hat{F}_2^1 & \hat{F}_3^1 & \cdots \\ \hat{F}_1^2 & \hat{F}_2^2 & \hat{F}_3^2 & \cdots \end{bmatrix} \tag{3.12}$$

the piecewise linear approximation of the harmonic map is

$$\mathbf{F}(x) = \sum_{W \in \mathcal{E}} \sum_{j=1}^{3} \hat{\mathbf{F}}_{Q_{j,W}}\, \phi_{j,W}(x)\,, \tag{3.13}$$

where the subscript $Q_{j,W}$ is an index accessing a column of the solution matrix $\mathbf{F}$ whose rows are the solutions $\hat{F}^1$ and $\hat{F}^2$. The basis functions $\phi_{j,W}$ have the same form as Eq. (3.4) and are computed on each element $W \in \mathcal{E}$ by solving a similar linear system to Eq. (3.5).

Given that $\mathbf{F}$ is a coordinate transform, using it as such in any calculus expressions (chain rule, integral change of variable) brings about the need for the Jacobian matrix associated with $\mathbf{F}$. By definition, the Jacobian matrix of $\mathbf{F}$ on element $W \in \mathcal{E}$ is

$$D\mathbf{F}^W = \begin{bmatrix} \dfrac{\partial F_1|_W}{\partial x_1} & \dfrac{\partial F_1|_W}{\partial x_2} \\ \dfrac{\partial F_2|_W}{\partial x_1} & \dfrac{\partial F_2|_W}{\partial x_2} \end{bmatrix},$$

where F_1, F_2 are the scalar-valued components of $\mathbf{F}$, and $|_W$ denotes the restriction to element $W \in \mathcal{E}$.

Let the piecewise linear basis functions on element $W \in \mathcal{E}$ be

$$\phi_{j,W}(x) = a_0^{j,W} + a_1^{j,W} x_1 + a_2^{j,W} x_2.$$

Then, the Jacobian matrix associated with $\mathbf{F}$ on element W is

$$(D\mathbf{F}_W)_{i,j} = \sum_{n=1}^{2} \widehat{F}_n^{i,W} a_n^{j,W}$$

where $i, j = 1, 2$ and the matrix is referenced as $D\mathbf{F}_W$. Notice that the terms in the matrix depend only on the constants $a_n^{j,W}$ and $\widehat{F}_n^{i,W}$ for $i, j = 1, 2$ and $n = 1, 2$. That the Jacobian matrix is constant on each element $W \in \mathcal{E}$ is a consequence of the piecewise linear representation of $\mathbf{F}$. With this fact, the determinant of the Jacobian matrix, sometimes called the Jacobian,

$$J_{\mathbf{F},W} = \det D\mathbf{F}_W$$

is also constant on each element $W \in \mathcal{E}$.

Properties of the approximate harmonic map described here are extremely important and deserve to be summarized more succinctly. The harmonic map $\mathbf{F}$

- is component-wise, piecewise linear,
- does not change domain boundary,
- has element-wise constant Jacobian matrices, and

- its mesh $\mathcal{E}$ conforms to internal boundaries.

3.3.2 HCFEM Basis Functions

I will discuss two methods of constructing basis functions using the fine scale information contained in the harmonic map $\mathbf{F}$. I first describe the localized Galerkin basis as presented by Owhadi and Zhang (2006), which is the non-conforming basis discussed in Chapter 2. I then present a modification to their construction that yields a conforming finite element basis. The application of each basis centers around the element-wise stiffness integrals. Let it be understood that all references to the harmonic map refer to its piecewise linear approximation.

Localized Galerkin Method

Before defining the basis functions, some preliminary constructs are needed. Suppose a triangular element $K \in \mathcal{T}$ has vertices p_1, p_2, p_3. The gradient of any scalar-valued function $v : \mathbb{R}^2 \to \mathbb{R}$ can be approximated on a triangular element K by the so-called coarse gradient defined as

$$\nabla^K v = \begin{bmatrix} (p_2 - p_1)^T \\ (p_3 - p_1)^T \end{bmatrix}^{-1} \begin{bmatrix} v(p_2) - v(p_1) \\ v(p_3) - v(p_1) \end{bmatrix}, \tag{3.14}$$

where the notation $(p_2 - p_1)^T$ represents a row vector. Though this representation is not commonly seen in the literature, it is nonetheless equivalent to the gradient of a piecewise linear approximation of v on a triangular element K. One can consult Botsch et al. (2010, pg. 44) for an alternative form of the coarse gradient.

Since a coordinate transform does not change the function values of v, Owhadi and Zhang (2006) exploit Eq. (3.14) along with this fact to write the coarse gradient of v in terms of harmonic coordinates, or $\mathbf{F}$-coordinates,

$$\nabla^K_{\mathbf{F}} v = \begin{bmatrix} (\mathbf{F}(p_2) - \mathbf{F}(p_1))^T \\ (\mathbf{F}(p_3) - \mathbf{F}(p_1))^T \end{bmatrix}^{-1} \begin{bmatrix} v(p_2) - v(p_1) \\ v(p_3) - v(p_1) \end{bmatrix}. \tag{3.15}$$

Here, each vertex p_1, p_2, p_3 of the triangle K is represented in $\mathbf{F}$-coordinates as $\mathbf{F}(p_1)$, $\mathbf{F}(p_2)$, $\mathbf{F}(p_3)$ forming a new triangle $\mathbf{F}(K)$.

Using the above coarse gradient definitions, I will show how one can arrive at the basis functions defined by Owhadi and Zhang (2006). Define the nodal basis functions on element K by the conditions

$$\begin{cases} \xi_{j,K}(p_i) = \delta_{ij}, & i, j = 1, 2, 3, \\ \nabla_{\mathbf{F}} \xi^K_j = \text{constant}, & \text{on } K \text{ for all } j = 1, 2, 3. \end{cases}$$

The operator $\nabla_{\mathbf{F}}$ is the usual gradient with respect to the $\mathbf{F}$-coordinates and should not be confused with the coarse gradient operator $\nabla^K_{\mathbf{F}}$. Consider the basis function $\xi_{1,K}$ associated with vertex p_1. The gradient condition implies

$$\xi_{1,K}(x) = A + \mathbf{F}(x)^T g_1,$$

where A is a real constant and g_1 is the coarse gradient of $\xi_{1,K}$ with respect to $\mathbf{F}$-

coordinates. An application of the nodal property with definition Eq. (3.15) gives

$$\begin{aligned} g_1 &= \begin{bmatrix} (\mathbf{F}(p_2) - \mathbf{F}(p_1))^T \\ (\mathbf{F}(p_3) - \mathbf{F}(x_1))^T \end{bmatrix}^{-1} \begin{bmatrix} \xi_{1,K}(p_2) - \xi_{1,K}(p_1) \\ \xi_{1,K}(p_3) - \xi_{1,K}(p_1) \end{bmatrix} \\ &= \begin{bmatrix} (\mathbf{F}(p_2) - \mathbf{F}(p_1))^T \\ (\mathbf{F}(p_3) - \mathbf{F}(p_1))^T \end{bmatrix}^{-1} \begin{bmatrix} -1 \\ -1 \end{bmatrix} \end{aligned}$$

since $\xi_{1,K}(p_1) = 1$ and $\xi_{1,K}(p_2) = \xi_{1,K}(p_3) = 0$. A second application of the nodal property gives $A + \mathbf{F}(p_1)^T g_1 = 1$ so that

$$\xi_{1,K}(p_1) = 1 + (\mathbf{F}(x) - \mathbf{F}(p_1))^T g_1.$$

Repeating this procedure for $j = 2, 3$ reveals that the coarse gradients g_j are equivalent to the coarse gradients $\nabla^K_{\mathbf{F}} \phi_{j,K}$, where $\phi_{j,K}$ are the usual piecewise linear basis functions on K defined in Eq. (3.4). Therefore, the localized Galerkin basis functions on each element K are

$$\xi_{j,K}(x) = \begin{cases} 1 + (\mathbf{F}(x) - \mathbf{F}(p_j))^T \nabla^K_{\mathbf{F}} \phi_{j,K}, & \text{if } p_j \sim K, x \in K, \\ 0, & \text{otherwise,} \end{cases}$$

where the notation $p_j \sim K$ means that p_j is a vertex of K. These are the nonconforming basis functions described by Owhadi and Zhang (2006).

Implementing this basis in the discrete weak form is accomplished by replacing the usual finite element basis functions $\phi_{j,K}$ in Eq. (3.7) with the localized Galerkin basis as

$$\int_K \nabla\phi_{i,K}\,\alpha\,(x)\;\nabla\phi_{j,K} \to \int_K \nabla\xi_{i,K}\,\alpha\,(x)\;\nabla\xi_{j,K}(x)$$

Since both sets of basis functions are nodal, the degrees of freedom on the mesh $\mathcal{T}$ do not change. That is, the elements of the solution vector U for the discrete model problem remain associated with the same nodes of $\mathcal{T}$.

Notice that the basis functions $\xi_{j,K}$ are composite functions. Applying the chain rule, the gradient of the localized Galerkin basis is

$$\nabla\xi_{j,K} = D\mathbf{F}^T\,\nabla^K_{\mathbf{F}}\phi_{j,K}$$

where $D\mathbf{F}$ is the Jacobian matrix of $\mathbf{F}$. Then

$$\int_K \nabla\xi_{i,K}\,\alpha\,(x)\;\nabla\xi_{j,K}(x) = \int_K D\mathbf{F}^T\,\nabla^K_{\mathbf{F}}\phi_{i,K}(x)\,\alpha\,(x)\;D\mathbf{F}^T\,\nabla^K_{\mathbf{F}}\phi_{j,K}(x)$$

Since $\mathbf{F}$ is piecewise linear on $\mathcal{E}$, recall that $D\mathbf{F}$ is constant on each element $W \in \mathcal{E}$. Furthermore, the object $\nabla^K_{\mathbf{F}}\phi_{j,K}$ is also constant, but on each element $K \in \mathcal{T}$. By construction, the coefficient function α is constant on elements of the auxiliary mesh $\mathcal{E}$. These observations imply that all quantities in the integrand are constant when evaluated in an element $W \in \mathcal{E}$.

Since the primary and auxiliary meshes represent the same domain, a covering for

K in terms of the elements $W \in \mathcal{E}$ can be obtained through a mesh-element intersection. The strength of this method is derived from the fact that $\mathcal{E}$ is a fitted mesh of the domain. This means interfaces within the element K are resolved. However, there is no reason to expect the edges or vertices of triangles in $\mathcal{T}$ to align with those in $\mathcal{E}$. Thus, the intersection object $K \cap \mathcal{E}$ is made up of polygonal pieces corresponding to triangles in $\mathcal{E}$ but not necessarily triangles themselves.

Computing the mesh-element intersection $K \cap \mathcal{E}$, the element K can be represented by a collection of polygonal cells. The intersection of K with $\mathcal{E}$ is the set of cells described by

$$K \cap \mathcal{E} = \left\{ \pi : \pi = K \cap W,\ \forall W \in \mathcal{E},\ \text{with area}\,(\pi) > 0 \right\}.$$

Here area (π) denotes the area of polygonal cell π, which is zero if π is not closed (for instance, a point or line). This collection can be comprised of triangles, quadrilaterals, pentagons, and hexagons depending upon how individual triangular elements intersect. Call the element W the parent element of the polygon π. Nothing about the parent element changes, and the related polygonal cell π inherits all the properties of the parent (basis functions, Jacobian matrix).

Using the mesh-element intersection and constant quantities above, the stiffness integral is now written in terms of the collection of polygonal cells as

$$\int_K D\mathbf{F}^T \, \nabla_{\mathbf{F}}^K \phi_{i,K}(x) \, \alpha\,(x) \; D\mathbf{F}^T \, \nabla_{\mathbf{F}}^K \phi_{j,K}(x)$$
$$= \sum_{\pi \in K \cap \mathcal{E}} \mathrm{area}\,(\pi) \, D\mathbf{F}_W^T \, \nabla_{\mathbf{F}}^K \phi_{i,K} \, \alpha\,(\pi) \; D\mathbf{F}_W^T \, \nabla_{\mathbf{F}}^K \phi_{j,K},$$

where $\alpha\,(\pi)$ is the value of the coefficient function on the polygonal cell π which is part of element an $W \in \mathcal{E}$. Observe that this summation is not an approximation. This method removes the possibility of error due to computing the integrals by quadrature. Thus, the accuracy is primarily affected by how well the piecewise linear approximation to the harmonic map is resolved and the level of non-conformity exhibited by localized Galerkin basis. This implementation by mesh-element intersections is not found in the literature, so I believe it is novel.

Conforming Harmonic Coordinate FEM

The localized Galerkin basis functions can be expressed as a composition rule

$$\xi_{j,K}(x) = \psi_{j,K}(x)|_{x \in K} = \theta_{j,\mathbf{F}(K)} \circ \mathbf{F}(x)|_{x \in K}$$

where $\theta_{j,\mathbf{F}(K)}$ is a piecewise linear basis computed on the mapped element $\mathbf{F}(K)$ and $|_{x \in K}$ denotes the restriction to element $K \in \mathcal{T}$. I will discuss first why this restriction to K creates a non-conforming basis. This will lead to a method of construction for the conforming harmonic coordinate basis.

Recall that $\mathbf{F}(K)$ is the triangle formed by mapping the vertices of K to $\mathbf{F}$-coordinates (harmonic coordinates). Let $y = (y_1, y_2)$ denote a point in these new coordinates. The piecewise linear basis functions $\theta_{j,\mathbf{F}(K)}$ on $\mathbf{F}(K)$ have the form

$$\theta_{j,\mathbf{F}(K)} = a_j^{0,K} + a_j^{1,K} y_1 + a_j^{2,K} y_2 \tag{3.16}$$

where the coefficients $a_j^{n,K}$ for $n = 0, 1, 2$ and $j = 1, 2, 3$ are determined by applying the nodal property and solving a system of equations as in Eq. (3.5).

The support of $\theta_{j,\mathbf{F}(K)}$ is the triangle $\mathbf{F}(K)$. The question is: Where did the values $y \in \mathbf{F}(K)$ originate? That is, where are the values $x = \mathbf{F}^{-1}(y)$ for all $y \in \mathbf{F}(K)$? When $\mathbf{F}$ is the identity map $\mathbf{F}(x) = x$ for all $x \in \Omega$, the answer is $y = x$ and the elements are unchanged. However, there is little hope that any interesting problem will have such a simple associated harmonic map. Therefore, the pre-image of points $y \in \mathbf{F}(K)$ could be significantly distorted away from K. Therefore, the answer for general harmonic maps is the points $y \in \mathbf{F}(K)$ may originate from a non-triangular element in the domain Ω.

We can examine this effect in two ways. Suppose the harmonic map applied to K produces the distorted object as in Fig. 3.2. Here, the triangle $\mathbf{F}(K)$ is superimposed on the distorted element. In harmonic coordinates, the straight-sided element K is no longer a triangle. Pictorially, the figure shows the composition rule with restriction to K. Evaluate $\mathbf{F}$ at some point $x \in K$ to obtain a point y in harmonic coordinates. However, this y is not necessarily inside $\mathbf{F}(K)$, as can be seen in the figure. Thus, the

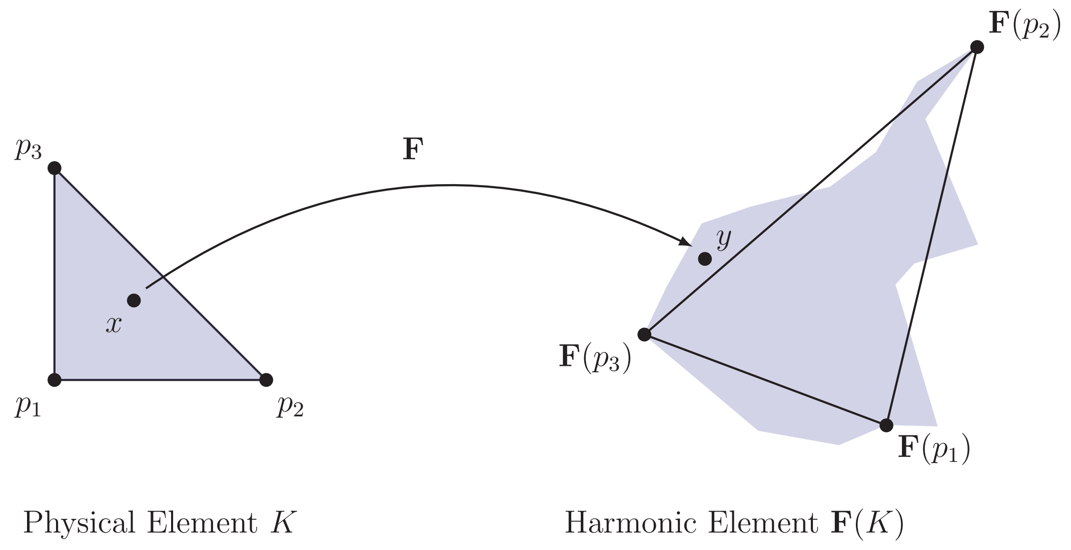


Figure 3.2: Illustration of mapping an element K to harmonic coordinates. The shaded triangular region in K is mapped to the distorted region with vertices $\mathbf{F}(p_j)$ for $j = 1, 2, 3$. A point $x \in K$ is mapped to a point $y = \mathbf{F}(x)$ outside the triangular region $\mathbf{F}(K)$.

new point may be outside the true support of $\theta_{j,\mathbf{F}(K)}$. In order to enforce the restriction to K, one must evaluate the functions $\theta_{j,\mathbf{F}(K)}$ at points outside the support. This is the source of the non-conformity in the localized Galerkin basis.

Alternatively, the inverse harmonic map applied to the triangular element $\mathbf{F}(K)$ produces a distorted shape instead of the original element K. This is shown in Fig. 3.3, where the shaded region of $\mathbf{F}(K)$ is mapped under $\mathbf{F}^{-1}$ to produce the distorted region superimposed on K. A point $y \in \mathbf{F}(K)$ may be mapped to a point $x \in \Omega$ outside K. This also shows that some points $x \in K$ may not have an image in the triangular element $\mathbf{F}(K)$ where the functions $\theta_{j,\mathbf{F}(K)}$ are defined. Yet this distorted, shaded region shown in Fig. 3.3 represents the true support of the composite basis functions, and the issue now is how this support should be computed.

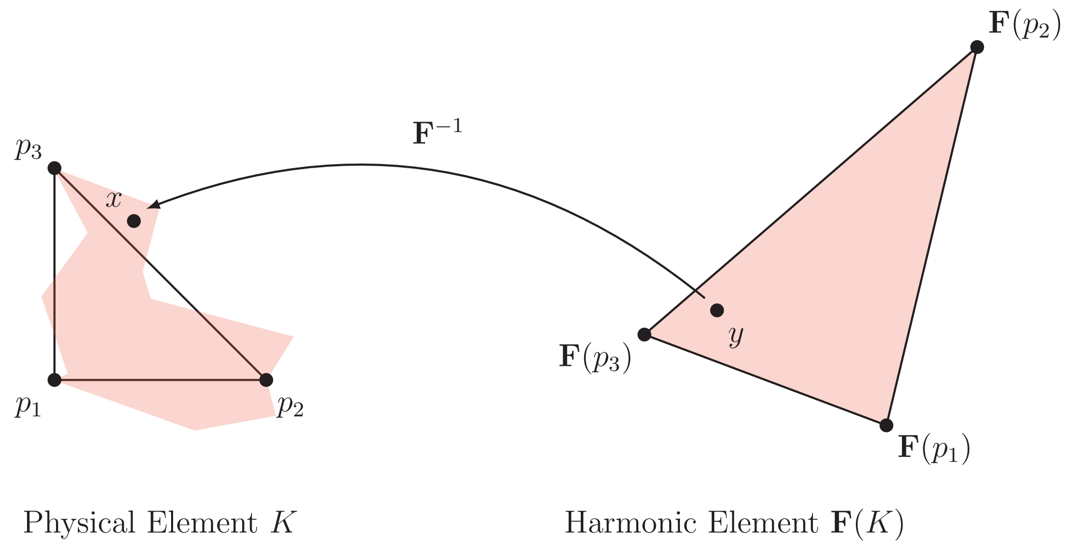


Figure 3.3: Illustration of mapping an element $\mathbf{F}(K)$ to physical coordinates. The shaded triangular region in $\mathbf{F}(K)$ is mapped to the distorted region with vertices p_j for $j = 1, 2, 3$. A point $y \in \mathbf{F}(K)$ is mapped to a point $x = \mathbf{F}^{-1}(y)$ which is outside the triangular region K.

Instead of restricting the composition to K and introducing a non-conformity, the actual support of the composite basis can be computed using the same intersection technique I applied in the previous section. The price to pay is the support of $\psi_{j,K}$ is no longer localized and triangular. However, this will not be a problem since $\theta_{j,\mathbf{F}(K)}$ is supported on the triangular element $\mathbf{F}(K)$. Instead of computing the mesh-element intersections in the physical domain, I compute the intersections in the harmonic domain.

Let $\widetilde{K}$ denote the distorted support for $\psi_{j,K}$. A representation of $\widetilde{K}$ is obtained indirectly by the following procedure. Map the mesh $\mathcal{E}$ to harmonic coordinates to create the mesh $\mathbf{F}(\mathcal{E})$. The vertices of this mapped mesh are nothing more than the values $\hat{\mathbf{F}}$ described in Eq. (3.12). In harmonic coordinates, the mesh-element intersection $\mathbf{F}(K) \cap \mathbf{F}(\mathcal{E})$ is the set of polygonal cells

$$\mathbf{F}(K) \cap \mathbf{F}(\mathcal{E}) = \{\pi : \pi = \mathbf{F}(K) \cap \mathbf{F}(W),\ \forall W \in \mathcal{E},\ \text{area}\,(\pi) > 0\}\,.$$

Notice this collection of polygons has the same form as in the previous section. The distorted support is now simply the inverse harmonic map applied to the mesh-element intersection

$$\widetilde{K} = \mathbf{F}^{-1}(\mathbf{F}(K) \cap \mathbf{F}(\mathcal{E})).$$

These basis functions are implemented in the stiffness integrals by the replacement

$$\int_K \nabla\phi_{i,K}\alpha\,(x)\,\nabla\phi_{j,K} \to \int_{\widetilde{K}} \nabla\psi_{i,K}\alpha\,(x)\,\nabla\psi_{j,K}$$

where integration now takes place over the distorted support $\widetilde{K}$. Since the basis functions $\psi_{j,K}$ still satisfy the nodal property on the vertices of the element K, the degrees of freedom are associated with the same nodes of K.

Applying the chain rule, the gradient of the composite basis functions $\psi_{j,K}$ for $j = 1, 2, 3$ is

$$\begin{aligned}\nabla\psi_{j,K} &= \nabla(\theta_{j,\mathbf{F}(K)} \circ \mathbf{F}(x)) \\ &= D\mathbf{F}^T\,\nabla_{\mathbf{F}}\theta_{j,\mathbf{F}(K)} \circ \mathbf{F}(x)\end{aligned}$$

Substituting this expression into the integral yields

$$\int_{\widetilde{K}} \nabla\psi_{i,K}\alpha\,(x)\,\nabla\psi_{j,K} = \int_{\widetilde{K}} D\mathbf{F}^T\,\nabla_{\mathbf{F}}\theta_{i,\mathbf{F}(K)}\circ\mathbf{F}(x)\,\alpha\,(x)\;D\mathbf{F}^T\,\nabla_{\mathbf{F}}\theta_{j,\mathbf{F}(K)}\circ\mathbf{F}(x)$$

With the definition of $\widetilde{K}$ in terms of the intersection object, a change of variable to harmonic coordinates gives the stiffness integral

$$\int\limits_{\mathbf{F}^{-1}(\mathbf{F}(K)\cap\mathbf{F}(\mathcal{E})} D\mathbf{F}^T\,\nabla_{\mathbf{F}}\theta_{i,\mathbf{F}(K)}\circ\mathbf{F}(x)\,\alpha\,(x)\;D\mathbf{F}^T\,\nabla_{\mathbf{F}}\theta_{j,\mathbf{F}(K)}\circ\mathbf{F}(x) =$$
$$\int\limits_{\mathbf{F}(K)\cap\mathbf{F}(\mathcal{E})} D\mathbf{F}^T\,\nabla_{\mathbf{F}}\theta_{i,\mathbf{F}(K)}(y)\,\alpha\circ\mathbf{F}^{-1}(y)\,D\mathbf{F}^T\,\nabla_{\mathbf{F}}\theta_{j,\mathbf{F}(K)}(y)\,\det D\mathbf{F}^{-1}$$

The integral above is computed by exploiting the constant terms over the polygon representation $\mathbf{F}(K)\cap\mathbf{F}(\mathcal{E})$.

First, note that $\nabla_{\mathbf{F}}\theta_{j,\mathbf{F}(K)}$ is constant on $\mathbf{F}(K)$. This implies that $\nabla_{\mathbf{F}}\theta_{j,\mathbf{F}(K)}$ is constant over all polygons in $\mathbf{F}(K)\cap\mathbf{F}(\mathcal{E})$. Next, the coefficient α is constant on elements $W\in\mathcal{E}$, so the coefficient function is also constant on elements $\mathbf{F}(W)$. By construction, the Jacobian matrices of $\mathbf{F}$ are constant on elements $W\in\mathcal{E}$, which means they too are constant on the mapped elements $\mathbf{F}(W)$. Therefore, all terms in the integrand are constant when evaluated on a polygonal cell $\pi\in\mathbf{F}(K)\cap\mathbf{F}(\mathcal{E})$. This means the integral now has the form

$$\int\limits_{\mathbf{F}(K)\cap\mathbf{F}(\mathcal{E})}\text{integrand} = \sum_{\pi\in\mathbf{F}(K)\cap\mathbf{F}(\mathcal{E})}\int_{\pi}\text{constant terms.}$$

Thus, the stiffness integral is transformed into the summation

$$\int\limits_{\mathbf{F}(K)\cap\mathbf{F}(\mathcal{E})} D\mathbf{F}^T\,\nabla_{\mathbf{F}}\theta_{i,\mathbf{F}(K)}(y)\,\alpha\circ\mathbf{F}^{-1}(y)\,D\mathbf{F}^T\,\nabla_{\mathbf{F}}\theta_{j,\mathbf{F}(K)}(y)\,\det D\mathbf{F}^{-1} =$$

$$\sum_{\pi\in\mathbf{F}(K)\cap\mathbf{F}(\mathcal{E})} \operatorname{area}(\pi)\,D\mathbf{F}_W^T\,\nabla\theta_{i,\mathbf{F}(K)}\,\alpha(\pi)\,D\mathbf{F}_W^T\,\nabla\theta_{j,\mathbf{F}(K)}\,J_{\mathbf{F},W}^{-1}.$$

3.3.3 Global Stiffness Matrix Assembly

Both the localized Galerkin and conforming HCFEM bases were implemented in previous section using the context of the stiffness matrix. Using those results, the stiffness matrices are computed as

$$\mathcal{S}_{C_{i,K},C_{j,K}} = \sum_{\pi\in K\cap\mathcal{E}} \operatorname{area}(\pi)\,D\mathbf{F}_W^T\,\nabla_{\mathbf{F}}^K\phi_{i,K}\,\alpha(\pi)\;D\mathbf{F}^W\,\nabla_{\mathbf{F}}^K\phi_{j,K}$$

for the localized Galerkin method of Owhadi and Zhang (2006) and

$$\mathcal{S}_{C_{i,K},C_{j,K}} = \sum_{\pi\in\mathbf{F}(K)\cap\mathbf{F}(\mathcal{E})} \operatorname{area}(\pi)\,D\mathbf{F}_W^T\,\nabla\theta_{i,\mathbf{F}(K)}\,\alpha(\pi)\,D\mathbf{F}_W^T\,\nabla\theta_{j,\mathbf{F}(K)}\,J_{\mathbf{F},W}^{-1}$$

for the new HCFEM. Thus, the stiffness matrix is assembled by visiting each element and computing the summations associated with that element. Here the process is fundamentally the same as Algorithm 3.1 for the standard FEM. However, the application of Gauss quadrature is replaced with a summation over polygonal cells from the intersection object. These stiffness assembly algorithms are summarized in

Algorithm 3.2 and Algorithm 3.3.

At the polygon cell level, all of the quantities are constant. Certainly the mesh intersections should be computed beforehand. Additionally, the Jacobian matrices of the harmonic map on the auxiliary mesh can be computed in advance. Precomputing these items means the assembly algorithms are very inexpensive. In fact, the primary cost of these implementations is almost entirely related to computing the intersections.

Algorithm 3.2 Localized Galerkin Global Stiffness Assembly

```
1: Given $\mathcal{T}$, $C$, $\mathcal{E}$, and harmonic map $\mathbf{F}$
2: Compute $D\mathbf{F}_W$ and $J_{\mathbf{F},W}$ on each $W \in \mathcal{E}$
3: Compute gradients $\nabla\phi_{i,K}$ on each $K \in \mathcal{T}$
4: Compute coarse gradients $\nabla_{\mathbf{F}}^{K}\phi_{i,K}$
5: $\mathcal{S} = 0$
6: for $K \in \mathcal{T}$ do
7:   Compute intersection $\mathcal{I} = K \cap \mathcal{E}$
8:   for $1 \leq ni \leq n_{sh}$ do;   $i = C_{ni,K}$
9:     for $1 \leq nj \leq n_{sh}$ do;   $j = C_{nj,K}$
10:      $s = 0$
11:      for $\pi \in \mathcal{I}$ do
12:        Get $D\mathbf{F}_W$ from parent element $W \in \mathcal{E}$ of $\pi$
13:        $s \overset{+}{=} \text{area}(\pi)\ D\mathbf{F}_W^T \nabla_{\mathbf{F}}^{K}\phi_{i,K} \cdot \alpha(\pi)\, D\mathbf{F}_W^T \nabla_{\mathbf{F}}^{K}\phi_{j,K}$
14:      end for
15:      $\mathcal{S}(i,j) = \mathcal{S}(i,j) + s$
16:    end for
17:  end for
18: end for
```

3.3.4 Global Mass Matrix Assembly

As seen in the previous section, computing the inner products that make up the stiffness matrix entries is straight-forward since the integrals over the primary mesh

Algorithm 3.3 Conforming HCFEM Global Stiffness Assembly

```
 1: Given $\mathcal{T}$, $C$, $\mathcal{E}$, and harmonic map $\mathbf{F}$
 2: Construct mapped meshes $\mathbf{F}(\mathcal{T})$ and $\mathbf{F}(\mathcal{E})$
 3: Compute $D\mathbf{F}_W$ and $J_{\mathbf{F},W}$ on each $W \in \mathcal{E}$
 4: Compute $\nabla\theta_{i,\mathbf{F}(K)}$ on each $\mathbf{F}(K) \in \mathbf{F}(\mathcal{T})$
 5: $\mathcal{S} = 0$
 6: for $\mathbf{F}(K) \in \mathbf{F}(\mathcal{T})$ do
 7:   Compute intersection $\mathcal{I} = \mathbf{F}(K) \cap \mathbf{F}(\mathcal{E})$
 8:   for $1 \le ni \le n_{sh}$ do;   $i = C_{ni,K}$
 9:     for $1 \le nj \le n_{sh}$ do;   $j = C_{nj,K}$
10:       $s = 0$
11:       for $\pi \in \mathcal{I}$ do
12:         Get $J_{\mathbf{F},W}$, $D\mathbf{F}_W$ from parent element $W \in \mathcal{E}$ of $\pi$
13:         $s \overset{\pm}{=} \text{area}(\pi)\ J^{-1}_{\mathbf{F},W}\, D\mathbf{F}^T_W \nabla\theta_{i,\mathbf{F}(K)} \cdot \alpha(\pi)\, D\mathbf{F}^T_W \nabla\theta_{j,\mathbf{F}(K)}$
14:       end for
15:       $\mathcal{S}(i,j) = \mathcal{S}(i,j) + s$
16:     end for
17:   end for
18: end for
```

elements reduce to sums of constant terms. However, computing the contributions to the mass matrix requires integration of the basis functions on each piece of polygonal area arising from the intersections. One could further subdivide individual cells more complicated than a triangle and apply a simple quadrature on the newly refined polygon. This subdivision would produce triangles and allow a direct application of an appropriate quadrature algorithm. Since no quadrature nodes are used to compute the HCFEM stiffness integrals, I choose compute the mass matrix integrals directly on these polygonal areas without subdivision by applying Green's theorem. As with the HCFEM stiffness integrals, any inaccuracy will be due to the precision of the harmonic map. I will consider the elemental mass matrix integrals for each basis and show that their evaluation ultimately amounts to computing line integrals that are identical in form.

Implementing the localized Galerkin basis for mass integrals involves the simple replacement of the standard FEM basis in Eq. (3.8)

$$\int_K \beta(x)\ \phi_{i,K}\,\phi_{j,K} \to \int_K \beta(x)\ \xi_{i,K}\,\xi_{j,K}.$$

This integral can be written in terms of the mesh intersection object $K \cap \mathcal{E}$ as

$$\int_K \beta(x)\ \xi_{i,K}\,\xi_{j,K} = \sum_{\pi \in K \cap \mathcal{E}} \beta(\pi) \int_\pi \xi_{i,K}\,\xi_{j,K},$$

where I have used the fact that the coefficient function β is constant on elements of the auxiliary mesh. Recall that these basis functions are defined by a composition rule involving the harmonic map $\mathbf{F}$. Since the harmonic map is a piecewise linear function on $\mathcal{E}$, the composition with the linear functions $\theta_{j,\mathbf{F}(K)}$ results in a piecewise linear function $\xi_{j,K}$ on K. On the polygonal cells π, the localized Galerkin basis is a linear function. Hence, the integrand is the product of two linear functions. Therefore, mass matrix entries involving the localized Galerkin basis amount to integrating products of linear functions over polygons.

For clarity, I will explicitly write the localized Galerkin basis function associated with a polygon π. Let W be the parent element of a polygon π from the mesh-element intersection $K \cap \mathcal{E}$. The localized Galerkin basis functions on the parent element W can be written in terms of the coarse gradient and the components of the harmonic map as

$$\begin{aligned}\xi_{j,K}(x) &= 1 + (\mathbf{F}(x) - \mathbf{F}(p_j))^T \, \nabla^K_{\mathbf{F}} \phi_{j,K} \\ &= s_0 + s_1 \, F_1(x) + s_2 \, F_2(x),\end{aligned}$$

where s_0, s_1, and s_2 are constants. Here

$$\begin{aligned}s_0 &= 1 - \left[\nabla^K_{\mathbf{F}} \phi_{j,K}\right]_1 F_1(p_j) - \left[\nabla^K_{\mathbf{F}} \phi_{j,K}\right]_2 F_2(p_j) \\ s_1 &= \left[\nabla^K_{\mathbf{F}} \phi_{j,K}\right]_1, \\ s_2 &= \left[\nabla^K_{\mathbf{F}} \phi_{j,K}\right]_2,\end{aligned}$$

where $\left[\nabla^K_{\mathbf{F}} \phi_{j,K}\right]_i$ is the ith component of the coarse gradient, which is constant on K. Using the piecewise linear representation of the harmonic map, the localized Galerkin basis functions on the polygon π are

$$\xi_{j,K}(x) = s_0 + \sum_{i=1}^{3} (s_1 \hat{F}^1_{Q_{i,W}} + s_2 \hat{F}^2_{Q_{i,W}}) \phi_{i,W}(x), \tag{3.17}$$

where the coefficient arrays $\hat{F}^1$ and $\hat{F}^2$ are the solution vectors for the components of the harmonic map defined in Section 3.3.1, Q is the connectivity array for $\mathcal{E}$, and $j = 1, 2, 3$. Since the functions $\phi_{i,W}$ are linear on the parent element W, the localized Galerkin basis function are linear on the polygon π. Thus, the integrands are simply products of linear functions on polygons.

Using the conforming HCFEM basis in the mass matrix integrals involves the replacement

$$\int_K \beta\,(x)\;\phi_{i,K}\,\phi_{j,K} \rightarrow \int_{\widetilde{K}} \beta\,(x)\;\psi_{i,K}\,\psi_{j,K},$$

where integration now takes place over the distorted support $\widetilde{K}$. Recall that basis functions $\psi_{j,K}$ arise from the composition rule

$$\psi_{j,K}(x) = \theta_{j,\mathbf{F}(K)} \circ \mathbf{F}(x).$$

The distorted support associated with K for this basis is defined in terms of the mesh-element intersection $\mathbf{F}(K) \cap \mathbf{F}(\mathcal{E})$ as

$$\widetilde{K} = \mathbf{F}^{-1}(\mathbf{F}(K) \cap \mathbf{F}(\mathcal{E})).$$

Using this fact and a change of variable to harmonic coordinates, the mass matrix integral becomes

$$\int_{\widetilde{K}} \beta\,(x)\;\psi_{i,K}\,\psi_{j,K} = \sum_{\pi \in \mathbf{F}(K) \cap \mathbf{F}(\mathcal{E})} \beta(\pi)\,J^{-1}_{\mathbf{F},W} \int_{\pi} \theta_{i,\mathbf{F}(K)}\,\theta_{j,\mathbf{F}(K)}$$

On the element $\mathbf{F}(K)$, the basis functions $\theta_{j,\mathbf{F}(K)}$ are linear functions. Therefore, computing the mass matrix integrals for the conforming HCFEM basis also reduces to integrating the product of linear functions over polygons.

Since these integrals are essentially identical, I will focus on integrating the product $\theta_{i,\mathbf{F}(K)}\,\theta_{j,\mathbf{F}(K)}$ over some polygon π without loss of generality. Substituting the definition of $\theta_{j,\mathbf{F}(K)}$ from Eq. (3.16) into the integrand, the inner product becomes

$$\int_\pi \theta_{i,\mathbf{F}(K)}\,\theta_{j,\mathbf{F}(K)} = \int_\pi \left(a_0^{i,K} + a_1^{i,K} y_1 + a_2^{i,K} y_2\right)\left(a_0^{j,K} + a_1^{j,K} y_1 + a_2^{j,K} y_2\right)\,dy_1 dy_2$$

Collecting like terms and applying the linearity of the integral operator, this integral expands to

$$\begin{aligned}\int_\pi \theta_{i,\mathbf{F}(K)}\,\theta_{j,\mathbf{F}(K)} &= \int_\pi \left(a_0^{i,K} + a_1^{i,K} y_1 + a_2^{i,K} y_2\right)\left(a_0^{j,K} + a_1^{j,K} y_1 + a_2^{j,K} y_2\right)\,dy_1 dy_2 \\ &= a_0^{i,K} a_0^{j,K} \operatorname{area}(\pi) + \left(a_0^{i,K} a_1^{j,K} + a_1^{i,K} a_0^{j,K}\right) \int_\pi y_1\,dy_1 dy_2 \\ &\quad + \left(a_0^{i,K} a_2^{j,K} + a_2^{i,K} a_0^{j,K}\right) \int_\pi y_2\,dy_1 dy_2 \\ &\quad + \left(a_1^{i,K} a_2^{j,K} + a_2^{i,K} a_1^{j,K}\right) \int_\pi y_1 y_2\,dy_1 dy_2 \\ &\quad + a_1^{i,K} a_1^{j,K} \int_\pi y_1^2\,dy_1 dy_2 \\ &\quad + a_2^{i,K} a_2^{j,K} \int_\pi y_2^2\,dy_1 dy_2\end{aligned} \tag{3.18}$$

The polygons π are always simple, closed polygons with at most six sides (hexagons). Thus, a method is needed to compute integrals of the form

$$\int_\pi y_1^\mu\, y_2^\nu\, dy_1\, dy_2 \tag{3.19}$$

where $\mu, \nu \in \{0, 1, 2\}$ and π is a simple, closed polygon.

Recall that Green's theorem in 2D (Anton, 1988, pg. 1145) provides the conditions for converting an integral over an area to an integral over the boundary of that area. Applying Green's theorem, the integral Eq. (3.19) of the monomial $y_1^\mu y_2^\nu$ over a polygon π arising from the intersection can be written as

$$\int_\pi y_1^\mu y_2^\nu \, dy_1 dy_2 = \int_{\partial\pi} \frac{y_1^{\mu+1}}{\mu+1} y_2^\nu \, dy_2.$$

Let the vertices of π be the set of points (r_i, s_i) for $i = 1, \ldots, N$, with $(r_{N+1}, s_{N+1}) = (r_1, s_1)$. With the vertices defined this way, Cattani and Paoluzzi (1990) show that this boundary integral has the closed form

$$\int_{\partial\pi} \frac{y_1^{\mu+1}}{\mu+1} y_2^\nu \, dy_2 = \frac{1}{\mu+1} \sum_{i=1}^{N} \sum_{n=0}^{\mu+1} \sum_{m=0}^{\nu} r_i^{\mu+1-n} R_i^n s_i^{\nu-m} S_i^{m+1} \binom{\mu+1}{n} \binom{\nu}{m} \frac{1}{m+n+1}$$

where $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ is the binomial coefficient, and the quantities $R_i = r_{i+1} - r_i$ and $S_i = s_{i+1} - s_i$ are the lengths of the projections of each edge onto the coordinate axes. This result means that the mass matrix integrations using mesh-element intersections is exact for the given basis functions and piecewise constant coefficient functions. Global matrix assembly routines are shown in Algorithm 3.4 (localized Galerkin) and Algorithm 3.5 (conforming HCFEM).

Algorithm 3.4 Localized Galerkin Global Mass Matrix Assembly

```
1: Given $\mathcal{T}$, $C$, $\mathcal{E}$, $Q$, and harmonic map $\mathbf{F}$
2: Compute gradients $\nabla\phi_{i,K}$ on each $K \in \mathcal{T}$
3: Compute coarse gradients $\nabla_{\mathbf{F}}^{K}\phi_{i,K}$
4: $\mathcal{M} = 0$
5: for $K \in \mathcal{T}$ do
6:   Compute intersection $\mathcal{I} = K \cap \mathcal{E}$
7:   for $1 \leq ni \leq 3$ do;   $i = C_{ni,K}$
8:     for $1 \leq nj \leq 3$ do;   $j = C_{nj,K}$
9:       $m = 0$
10:      for $\pi \in \mathcal{I}$ do
11:        Compute integral using closed form
12:        Use basis defined by Eq. (3.17)
13:        $m \overset{\pm}{=} \beta(\pi) \int_{\pi} \xi_{ni,K}\, \xi_{nj,K}$
14:      end for
15:      $\mathcal{M}(i,j) = \mathcal{M}(i,j) + m$
16:    end for
17:  end for
18: end for
```

Algorithm 3.5 Conforming HCFEM Mass Matrix Assembly

```
1: Given $\mathcal{T}$, $C$, $\mathcal{E}$, $Q$, and harmonic map $\mathbf{F}$
2: Construct the mapped meshes $\mathbf{F}(\mathcal{T})$ and $\mathbf{F}(\mathcal{E})$
3: Compute $J_{\mathbf{F},W}$ on each $W \in \mathcal{E}$
4: Compute $\theta_{j,\mathbf{F}(K)}$ on each $\mathbf{F}(K) \in \mathbf{F}(\mathcal{T})$
5: $\mathcal{M} = 0$
6: for $\mathbf{F}(K) \in \mathbf{F}(\mathcal{T})$ do
7:   Compute intersection $\mathcal{I} = \mathbf{F}(K) \cap \mathbf{F}(\mathcal{E})$
8:   for $1 \leq ni \leq 3$ do;   $i = C_{ni,K}$
9:     for $1 \leq nj \leq 3$ do;   $j = C_{nj,K}$
10:      $m = 0$
11:      for $\pi \in \mathcal{I}$ do
12:        Compute integral using closed form
13:        Get $J_{\mathbf{F},W}$ from parent element $W \in \mathcal{E}$ of $\pi$
14:        $m \overset{\pm}{=} \beta(\pi)\, J_{\mathbf{F},W}^{-1} \int_{\pi} \theta_{ni,\mathbf{F}(K)}\, \theta_{nj,\mathbf{F}(K)}$
15:      end for
16:      $\mathcal{M}(i,j) = \mathcal{M}(i,j) + m$
17:    end for
18:  end for
19: end for
```

3.3.5 Load Vector Assembly

Implementations of the standard FEM often use the mass matrix to compute the load vector. Briefly, suppose f is approximated in the basis on the mesh $\mathcal{T}$ by

$$f\left(x\right) \approx \sum_{K\in\mathcal{T}} \sum_{j=1}^{3} \hat{f}_{C_{j,K}}\, \phi_{j,K}(x),$$

where $\hat{f}_{C_{j,K}} = f(P_{C_{j,K}})$ and P is the set of vertices of $\mathcal{T}$. Substituting this expression for f into the inner product Eq. (3.9) for the components of the load vector gives

$$\begin{aligned}\mathcal{F}_{C_{i,K}} &\overset{\pm}{=} \int_K f\left(x\right) \phi_{i,K} \\ &= \sum_{j=1}^{3} \hat{f}_{C_{j,K}} \int_K \phi_{i,K}\, \phi_{j,K}.\end{aligned}$$

Notice that the integral is simply the definition of the mass matrix with $\beta = 1$. The observation is that this expression is equivalent to evaluating the load function f at the vertices of $\mathcal{T}$ and multiplying that vector by the mass matrix computed with $\beta = 1$. However, the high-resolution HCFEM basis functions (non-conforming or conforming) do not necessarily provide a good approximation to f by this method.

Given that the HCFEM basis functions are actually piecewise linear on $\mathcal{E}$, a more suitable approximation for the load function would be the piecewise linear approximation on the fitted auxiliary mesh $\mathcal{E}$. With the auxiliary mesh $\mathcal{E}$ and connectivity array Q, the piecewise linear approximation of f is given by

$$f(x) \approx \sum_{W \in \mathcal{E}} \sum_{j=1}^{3} \hat{f}_{Q_{j,W}} \, \phi_{j,W}(x), \tag{3.20}$$

where $\hat{f}_{Q_{j,W}} = f(P_{Q_{j,W}})$ and P is the set of vertices of $\mathcal{E}$.

This approximation for f is used to construct the load vector with the localized Galerkin basis as follows. Consider an element $K \in \mathcal{T}$ and its intersection with the auxiliary mesh $K \cap \mathcal{E}$. The contribution to the load vector from this element is

$$\begin{aligned} \int_K f(x)\, \xi_{i,K} &= \int\limits_{K \cap \mathcal{E}} f(x)\, \xi_{i,K} \\ &= \sum_{\pi \in K \cap \mathcal{E}} \int_\pi f(x)\, \xi_{i,K} \end{aligned}$$

Substituting this approximation of f on the parent element W of polygon π gives

$$\int_\pi f(x)\, \xi_{i,K} = \sum_{j=1}^{3} \hat{f}_{Q_{j,W}} \int_\pi \phi_{j,W}\, \xi_{i,K}.$$

Now the integration method used for the HCFEM mass matrices is applicable since the functions $\phi_{j,W}$ and $\xi_{i,K}$ are piecewise linear on the parent element $W \in \mathcal{E}$ of π. This load vector assembly process is summarized in Algorithm 3.6.

The same approach can be applied to load vector integrals involving the conforming HCFEM basis. Here the integral domain for the in the load vector calculation

Algorithm 3.6 Localized Galerkin Global Load Vector Assembly

```
1: Given $\mathcal{T}$, $C$, $\mathcal{E}$, $Q$, and harmonic map $\mathbf{F}$
2: Compute basis functions $\phi_{i,W}$ on each $W \in \mathcal{E}$
3: Interpolate $f$ at vertices of $\mathcal{E}$, call it $\hat{f}$
4: $\mathcal{F} = 0$
5: for $K \in \mathcal{T}$ do
6:   Compute intersection $\mathcal{I} = K \cap \mathcal{E}$
7:   for $1 \leq ni \leq 3$ do;   $i = C_{ni,K}$
8:     $x = 0$
9:     for $\pi \in \mathcal{I}$ do
10:      for $1 \leq nj \leq 3$ do;   $j = Q_{nj,W}$
11:        Compute integral using closed form
12:        Use basis defined by Eq. (3.17)
13:        $x \overset{+}{=} \hat{f}_{Q_{nj,W}} \int_\pi \phi_{nj,W}\, \xi_{ni,K}$
14:      end for
15:    end for
16:    $\mathcal{F}(i) = \mathcal{F}(i) + x$
17:  end for
18: end for
```

Eq. (3.9) is replaced with integration over the distorted support $\widetilde{K}$ as

$$\int_K f(x)\,\phi_{i,K} \to \int_{\widetilde{K}} f(x)\,\psi_{i,K}.$$

Writing the distorted support in terms of the mesh-element intersection in harmonic coordinates gives

$$\int_{\widetilde{K}} f(x)\,\psi_{i,K} = \int\limits_{\mathbf{F}^{-1}(\mathbf{F}(K)\cap\mathbf{F}(\mathcal{E}))} f(x)\;\psi_{i,K}$$

Applying the definition of the composite basis functions $\psi_{i,K} = \theta_{i,\mathbf{F}(K)} \circ \mathbf{F}$ and transforming the integral to harmonic coordinates yields

$$\int_{\widetilde{K}} f\,(x)\,\psi_{i,K} = \int\limits_{\mathbf{F}(K)\cap\mathbf{F}(\mathcal{E})} f \circ \mathbf{F}^{-1}\,\theta_{i,\mathbf{F}(K)}\ \det D\mathbf{F}^{-1}$$

for the integrals associated with element K. With this representation, I can write the integral in terms of individual polygons $\pi \in \mathbf{F}(K) \cap \mathbf{F}(\mathcal{E})$.

Since the Jacobian of the harmonic map $\mathbf{F}$ is constant on each polygonal area π, a property which is inherited from the parent element $W \in \mathcal{E}$, the above integral becomes

$$\int\limits_{\mathbf{F}(K)\cap\mathbf{F}(\mathcal{E})} f \circ \mathbf{F}^{-1}\,\theta_{i,\mathbf{F}(K)}\ \det D\mathbf{F}^{-1} = \sum_{\pi\in\mathbf{F}(K)\cap\mathbf{F}(\mathcal{E})} J^{-1}_{\mathbf{F},W} \int_{\pi} f \circ \mathbf{F}^{-1}\,\theta_{i,\mathbf{F}(K)}$$

The basis functions $\theta_{i,\mathbf{F}(K)}$ are linear on all polygons π which leaves only the term $f \circ \mathbf{F}^{-1}$ to consider.

The piecewise linear nature of $\mathbf{F}$ on $\mathcal{E}$ has special consequences that can be exploited to write an approximation to $f \circ \mathbf{F}^{-1}$ by starting with Eq. (3.20). Forming the composition of Eq. (3.20) with $\mathbf{F}^{-1}$ gives

$$f \circ \mathbf{F}^{-1}(y) \approx \sum_{W\in\mathcal{E}} \sum_{j=1}^{3} \hat{f}_{Q_{j,W}}\,\phi_{j,W} \circ \mathbf{F}^{-1}(y).$$

Because $\mathbf{F}$ is piecewise linear on W, one can construct basis functions on the mapped element $\mathbf{F}(W)$ and relate them directly to the functions $\phi_{j,W}$ simply using $\mathbf{F}$ as an affine map. Let $\theta_{j,\mathbf{F}(W)}$ be the linear basis functions formed directly on $\mathbf{F}(W) \in \mathbf{F}(\mathcal{E})$. Since these functions satisfy the nodal property on $\mathbf{F}(W)$, and $\mathbf{F}$ is an invertible affine map (coordinate transform), $\theta_{j,\mathbf{F}(W)}$ are related to the basis functions on $W \in \mathcal{E}$ by

$$\phi_{j,W}(x) = \theta_{j,\mathbf{F}(W)} \circ \mathbf{F}(x).$$

Therefore, the approximation to $f \circ \mathbf{F}^{-1}$ is

$$f \circ \mathbf{F}^{-1}(y) \approx \sum_{W \in \mathcal{E}} \sum_{j=1}^{3} \hat{f}_{Q_{j,W}}\, \theta_{j,\mathbf{F}(W)}(y). \tag{3.21}$$

Obviously, the transformation is not necessary since the basis functions $\theta_{j,\mathbf{F}(W)}$ can be computed directly on the mapped elements $\mathbf{F}(W)$ for each $W \in \mathcal{E}$. Substituting the approximation Eq. (3.21) into the integral over a polygon π gives the approximate contribution to the load vector

$$\int_{\pi} f \circ \mathbf{F}^{-1}\, \theta_{i,\mathbf{F}(K)} \approx J^{-1}_{\mathbf{F},W} \sum_{j=1}^{3} \hat{f}_{Q_{j,W}} \int_{\pi} \theta_{j,\mathbf{F}(W)}\, \theta_{i,\mathbf{F}(K)}$$

The method of exact integration over polygons can now be used to compute the integrals exactly because both $\theta_{j,\mathbf{F}(W)}$ and $\theta_{i,\mathbf{F}(K)}$ are linear on π. This assembly process is summarized in Algorithm 3.7.

Algorithm 3.7 Conforming HCFEM Load Vector Assembly

```
1: Given $\mathcal{T}$, $C$, $\mathcal{E}$, $Q$, and harmonic map $\mathbf{F}$
2: Construct mapped meshes $\mathbf{F}(\mathcal{T})$ and $\mathbf{F}(\mathcal{E})$
3: Compute basis functions $\theta_{i,\mathbf{F}(K)}$ on each $\mathbf{F}(K) \in \mathbf{F}(\mathcal{T})$
4: Compute basis functions $\theta_{i,\mathbf{F}(W)}$ on each $\mathbf{F}(W) \in \mathbf{F}(\mathcal{E})$
5: Interpolate $f$ at vertices of $\mathcal{E}$, call it $\hat{f}$
6: $\mathcal{F} = 0$
7: for $K \in \mathcal{T}$ do
8:   Compute intersection $\mathcal{I} = \mathbf{F}(K) \cap \mathbf{F}(\mathcal{E})$
9:   for $1 \leq ni \leq 3$ do;   $i = C_{ni,K}$
10:    $x = 0$
11:    for $\pi \in \mathcal{I}$ do
12:      for $1 \leq nj \leq 3$ do;   $j = Q_{nj,W}$
13:        Compute integral using closed form, $J_{\mathbf{F},W}$ from parent element of $\pi$
14:        $x \overset{+}{=} \hat{f}_{Q_{nj,W}} \, J^{-1}_{\mathbf{F},W} \int_\pi \theta_{nj,\mathbf{F}(W)} \, \theta_{ni,\mathbf{F}(K)}$
15:      end for
16:    end for
17:    $\mathcal{F}(i) = \mathcal{F}(i) + x$
18:  end for
19: end for
```

3.4 Error Calculations

I have chosen to examine the relative error between the numerical and reference solutions in the $\ell^\infty(\Omega)$, $\ell^2(\Omega)$, and $L^2(\Omega)$ norms. The size of the solution in the same norm can be reported as a relative error

$$RE = \frac{\|u - u_h\|}{\|u\|},$$

which relates a percentage difference between the expected and computed solutions. Let it be understood that the reference solution u is either an analytical solution or a highly resolved numerical solution computed by the finite element using a fitted mesh of the domain Ω. Likewise, the numerical solution u_h is given by the solution vector U which holds the nodal approximations of u on the vertices of $\mathcal{T}$.

Suppose $\mathcal{T}$ has N_p nodes P. Then the $\ell^\infty(\Omega)$ error is simply

$$\|u - u_h\|_{\ell^\infty(\Omega)} = \max_{\substack{1 \le n \le N_p \\ p_n \in P}} |u(p_n) - u_h(p_n)|,$$

where $u_h(p_n)$ is the entry in the solution vector U corresponding to node p_n. I use the discrete infinity norm, also called the maximum norm, because this indicates the largest possible difference between the numerical approximation and the accepted or true solution at the vertices. Measuring the error this way reveals how well a method interpolates the solution at these vertices in the worst possible way. This provides a view of the nodal accuracy as if the mass and stiffness matrices are merely discrete operators constructed without knowledge of internal variations.

The $\ell^2(\Omega)$ error is computed by evaluating

$$\|u - u_h\|^2_{\ell^2(\Omega)} = \sum_{\substack{1 \le n \le N_p \\ p_n \in P}} |u(p_n) - u_h(p_n)|^2$$

where the values $u_h(p_n)$ are as described above. This norm is included because it is often used to measure the accuracy of finite difference methods.

Although the basis functions used to compute the solution U contain possibly fine scale information, it is interesting to look at the $L^2(\Omega)$ as if the numerical solution corresponds to the usual finite element basis on $\mathcal{T}$. Essentially, this measure of the error indicates how the use of these composite basis functions to construct mass and stiffness matrices improves the accuracy of an unfitted FEM. The numerical solution viewed as an unfitted FEM is

$$u_h(x) = \sum_{K \in \mathcal{T}} \sum_{j=1}^{3} U_{C_{j,K}} \phi_{j,K}(x), \tag{3.22}$$

where U is the solution vector obtained using the HCFEM basis functions. Thus, I am using the HCFEM solution with a standard FEM approximation. The $L^2(\Omega)$ error is

$$\|u - u_h\|_{L^2(\Omega)} = \sum_{K \in \mathcal{T}} \int_K (u - u_h)(u - u_h),$$

where u_h is the solution defined by Eq. (3.22). To compute the $L^2(\Omega)$ error, I use Gauss quadrature sufficient to integrate the standard piecewise linear basis functions on each element $K \in \mathcal{T}$.

Chapter 4

Software Implementation

A major component of this thesis project is the software implementation. I begin this chapter by discussing two existing software packages I used that were developed by third parties. The computational framework I established uses several `Trilinos` packages for the standard finite element assembly, basic matrix operations and linear solvers (Heroux and Willenbring, 2003). Unstructured mesh generation is provided by `Gmsh`, which is a powerful mesh generator with a versatile and user-friendly interface (Geuzaine and Remacle, 2009).

Beyond third-party software, I developed several classes that hide the complexity of the assembly algorithms described in the previous chapter. These classes are divided into four main groups: mesh, problem definition, assembly, and harmonic map. I begin by describing in detail my `tbTriMesh` C++ class whose primary unique feature is a concrete intersection algorithm for computing all the objects of the form

$K \cap \mathcal{E}$ mentioned in the previous chapter. To conclude the discussion of mesh objects, I introduce a wrapper class `tbMeshGen` for generating `tbTriMesh` objects using a geometry description. A complete definition of the model problem also needs the coefficient, boundary, and load functions. These functions are provided by the user based on the abstract class `tbMyProblem`. Since the assembly algorithms are provided in Chapter 3, the next focus is the design of a class for assembling the matrices and load vectors. Finally, I present a simple class for constructing a harmonic map and discuss its components.

With all these relatively independent classes, I show a simple example.

4.1 Third-Party Software

4.1.1 Trilinos

My software relies on the a number of packages from the `Trilinos` software project (Heroux et al., 2003a,b,c; Heroux and Willenbring, 2003; Sala et al., 2004). This is an extensive collection of software packages distributed under the Lesser Gnu Public License (LGPL) that provides a unified, object-oriented framework for the solution of large-scale physical problems. A chief goal of the `Trilinos` project is the integration of the many existing numerical solvers into a single, user-friendly development environment.

The most fundamental component of `Trilinos` is the linear algebra package

`Epetra`. `Epetra` is a set of linear algebra objects, such as distributed sparse matrices and vectors, that are accepted as input by all other `Trilinos` packages. Access to external dense and sparse linear solvers is provided by the interface classes in `Amesos`. This package unifies the many different interfaces used by solvers like Umfpack, Lapack and Scalapack under a single, object-oriented interface. I decided to use the sparse, serial solver `Amesos_KLU` that is provided with `Amesos` since my linear systems never exceeded the memory available on any of the computers available to me. All linear systems in this thesis are solved using `Amesos_KLU`.

For the standard finite element implementation described in Ch 3, I use the `Intrepid` package (Bochev et al., 2009). The `Intrepid` package is a set of tools specifically designed for computing the individual components of finite element matrices. I based my harmonic map solver implementation directly on an example from the `Intrepid` source code.

4.1.2 `Gmsh`

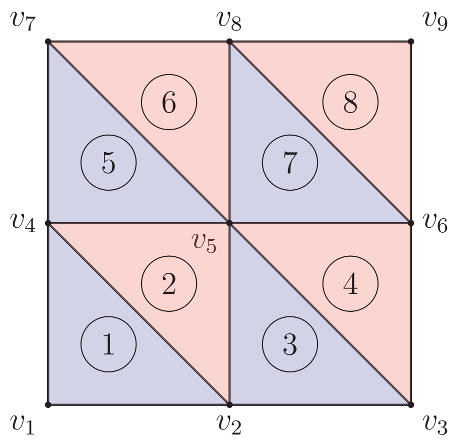
Structured triangular meshes of rectangular domains, such as those considered in this thesis, are trivial to construct. Although some generators are available for this purpose, the simplicity of such triangular meshes in this context allowed me to develop a robust and fast mesh generator. However, unstructured meshes, particularly those fitted to complex geometries, are much more difficult to generate. For such meshes, I use the software `Gmsh` (Geuzaine and Remacle, 2009).

`Gmsh` is an open-source 2D and 3D mesh generator. One particularly impressive and user-friendly aspect of `Gmsh` is its cross-platform, computer-aided drafting (CAD) interface, which works equally well on Linux, Mac OSX and Windows. Through this CAD interface, a user can specify a domain geometry and uniquely identify individual regions. A well-documented part of `Gmsh` is the syntax for model design. Models are defined through the GUI by primitive vector graphics components, such as points, lines, circular arcs, elliptical arcs, and Bezier curves. The internal, CAD software-dependent language of these models is hidden from the user to allow for extensibility to other CAD engines without the need for changing the model description. As a result, each component that can be drawn on the screen has a corresponding command that is written to a text file. It is this geometry specification file that I use to define model domains.

A lesser-known feature of `Gmsh` is the application programming interface (API) accessible by compiling the source code as a library.

4.2 A Simple Mesh Data Structure

A mesh is a collection triangular elements defined by a set of unique vertices. Each vertex is assigned a unique global identification number in the set of non-negative integers $\mathbb{N}_0$. These global indices are used to define the elements of the mesh through an element-vertex correspondence matrix $C \in \mathbb{N}_0^{3\times N_{el}}$ that defines how vertices are connected to form triangles. The array of vertices V and correspondence matrix C



$$C = \begin{bmatrix} 1 & 2 & 2 & 3 & 4 & 5 & 5 & 6 \\ 2 & 5 & 3 & 6 & 5 & 8 & 6 & 9 \\ 4 & 4 & 5 & 5 & 7 & 7 & 8 & 8 \end{bmatrix}$$

Element ①: [1 2 4]

Element ②: [2 5 4]

Element ③: [2 3 5]

. . . and so on. . .

Figure 4.1: An example of a simple mesh. The columns of the correspondence matrix C define the triangles in terms of the vertex global identification numbers, which are indicated by the subscript on v_j. Several examples of element connectivity are written explicitly. Circled numbers are the element global identification indices. Note the matrix C is not unique.

are the output from a mesh generator.

As a very simple example, consider the mesh in Fig. 4.1. Each of the eight elements is identified globally by the circled numbers. These global element identifiers also correspond to the columns of the correspondence matrix C in the figure. This example suggests a Mesh generators produce a correspondence matrix and vertex list This a very traditional and efficient method of storing mesh information.

These data are enough to suggest another possible data structure for a triangular mesh; in this case, the most primitive datum is a vertex and an element is made of vertices. Vertices are defined by their two real components and a global identifier together with a collection of operations as shown in the following C++ class.

```
class tbVertex {
public:
  tbVertex(double x, double y, long globalID, bool IsOnBndry);
  tbVertex operator+(const tbVertex&);
  tbVertex operator-(const tbVertex&);
  tbVertex operator=(const tbVertex&);
  double operator*(const tbVertex&); // dot product
  double distanceTo(const tbVertex&);
  bool IsOnBoundary();
  double x() const;
  double y() const;
  long globalID();
private:
  double _x;
  double _y;
  long _globalID;
  bool _IsOnBoundary;
}
```

Vertices are not allowed to change once they are defined. For this reason, the private data members for the coordinates, global identifier and boundary flag are accessible only through read-only methods.

Objects of this type are used in place of more primitive types, so the typical arithmetic operations are overloaded to allow direct addition, subtraction, scalar multiplication, and assignment. In addition to these basic operations, the method `distanceTo()` is provided to compute the Euclidean distance between two vertices.

With this approach, each triangular element is defined by its three vertices and a global identifier for the element. I prefer to keep the data encapsulated as much as possible, therefore basis function coefficients and important geometric quantities, such as area, are precomputed and stored within each triangle object. Importantly, the constructor for `tbTriangle` automatically computes these quantities. An important benefit of this this design is that unit testing can easily be performed on triangle objects in the absence of a mesh or even valid global identifiers.

```
class tbTriangle {
```

```
public:
  tbTriangle(const tbTriangle&);
  tbTriangle(const tbVertex&, const tbVertex&, const tbVertex&,
             long globalID, long regionTag);
  tbPWLBasis getBasis(int localVertexID);
  // ...
  // Various set/get methods for properties
  // ...
  std::vector<tbCell> Cell;
  tbMaterialProperty material;
private:
  std::vector<tbVertex> _vertex;
  long _globalID;

}
```

The element global identifier would normally indicate the column of the correspondence matrix C where the element vertices are found if that matrix were explicitly constructed. Although storing the matrix C and vertex data is more efficient, I have chosen to store data in this less efficient manner to

- facilitate a simplified intersection algorithm, and
- maintain a greater degree of data encapsulation at the element level.

Vertex data will be duplicated under this construct, but these duplicate data will only affect the amount of computer memory used.

With these basic vertex and element data structures, a mesh object is defined as

```
class tbTriMesh {
  public:
    // ... Constructors ...
    std::vector<tbTriangle> Element;
    // ... Methods ...
}
```

A mesh object is then, by design, simply a collection of elements and some set of functions (methods) that allow the user to modify mesh properties and perform operations on mesh data. The usage of `std::vector` containers simplifies memory allocation and clean-up as the objects are used and discarded.

4.3 Mesh Generator Class

One key design paradigm of this software is the internal data structure should be independent of the method used to generate the data. The case of mesh generation is no exception. Regardless of the method used to construct a mesh, the result is always a usable `tbTriMesh` object with all elements fully defined. Thus, the mesh generator class needs to know the means by which a mesh should be generated along with a definition of the domain that the underlying mesh generator will understand.

The mesh generator class is defined as

```
class tbMeshGen {
  public:
    tbMeshGen(std::string generatorType, std::string modelDescription);
    tbTriMesh GenerateTriMesh();
    void setNumRefinements(int refinements);
    ...
  private:
    // ...private data...
}
```

where

```
generatorType = "Gmsh"
```

to use Gmsh or

```
generatorType = "Structured"
```

to use the simple structured mesh generator.

The user requests a mesh generator object by instantiating the class `tbMeshGen` whose constructor takes a string argument identifying which underlying mesh generator software to use. For instance,

```
tbMeshGen smg("Structured","structured.stm")
```

creates a structured mesh generator object using the structured mesh file input file `structured.stm`. Since the structured mesh generator is not aware of regions and materials, the input file is simple. The file must have extension `*.stm` with the following format:

```
number-of-elements
xmin
xmax
ymin
ymax
boundary-condition-1
boundary-condition-2
boundary-condition-3
boundary-condition-4
```

The structured mesh generator attempts to build a mesh with the requested number of elements by equally dividing the edges. Since this is not always possible, the generator uses values N and M such that $2NM$ is closest to the desired number of elements and N and M are themselves close in value. Additional input values are ignored past the final boundary condition.

Instances of unstructured meshes are generated similarly using

```
tbMeshGen umg("Gmsh","geometry.geo");
```

which will use Gmsh to create a mesh of the domain described in the file `myModel.geo`, which has a particular format that Gmshunderstands. The format of this file is described in the Gmsh documentation (Geuzaine and Remacle, 2009).

Any mesh generator can be used by adding a subroutine to call that mesh generator and populate a `tbTriMesh` object.

4.4 Problem Defintion

The problem definition is given by the user in two components. The first component, described above, is the model definition file. In the model, each region is assigned a unique index. These indices allow the user to reference the mesh regions and prescribe material properties. The other component of the model definition is provided by the user as a concrete instance of the abstract class `tbMyProblem`.

```
class tbMyProblem {
public:
  std::vector<tbMaterialProperty> mat_array;
  double materialfunc(cont tbVertex&) = 0;
  double thebcfunc(cont tbVertex&) = 0;
  double therhsfunc(const tbVertex&) = 0;
  double my_exact(const tbVertex&) = 0;
  tbVertex my_exact_grad(const tbVertex&) = 0;
}
```

Using a pure virtual base class means that any problem definition can be referenced as a type `tbMyProblem` without regard to the specific implementation. Since the methods are pure abstract functions, the user must define an implementation for each. In particular, my assembly algorithms use `therhsfunc()` and `thebcfunc()` in constructing the finite element matrices and load vector. The functions for the exact solution and its gradient are used by the error class `tbError`.

In this class, a public property `mat_array` is a `std::vector` holds the material properties defined by region index. The user is required to provide the proper number of regions corresponding to the mesh described by the model definition file. This assignment is easily accomplished in a user-defined constructor for the concrete instance of `tbMyProblem`. Alternatively, a function specifically for material property assignment can be added to the derived class.

4.5 Mesh Intersection Algorithm

Intersecting meshes of triangular elements ultimately reduces to intersecting individual triangles. At a high level, intersection is an operation on meshes and the mesh object `tbTriMesh` contains a method

```
tbTriMesh::Intersect(const tbTriMesh& othermesh)
```

The process by which intersections are computed should remain unknown to the mesh object. This design means changes to the intersection algorithms at the element level will not change the interface at the mesh level. The implementation of `tbTriMesh::Intersect()` simply requests that each triangle of the current mesh attempt to intersect itself with elements of the input mesh. Thus, the mesh intersection algorithm reduces to computing intersections between individual triangles.

Intersecting two triangles can be broken into three major parts. At first, the question is whether or not the triangles are even close enough to warrant further computation. To this end, I enclose each triangle in a minimum bounding circle and test the intersection of these simplified objects since fewer floating point operations and memory accesses are required. If the bounding circles overlap, then the triangles are tested for intersection by applying the Separating Axis Theorem. The case of complete inclusion is also covered by this powerful theorem using a few simple interval tests, which further reduces the need for expensive point-in-triangle tests. Point-in-triangle evaluations are the final step before adding new vertices to the data structure. New vertices are added by using edge-edge intersections.

4.5.1 Minimum Bounding Circle

A common problem in computer graphics is the need to determine when two complex objects are interacting, such as characters in a video game. Complex objects under these circumstances are enclosed by simple shape that requires very few floating point operations to determine intersection. For triangles, a reasonable choice of simple shape is the minimum bounding circle. Circles provide an efficient method for checking proximity since the test requires only the distance between centers and the sum of the radii.

A simple test for the intersection of triangles is the bounding circle test. The algorithm for computing a minimum bounding circle, the smallest circle that contains all the triangle vertices, is well-known and documented in geometry REFS and computer graphics literature REF. One can observe that the center of a bounding circle will lie either at the barycenter of the triangle or at the midpoint of the longest edge REF. The radius is then determined by the distance between this center and the triangle vertex furthest from the center.

Each `tbTriangle` object contains the center and radius of its own bounding circle in this design. The `Intersect` method uses the bounding circles of each triangle as a first inexpensive test for intersection. This test is accomplished by computing the distance between the bounding circle centers and comparing that distance to the sum of the radii. If for $j = 1, 2$ the center and radius of triangle T_j are c_j and R_j respectively, then

$$\operatorname{dist}(c_1, c_2) \leq R_1 + R_2$$

when the circles are touching. Unfortunately, the bounding circle test can return a false-positive which means that the circles overlap but the triangles do not. However, the bounding circle test is computationally inexpensive and further reduces the cost of the intersection algorithm by ignoring triangles that cannot be intersecting. To simplify the storage, I introduce a data type

```
class tbCircle
{
  tbVertex center;
```

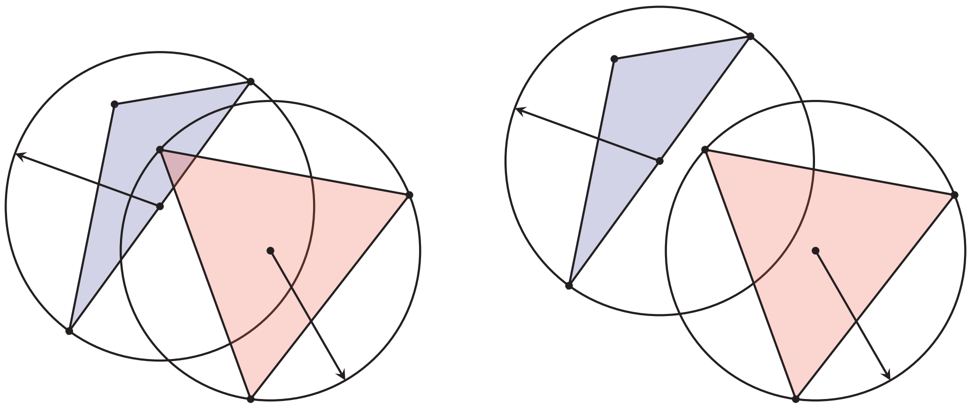

(a) Intersecting triangles. (b) Disjoint triangles.

Figure 4.2: Triangle intersection is tested first with bounding circles. Shown here are two cases where bounding circles overlap. The triangles overlap in the first case, but the second case needs further testing.

```
  double radius;
}
```

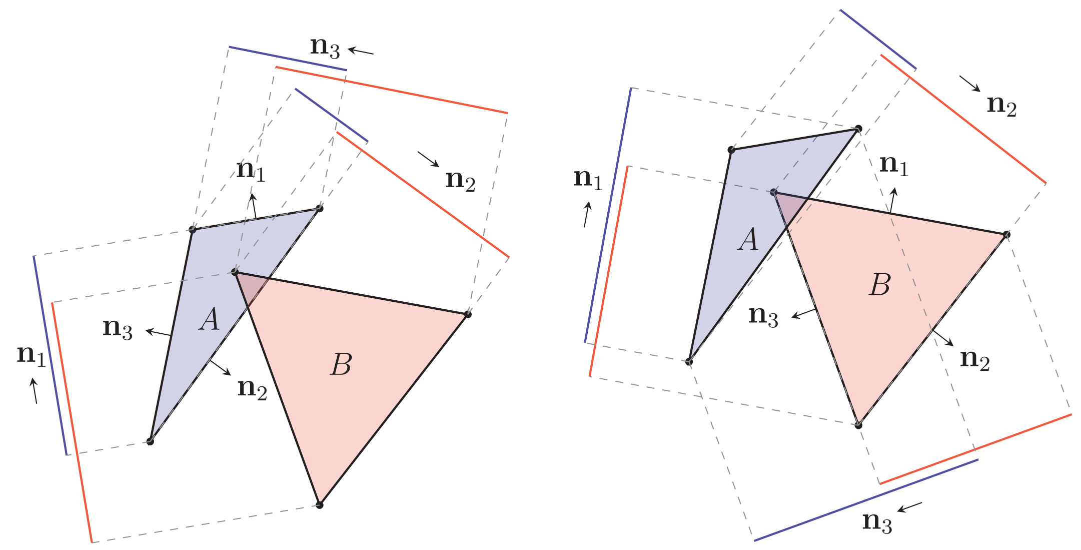
and add on object of that type to as a data member in `tbTriangle` class

```
tbCircle tbTriangle::boundingCircle
```

This encapsulation within the class `tbTriangle` means we can test bounding circle intersection using a method

```
bool tbTriangle::touchesBallsWith(const tbTriangle& t)
```

which returns `true` if the bounding circles intersect and `false` otherwise. The computational cost of the bounding circle test implemented here is linear in the number of triangles. On a computer with a 2.7 GHz Intel i7 processor, computing the bounding circles for 10^6 triangles had a wall time of 0.061 sec.



(a) Projecting onto normals of triangle A. (b) Projecting onto normals of triangle B.

Figure 4.3: The separating axis test involves projecting the two triangles A and B onto each of the 6 normal directions. When all intervals overlap, the triangles A and B are deemed intersecting since they have no separating axis.

4.5.2 Separating Axis Theorem

Provided the bounding circle test is successful, the next step to test intersection is the application of the Separating Axis Theorem (SAT). The SAT states roughly that two convex polytopes do not intersect which provides a criterion for two convex polyg This theorem is widely known in convex analysis, but the most likely application currently seen is collision detection in video game software REF. Before stating the theorem I wish to introduce the following notation: interior of set T is $\mathring{T}$, and $\langle a, b \rangle$ is the inner product of vectors $a, b \in \mathbb{R}^2$. A statement of the theorem adapted to this application from Golshtein and Tretyakov (1996) follows.

Theorem 4.5.1 (Separating Axis Theorem (Triangles))**.** *Let T_1 and T_2 be triangles in $\mathbb{R}^2$ such that $\mathring{T}_1 \cap \mathring{T}_2 = \emptyset$. Then there exists a nonzero vector $\mathbf{n} \in \mathbb{R}^2$ and a scalar*

γ such that

a) $\langle x, \mathbf{n} \rangle \leq \gamma$ for $x \in T_1$ and $\langle x, \mathbf{n} \rangle \geq \gamma$ for $x \in T_2$,

b) $\langle x_*, \mathbf{n} \rangle \neq \gamma$ for some $x_* \in T_1 \cup T_2$.

This means that there exists a vector $\mathbf{n}$ such that the projections of T_1 and T_2 onto $\mathbf{n}$ are separated on that axis, which is a line in $\mathbb{R}^2$, at the value γ. In other words, the vector $\mathbf{n}$ and scalar γ define a hyperplane (line) in $\mathbb{R}^2$ such that T_1 and T_2 are separated.

Although the SAT does not provide an algorithm for determining a separating axis, applying the SAT is straight-forward for triangles in $\mathbb{R}^2$. Since SAT is an existence theorem, finding any direction $\mathbf{n}$ means the two triangles cannot intersect. By computing the normal direction to the edges of each triangle, a total of at most six test directions can be established. A separating axis $\mathbf{n}$ may be found before all six tests are performed, which provides an early exit from the algorithm. If each of the six normal vectors are tested and pass the comparison test, then the triangle T_2 could be completely inside T_1. With six additional interval tests, total inclusion of T_2 in T_1 can be determined without using more point-in-polygon tests. Thus, the SAT test provides three possible states in one algorithm: exclusion, intersection, and total inclusion. Pseudocode for my implementation appears in Algorithm 4.1.

4.5.3 Point Masking

If the Separating Axis Theorem test reveals that a two triangles intersect but one is not completely inside the other, then further tests are necessary to determine

Algorithm 4.1 Separating Axis Test for Triangles

```
1: Given triangles $T_1$, $T_2$ in $\mathbb{R}^2$
2: Compute edge normal vectors $\{\mathbf{n}_j^1\}_{j=1}^3$ for $T_1$
3: Compute edge normal vectors $\{\mathbf{n}_j^2\}_{j=1}^3$ for $T_2$
4: Let each vertex be a vector with respect to the origin of $\mathbb{R}^2$
5: $p1$ and $p2$ are the projected intervals
6: Initialize inclusion counter $ic = 0$
7: for $j = 1 \ldots 3$ do
8:   Project the vertices of $T_1$ and $T_2$ onto $T_1$ edge normals
9:   $p1 = T_1 \text{proj}\, \mathbf{n}_j^1$
10:  $p2 = T_2 \text{proj}\, \mathbf{n}_j^1$
11:  If $p1 \cap p2 = \emptyset$ then Exit with intersection status False
12:  If $p2 \in p1$, then $ic = ic + 1$
13: end for
14: for $j = 1 \ldots 3$ do
15:  Project the vertices of $T_1$ and $T_2$ onto $T_2$ edge normals
16:  $p1 = T_1 \text{proj}\, \mathbf{n}_j^2$
17:  $p2 = T_2 \text{proj}\, \mathbf{n}_j^2$
18:  If $p1 \cap p2 = \emptyset$ then Exit with status False
19:  If $p2 \in p1$, then $ic = ic + 1$
20: end for
21: Intersection status True
22: If $ic = 6$, then $T_2$ inside $T_1$
```

the polygon $\pi = T_1 \cap T_2$. In order to reduce the number of vertices visited by the remaining algorithms, I have implemented a simple masking routine that marks vertices that match within some specified tolerance. The `tbTriangle` object has a method

```
tbMask tbTriangle::maskMatchingVertices(tbTriangle& t)}
```

where the returned object is a simple container defined as

```
class tbMask {
  tbMask();
  bool ignore[3];
}
```

A local instance of a mask object for each of the two triangles is created in method `Intersect` and used to remove existing vertices from consideration in the intersection code. When the class `tbMask` is initialized for either `tbTriangle` object, the matching vertices are marked using their local vertex numbering in the boolean array `ignore`. As the code moves through the remaining test cases, vertices that are found on edges are also marked further reducing the number of vertices visited. The primary purpose of this masking process is to improve robustness by eliminating vertices that may satisfy multiple criteria.

4.5.4 Point-In-Triangle

This point-in-triangle test uses the barycentric coordinates of the test point with respect to the triangle as a rejection criterion. Any point $p \in \mathbb{R}^2$ can be written as an affine combination of the vertices of trianlge T

$$p = \lambda_1 v_1 + \lambda_2 v_2 + \lambda_3 v_3$$

where $\{v_j\}_{j=1}^3$ are the vertices of T and the triplet $(\lambda_1, \lambda_2, \lambda_3)$ is called the barycentric coordinates of p. The triplet $(\lambda_1, \lambda_2, \lambda_3)$ is subject to the constraint that $\sum_j \lambda_j = 1$ for all $p \in \mathbb{R}^2$, which also means $\lambda_3 = 1 - \lambda_1 - \lambda_2$. For any p inside T, $0 \leq \lambda_j \leq 1$ for $j = 1, 2, 3$. Given the constraint on λ_3, the expression for p can be written in terms of only λ_1 and λ_2 as

$$p - v_3 = \lambda_1(v_1 - v_3) + \lambda_2(v_2 - v_3).$$

The benefit of this representation is that two independent linear equations in λ_1, λ_2 are readily obtained by computing $(p - v_3) \cdot v_1$ and $(p - v_3) \cdot v_2$. Forming and solving the resulting linear system yields the barycentric coordinates for p in terms of T

$$\lambda_1 = \frac{\|r_1\|\langle r_2, r_0\rangle - \langle r_1, r_0\rangle\langle r_2, r_1\rangle}{\|r_0\|\|r_1\| - \langle r_0, r_1\rangle\langle r_1, r_0\rangle}$$
$$\lambda_2 = \frac{\|r_0\|\langle r_2, r_1\rangle - \langle r_0, r_1\rangle\langle r_2, r_0\rangle}{\|r_0\|\|r_1\| - \langle r_0, r_1\rangle\langle r_1, r_0\rangle}$$

where $r_0 = v_1 - v_3$, $r_1 = v_2 - v_3$, and $r_2 = p - v_3$. Finally, the point p is inside (or on the boundary) of triangle T if $\lambda_1, \lambda_2 \geq 0$ and $\lambda_1 + \lambda_2 \leq 1$. If the condition is satisfied, then the `tbMask::ignore` array index corresponding to p is set to `true`. A test for inclusion without the boundary is obtained by replacing the conditions on λ_1, λ_2 with strict inequalities.

There is a possibility of slight numerical errors falsely excluding p from the triangle T in the case of p on some edge of T. The next method checks explicitly for points very near edges.

4.5.5 Point-On-Edge

Test point p from T_2 on an edge e of triangle T_1. Edge e of T_1 has vertices $v, w \in \mathbb{R}^2$. Unless p, v, w are colinear, a point-on-edge test could result in a false-positive. Therefore, first check that the points are colinear enough to proceed using the cross product (essentially area test)

$$|\,(w - v) \times (p - v)\,| > \text{tol}$$

where tol is chosen very small. Provided the colinearity test is successful, examine the parametric equation for the edge. Any point u along the edge e is

$$u = v + t(w - v)$$

where $t \in (0, 1)$, which excludes exact matches with the end-points of e. Since we wish to know if p is on edge e, write

$$p - v = t(w - v).$$

Must be true componentwise. Componentwise,

$$t = \frac{p_1 - v_1}{w_1 - v_1}$$

and

$$t = \frac{p_2 - v_2}{w_2 - v_2}$$

Division by zero condition is possible when $w_1 - v_1$ or $w_2 - v_2$ are nearly zero. Pathological division-by-zero cases are excluded by the cross product test for colinearity. This condition is not possible for both, so a division-by-zero test on one denominator easily identifies the appropriate logical path for computing t. Provided $0 < t < 1$, the point p is on edge e between the vertices. Possible that some vertices of T_1 could be on edges of T_2. Test is performed again with the roles of T_2 and T_1 reversed.

4.5.6 Edge Intersection

All the tests above determine whether existing points should be included or excluded from the intersection $\pi = T_1 \cap T_2$. If all the vertices of T_2 are not accounted for in the `tbMask` object associated with that triangle, then each edge of T_2 is tested against each edge of T_1 to find intersection points. Let $s, t \in (0, 1)$ be parameters. Let e_1 be an edge of T_1 with vertices v, w. Let e_2 be an edge of T_2 with vertices q, r. Then form the linear system

$$\mathcal{A} = \begin{pmatrix} q - r & v - w \end{pmatrix}$$

If $|\det \mathcal{A}| <$ tol, then matrix is too close to singular. This means the lines are nearly parallel and there is not likely to be an intersection point within either segment. Right-hand side comes from the parametric equations $q - v$. Check that $0 < s < 1$ and $0 < t < 1$. If successful, then an intersection point exists. Compute it and return. Otherwise, no new point to compute.

4.6 Harmonic Map Class

This C++ class provides the functions needed to construct and evaluate the harmonic map as introduced in Section 3.3.1. The class has the structure

```
class tbHarmonicMap {
public:
  tbHarmonicMap(tbTriMesh&);
  void build();
  tbVertex eval(tbVertex);
  tbTriMesh transform(tbTriMesh&);
  double getJacobianMatrix(long globalElementID, long row, long col);
  double getJacobian(long globalElementID);
private:
  tbTriMesh* _mesh;
  std::vector<Jacobian> _Jac;
  std::vector<tbVertex> _hMapSol;
}
```

where the type `Jacobian` is a simple container class

```
class Jacobian{
public:
  double[2][2] matrix;
  double det;
}
```

that holds a Jacobian matrix and its determinant. Since a harmonic map is a piecewise linear finite element solution, the construction of a harmonic map requires a fully defined `tbTriMesh` object as input.

Constructing the harmonic map object could take a long time depending on the mesh resolution. Therefore, I initialize the object data using the constructor and use the method `build()` to compute the map. Given a mesh object `M`, harmonic map instantiation and construction are performed explicitly as

```
tbHarmonicMap HMap(M);
HMap.build();
```

Internally, the harmonic map command `build()` uses Trilinos packages Intrepid, Shards, Amesos, Epetra, Epetra_Ext, and Teuchos to compute the finite element solution based on the Intrepid example REF. During the call to `build()` the Jacobian matrices and determinants are precomputed and each is added to the private member `_Jac`. The nodal values of the harmonic map, which are themselves vectors, are stored in the private data member `_hMapSol`. Access to point-wise harmonic map values are obtained using the command `eval`.

Recall that the harmonic map is used to map meshes to harmonic coordinates. The method `transform()` evaluates the harmonic map at all verticies of the input `tbTriMesh` object. It is often the case that the original mesh and its data are still needed for other operations. For this reason, `transform()` creates and returns an entirely new mesh object with all derived data (basis functions, Jacobian matrices, and so on) computed in terms of the mapped vertices. Given a mesh object `M`, the harmonic map is applied using the `tbTriMesh` copy constructor as

```
tbTriMesh hM(HMap(M));
```

where `hM` is the mesh mapped to harmonic coordinates.

The remaining functions `getJacobianMatrix()` and `getJacobian()` are provided to access the private data stored in `_Jac`.

4.7 Usage Demonstration

With the detailed descriptions given for some of these classes the usage in the setting of a particular problem can be difficult to see. I wish to show how most of the details given above are hidden from the user. Suppose a model is given by the Gmsh geometry file `model.geo`. Further suppose that the user has constructed a derived class `cModel` with a default constructor (a constructor that can be called with no parameters). Finally, let a structured mesh of the domain described in `model.geo` be given by `model.stm`. Then the stiffness matrix and load vector corresponding to the structured mesh nodes are given by the following example.

```
// Construct problem definition
cModel model();
// Generate meshes
tbMeshGen gm("Gmsh", "model.geo");
tbTriMesh primary(gm.GenerateTriMesh());
tbMeshGen sm("Structured", "model.stm");
tbTriMesh auxiliary(sm.GenerateTriMesh());
// Set auxiliary mesh material properties
auxiliary.SetMaterialProperties(model);
// Map meshes
tbHarmonicMap HMap(auxiliary);
tbTriMesh hauxiliary(HMap.transform(auxiliary));
tbTriMesh hprimary(HMap.transform(primary));
// Perform intersection
primary.Intersect(auxiliary);
// Construct matrix and load vector
tbFEMatAssemble Assemble(model);
Epetra_FECrsMatrix S(Assemble.CellStiffness(hprimary,hauxiliary);
Epetra_FEVector F(Assemble.CellRHS_FineRep(hprimary, HMap,
    hauxiliary, S.RowMap()));
```

At the end of this example, the user is free to choose any of a number of solvers provided by `Trilinos` or export the matrix and load vector for use by any other software package. Some conversion may be necessary in the latter case, but such conversion is generally inexpensive.

Chapter 5

Results

In this chapter, I present two problems chosen to test the approximation capability of the new algorithm. The problem of a single circular inclusion, which has $\beta(x) = 0$, is solved on a square domain. This problem demonstrates that the new assembly method improves the accuracy of the solution. A final example shows that the new method performs well on multi-interface problems.

5.1 Circular Inclusion

Let $\Omega = [-1, 1] \times [-1, 1]$. Suppose the boundary Γ is a circle of radius $r_0 = 1/\sqrt{2\pi}$ centered at $(0, 0)$.

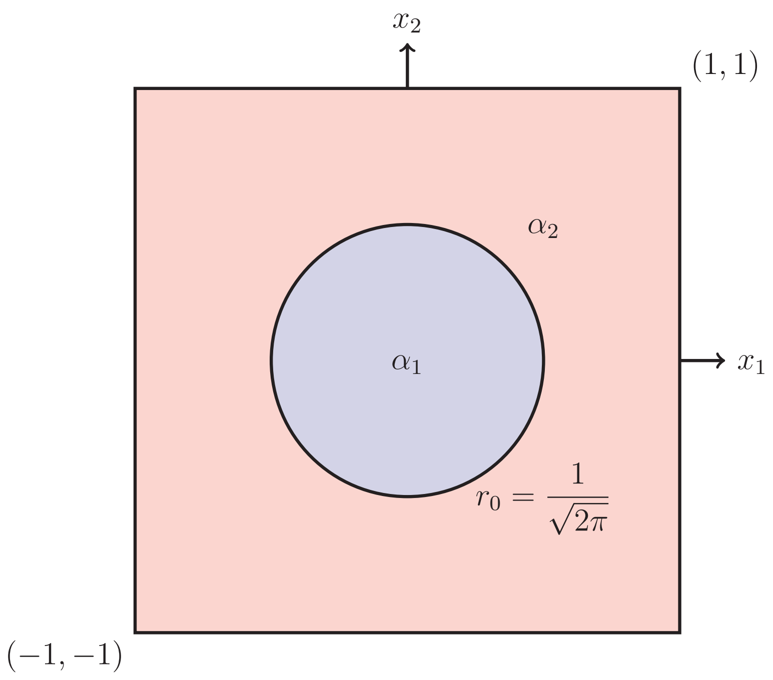


Figure 5.1: Circular inclusion. A circle of radius $r_0 = 1/\sqrt{2\pi}$ inside a biunit square domain. The coefficient values are α_1 inside the circle, and α_2 outside the circle.

Consider the equation

$$\begin{aligned} \nabla \cdot (\alpha(x) \nabla u) &= f && \text{in } \Omega, \\ u(x,y) &= g(x,y) && \text{on } \partial\Omega. \end{aligned}$$

When the coefficient function $\alpha(x)$ is piecewise constant, the exact solution has the form

$$u(x,y) = \begin{cases} \dfrac{r^p}{\alpha_1} & \text{for } r \leq r_0, \\ \dfrac{r^p}{\alpha_2} + \left(\dfrac{1}{\alpha_1} - \dfrac{1}{\alpha_2}\right) r_0^p & \text{for } r \leq r_0. \end{cases}$$

where $r = \sqrt{x^2 + y^2}$. Let $p = 3$. Then the right hand side and Dirichlet boundary condition are

$$
\begin{aligned}
f(x,y) &= 9r, \\
g(x,y) &= \frac{r^3}{\alpha_2} + \left(\frac{1}{\alpha_1} - \frac{1}{\alpha_2}\right) r_0^3.
\end{aligned}
$$

I examine the error for two cases of relatively large contrast.

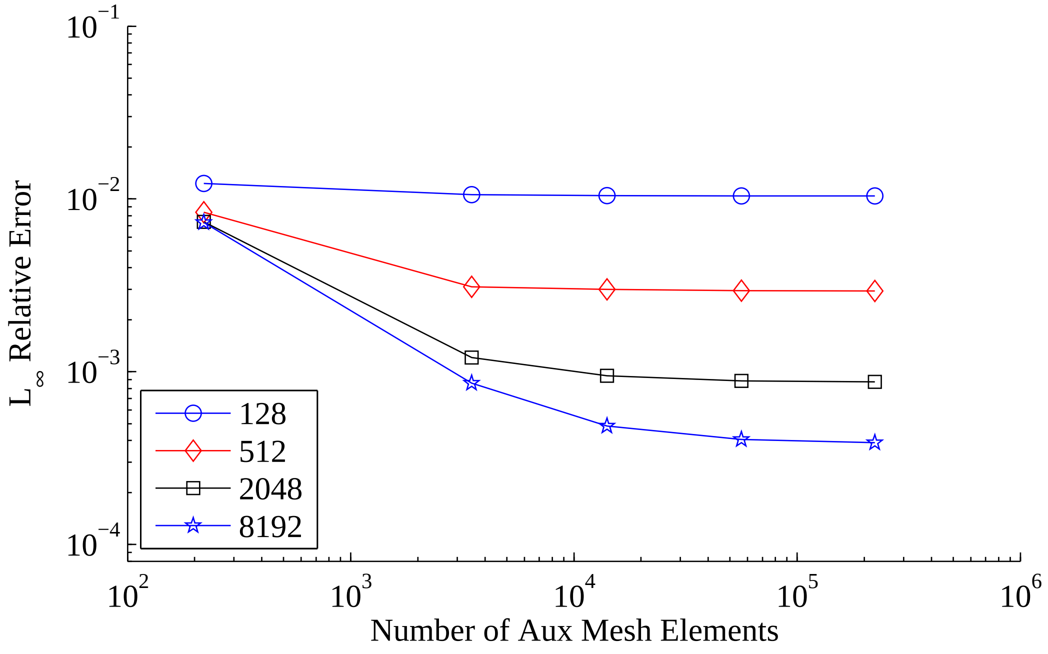
Circular Inclusion Contrast $\alpha_1 = 20$, $\alpha_2 = 1$

A first step in examining the accuracy of the solution is measuring the error as the auxiliary mesh is refined. This convergence is shown for a representative subset of the coarse meshes using the localized Galerkin basis Fig. 5.2a and the conforming HCFEM basis Fig. 5.2b. Auxiliary meshes with 220 to 220000 triangles are used to compute the harmonic map. Note that as the auxiliary mesh is refined, the nodal values of the solution are reduced and appear to converge as the resolution of the harmonic map is improved. This demonstrates that even on a relatively simple domain a large number of auxiliary mesh elements are required. Clearly, the conforming method reaches a lower error level in $\ell^\infty(\Omega)$.

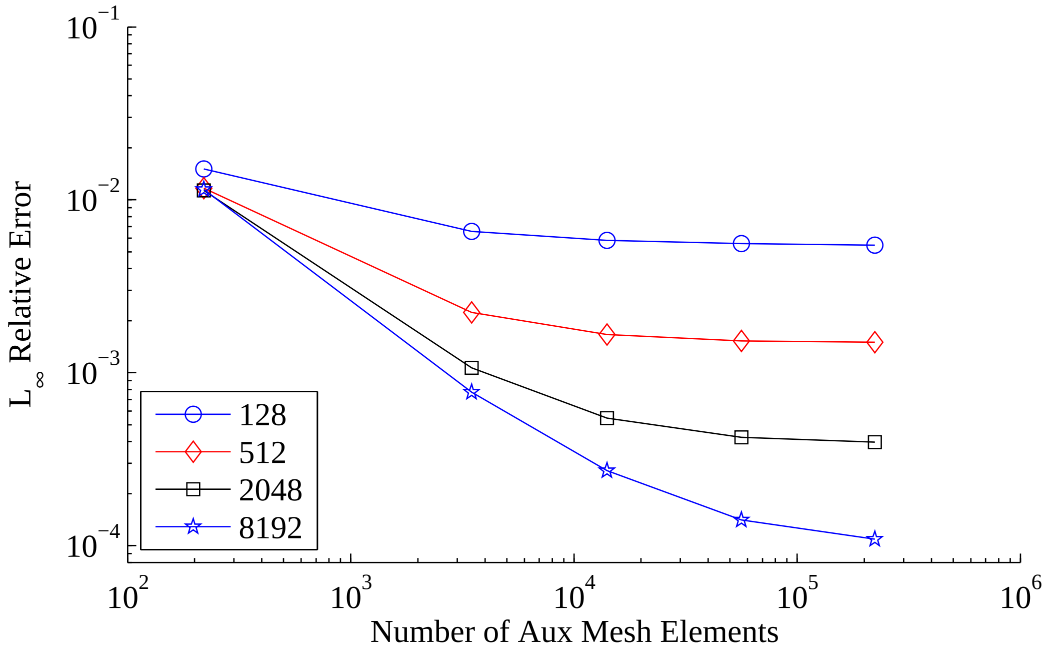
Using the highest auxiliary mesh refinement of 220000 elements, the solution is computed by each method on a family of coarse, structured meshes with 32 to 12800 triangular elements. The error measured in the $\ell^\infty(\Omega)$-norm is shown in Fig. 5.3. A circular inclusion poses a difficult scenario for an unfitted finite element method because the interface is never aligned with the mesh. Furthermore, the position of the interface within elements changes as the coarse mesh is refined. Since no fine-scale information is include in the unfitted FEM, the error level is poor and the rate of convergence is $\mathcal{O}(h)$. For this same set of coarse meshes, both the non-conforming and conforming HCFEM methods significantly improve the error level. However the least-squares convergence rate for the non-conforming method is $\mathcal{O}(h^{1.55})$ while the conforming HCFEM is has an optimal rate of $\mathcal{O}(h^{1.9})$.

Examining these errors in the $\ell^\infty(\Omega)$-norm reveals the worst-case nodal error. An alternative point of view is the relative discrete $\ell^2(\Omega)$-norm. These errors are shown in Fig. 5.4. Notice that the error level is higher for the non-conforming method. The rates of convergence estimated in $\ell^2(\Omega)$ for both the non-conforming and conforming method are comparable.

One may choose to use the coarse-scale solution as if it corresponds to a standard piecewise linear finite element method. Estimates for standard FEM are usually measured in the $L^2(\Omega)$-norm. This is shown in Fig. 5.5. Observe that the $L^2(\Omega)$-norm gives roughly the same rate of convergence and error level for the non-conforming and conforming methods. The $L^2(\Omega)$-norm tends to smooth the error, which is why these two methods appear comperable. However, the nodal values tell a completely different story in $\ell^\infty(\Omega)$.



(a) Non-conforming method using Galerkin localized elements.



(b) New HCFEM method using polygon intersection algorithm.

Figure 5.2: Error improvement in $L^\infty(\Omega)$ error as the auxiliary mesh is refined from 220 elements to $220,000$ elements. The error is shown for a selection of coarse meshes with $\alpha_1 = 20$, $\alpha_2 = 1$.

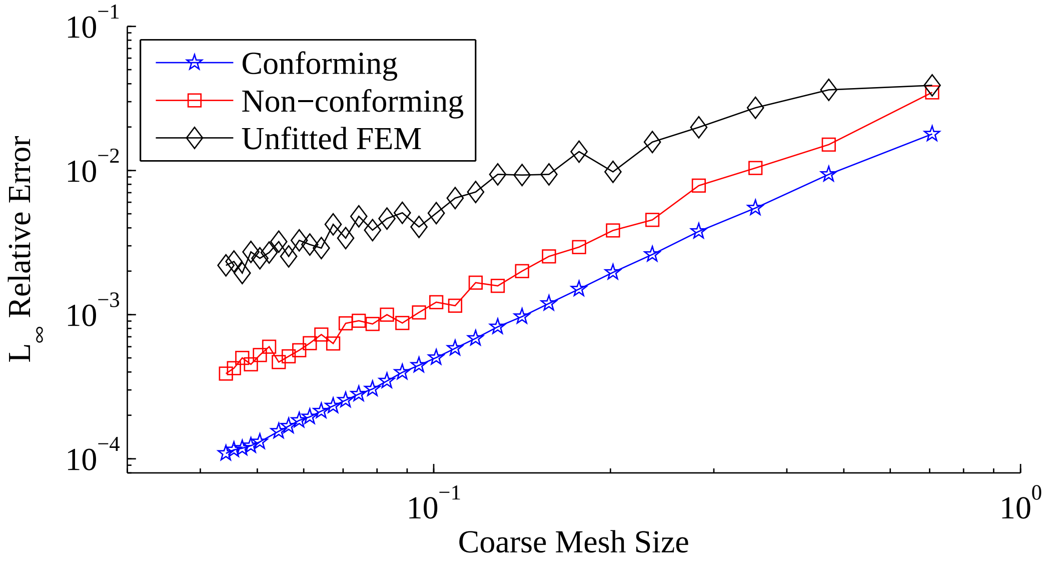


Figure 5.3: Plot showing $\ell^{\infty}(\Omega)$ convergence for non-conforming HCFEM, conforming HCFEM, and an unfitted finite element method. The coefficient function has values $\alpha_1 = 20$, $\alpha_2 = 1$. All methods solutions were computed on the same family of coarse meshes. The HCFEM methods are shown using the highest refinement for a uniform auxiliary mesh (220k elements). A least-squares estimate for the convergence rate is $\mathcal{O}(h^{1.9})$ for the conforming method and $\mathcal{O}(h^{1.55})$ for the non-conforming method. Unfitted is $\mathcal{O}(h)$.

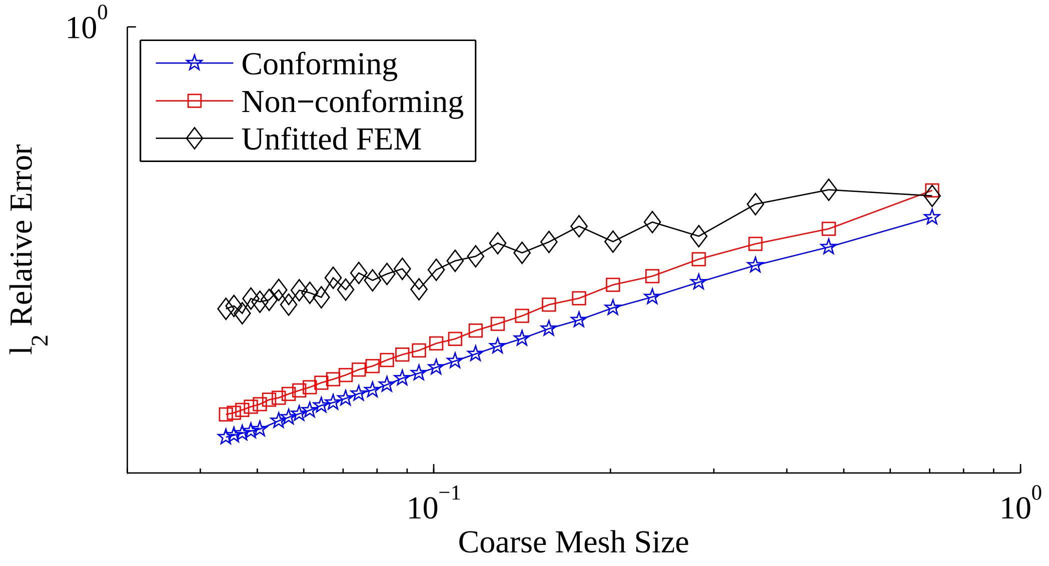


Figure 5.4: Plot showing $\ell^2(\Omega)$ convergence for non-conforming HCFEM, conforming HCFEM, and an unfitted finite element method. The coefficient function has values $\alpha_1 = 20$, $\alpha_2 = 1$. All methods solutions were computed on the same family of coarse meshes. The HCFEM methods are shown using the highest refinement for a uniform auxiliary mesh (220k elements). A least-squares estimate for the convergence rate is $\mathcal{O}(h^{1.73})$ for the conforming method and $\mathcal{O}(h^{1.72})$ for the non-conforming method. Unfitted is $\mathcal{O}(h)$.

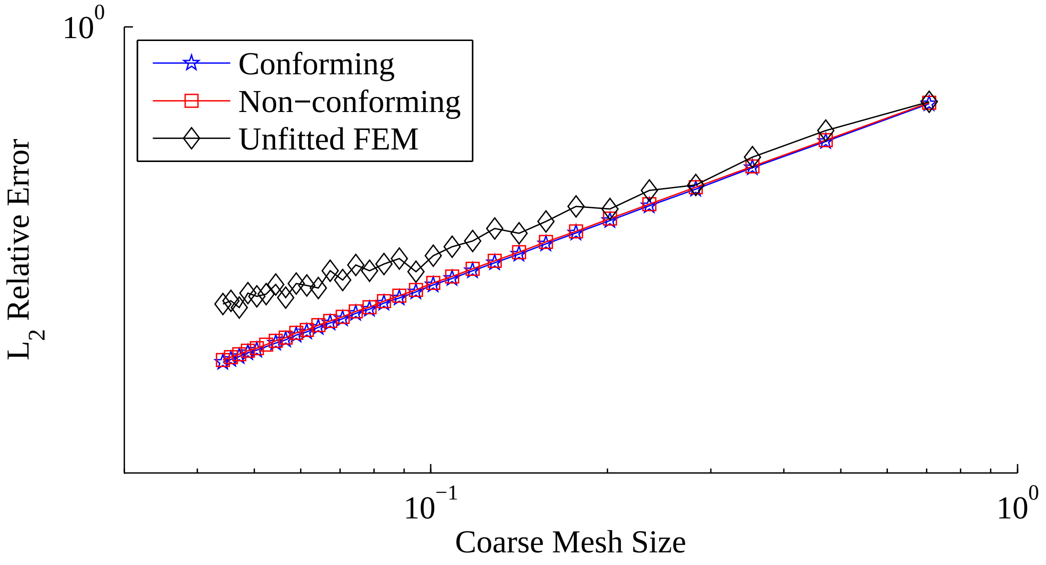


Figure 5.5: Plot showing $L^2(\Omega)$ convergence for non-conforming HCFEM, conforming HCFEM, and an unfitted finite element method. The coefficient function has values $\alpha_1 = 20$, $\alpha_2 = 1$. All methods solutions were computed on the same family of coarse meshes. The HCFEM methods are shown using the highest refinement for a uniform auxiliary mesh (220k elements). A least-squares estimate for the convergence rate is $\mathcal{O}(h^{1.98})$ for the conforming method and $\mathcal{O}(h^{1.96})$ for the non-conforming method. Unfitted is $\mathcal{O}(h^{1.5})$.

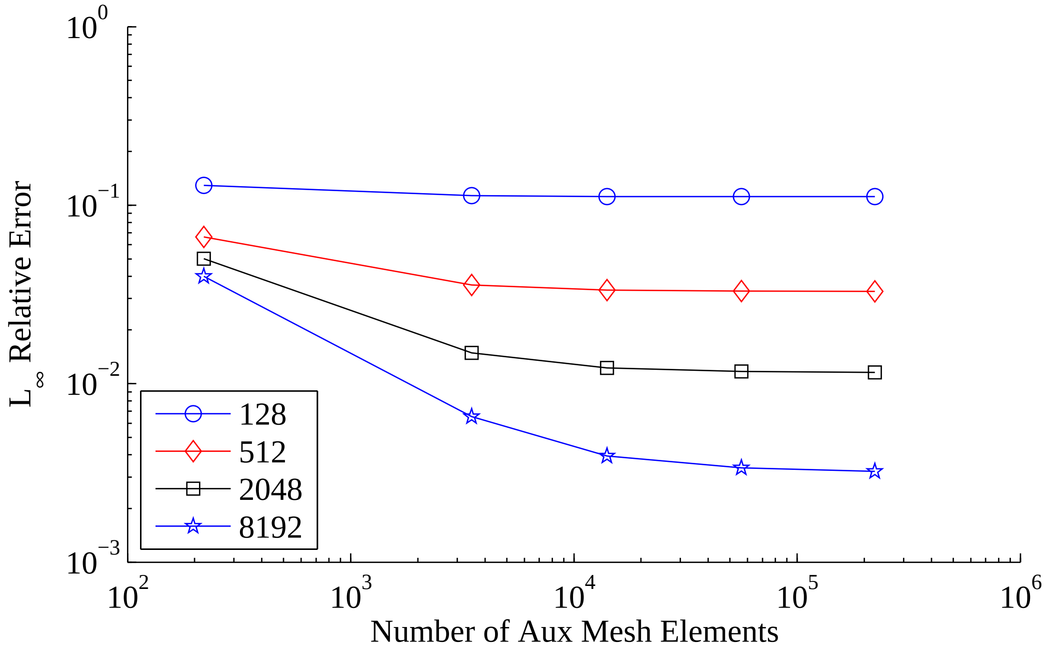
Circular Inclusion Contrast $\alpha_1 = 1$, $\alpha_2 = 20$

Reversing the contrast, convergence for the same auxiliary meshes is shown for a representative subset of the coarse meshes using the localized Galerkin basis Fig. 5.6a and the conforming HCFEM basis Fig. 5.6b. Similarly, the coarse mesh solutions converge as the auxiliary mesh is refined.

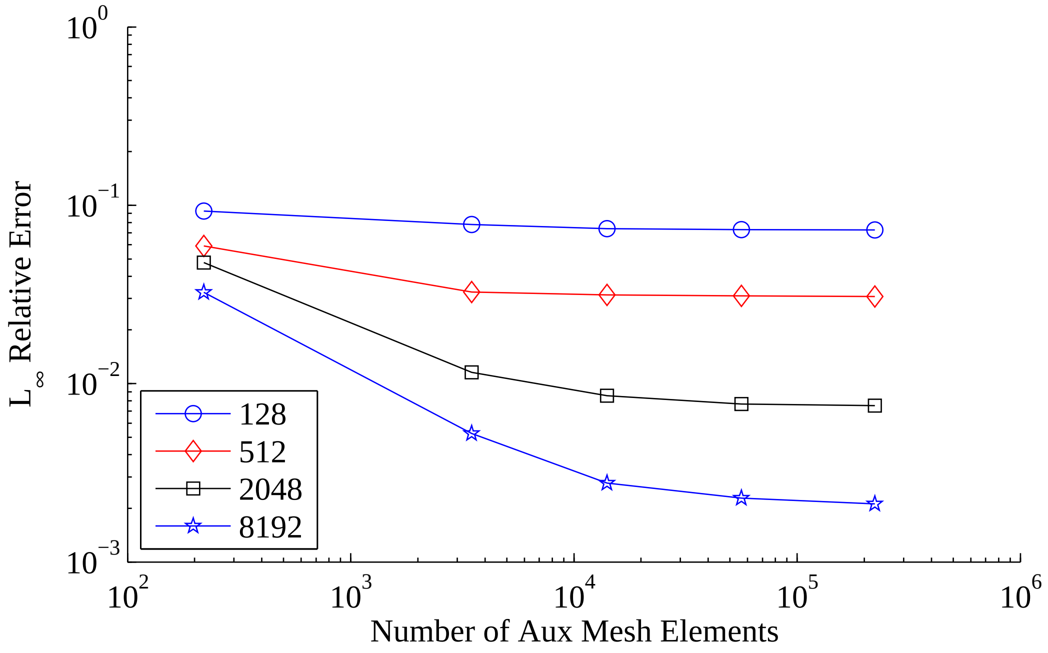
Using the highest auxiliary mesh refinement of 220000 elements, the solution is computed by each method on a family of coarse, structured meshes with 32 to 12800 triangular elements. The error measured in the $\ell^\infty(\Omega)$-norm is shown in Fig. 5.7. The unfitted method has the same degree of difficulty resolving the solution exhibiting $\mathcal{O}(h^{0.8})$ convergence. Interestingly, both the non-conforming and conforming methods show an approximate rate of convergence $\mathcal{O}(h^{1.84})$ after the first few meshes. For this particular contrast, it appears that the non-conforming and conforming methods are comparable.

These errors are shown in $\ell^2(\Omega)$-norm in Fig. 5.8. While the error levels are different, both the non-conforming and conforming methods have the same $\mathcal{O}(h^{1.7})$ convergence.

Finally, the $L^2(\Omega)$-norm errors are shown in Fig. 5.9. As with the first example, the rate of convergence is optimal for both methods. This demonstrates that a single measure of the error for interface problems is usually inadequate to truly determine how well the method will perform. In some cases, the behavior of the error in the $L^2(\Omega)$-norm is a good indicator of the overall solution error.



(a) Non-conforming method using Galerkin localized elements.



(b) New HCFEM method using polygon intersection algorithm.

Figure 5.6: Error improvement in $L^\infty(\Omega)$ error as the auxiliary mesh is refined from 220 elements to $220,000$ elements. The error is shown for a selection of coarse meshes with $\alpha_1 = 1$, $\alpha_2 = 20$. There is little difference between the error levels of the methods.

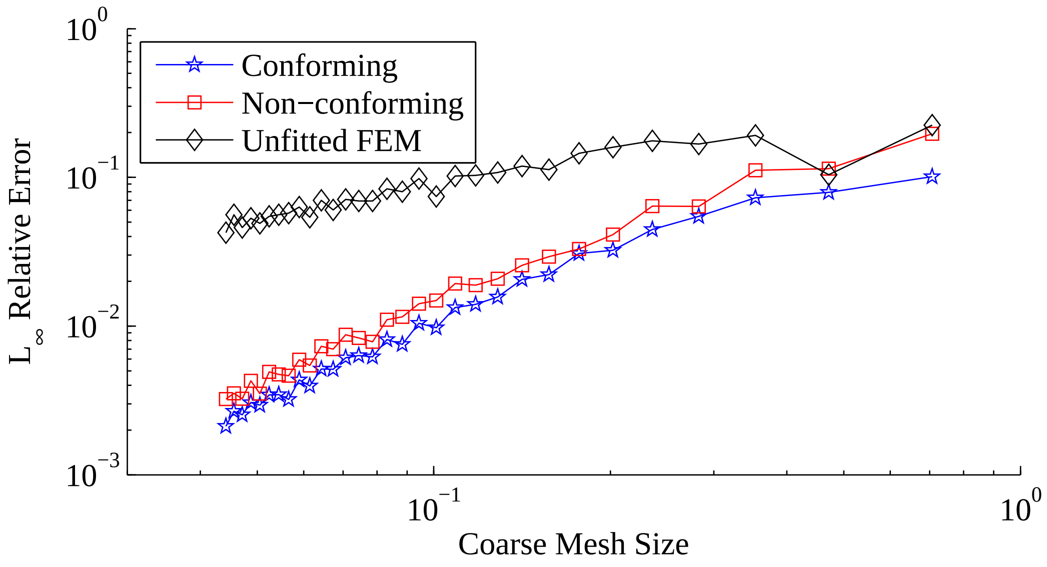


Figure 5.7: Plot showing $\ell^\infty(\Omega)$ convergence for non-conforming HCFEM, conforming HCFEM, and an unfitted finite element method. The coefficient function has values $\alpha_1 = 1$, $\alpha_2 = 20$. All methods solutions were computed on the same family of coarse meshes. The HCFEM methods are shown using the highest refinement for a uniform auxiliary mesh (220k elements). The methods appear to be pre-convergent for the first several meshes. A least-squares estimate for the convergence rate after that section is $\mathcal{O}(h^{1.84})$ for both the conforming and non-conforming methods. Unfitted is $\mathcal{O}(h^{0.8})$.

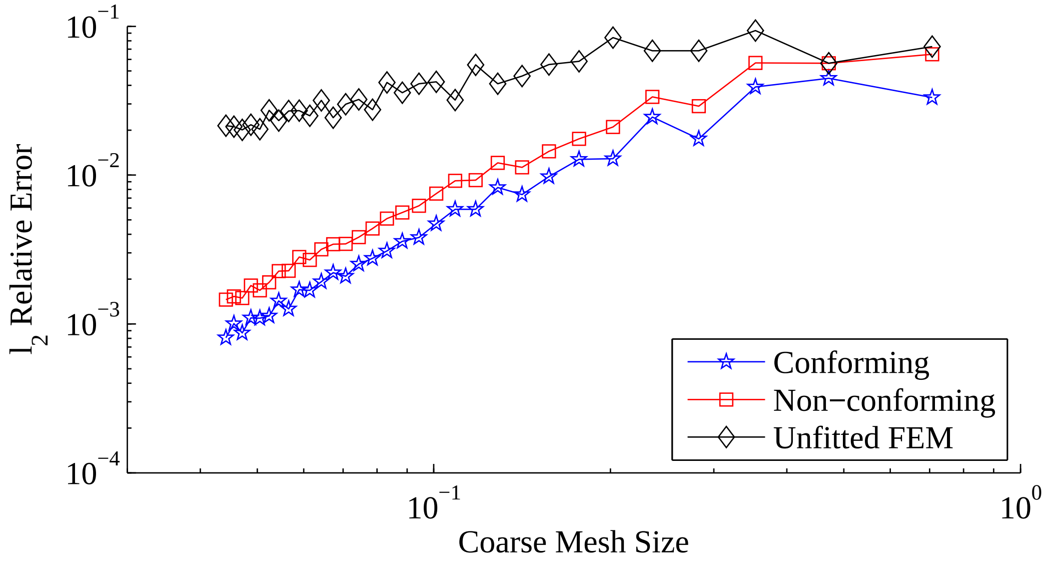


Figure 5.8: Plot showing $\ell^2(\Omega)$ convergence for non-conforming HCFEM, conforming HCFEM, and an unfitted finite element method. The coefficient function has values $\alpha_1 = 1$, $\alpha_2 = 20$. All methods solutions were computed on the same family of coarse meshes. The HCFEM methods are shown using the highest refinement for a uniform auxiliary mesh (220k elements). A least-squares estimate for the convergence rate is $\mathcal{O}(h^{1.73})$ for the conforming method and $\mathcal{O}(h^{1.72})$ for the non-conforming method. Unfitted is $\mathcal{O}(h)$.

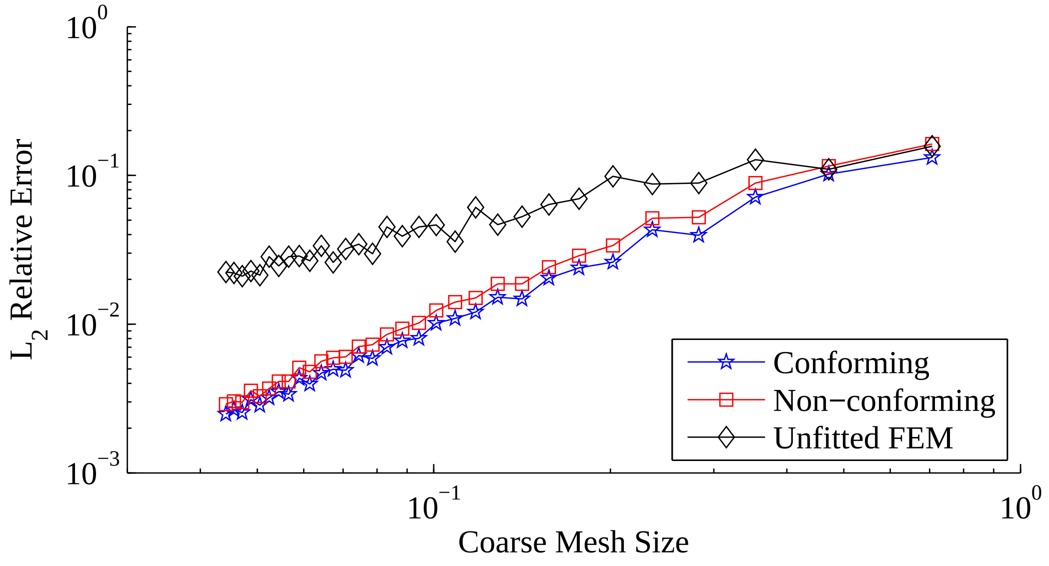


Figure 5.9: Plot showing $L^2(\Omega)$ convergence for non-conforming HCFEM, conforming HCFEM, and an unfitted finite element method. The coefficient function has values $\alpha_1 = 20$, $\alpha_2 = 1$. All methods solutions were computed on the same family of coarse meshes. The HCFEM methods are shown using the highest refinement for a uniform auxiliary mesh (220k elements). A least-squares estimate for the convergence rate is $\mathcal{O}(h^{1.98})$ for the conforming method and $\mathcal{O}(h^{1.96})$ for the non-conforming method. Unfitted is $\mathcal{O}(h^{1.5})$.

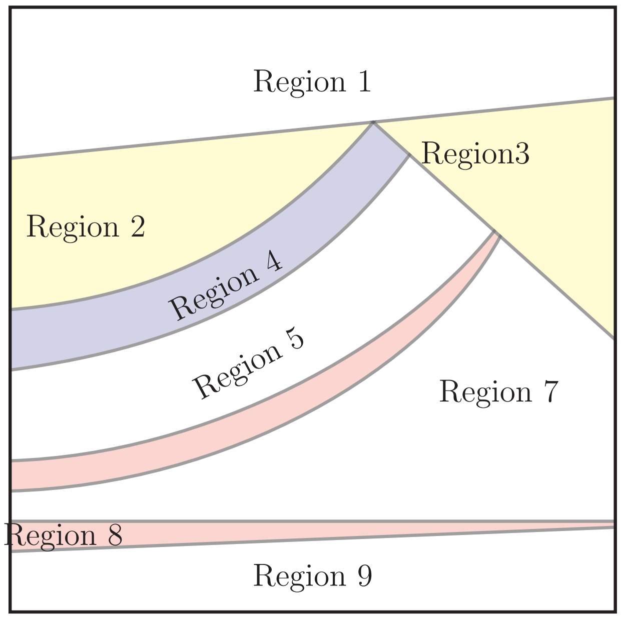


Figure 5.10: Marmousi-like. Regions 1,5,7 and 9 have $\alpha = 1$. Regions 2 and 3 have $\alpha = 2$. Region 6 has $\alpha = 10$ and region 8 has $\alpha = 20$.

5.2 Multi-layerd Medium

As a final test problem, I have chosen a multilayered medium inspired by the very complex Marmousi model.

For comparison, the problem is solved by finite elements on a fitted fine mesh with 2 280 000 elements for comparison. This fine mesh solution is used as an exact solution in the error calcluations as described in the previous chapter.

I seek the solution to the problem

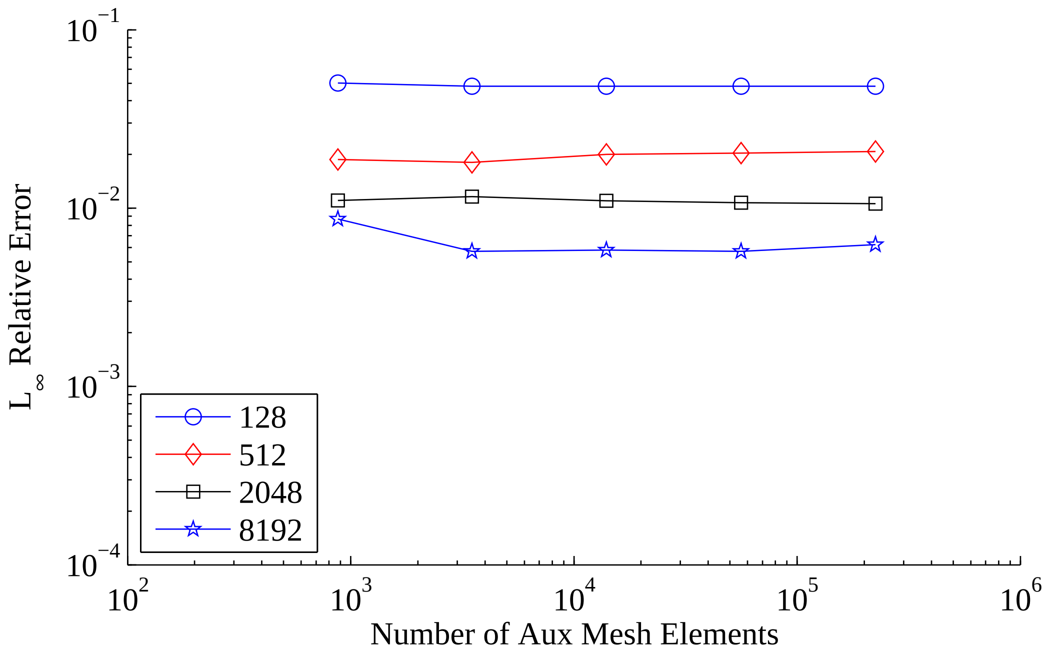
$$\begin{aligned} \nabla \cdot (\alpha(x) \nabla u) &= 1 && \text{in } \Omega, \\ u(x,y) &= 0 && \text{on } \partial\Omega. \end{aligned}$$

with the coefficient function $\alpha(x)$ defined as constant within each region. The coefficient is defined as follows: $\alpha = 1$ in regions $1, 5, 7, 9$, $\alpha = 2$ in regions $2, 3$, $\alpha = 10$ in region 6, and $\alpha = 20$ in region 8. The domain is shown in Fig. 5.10.

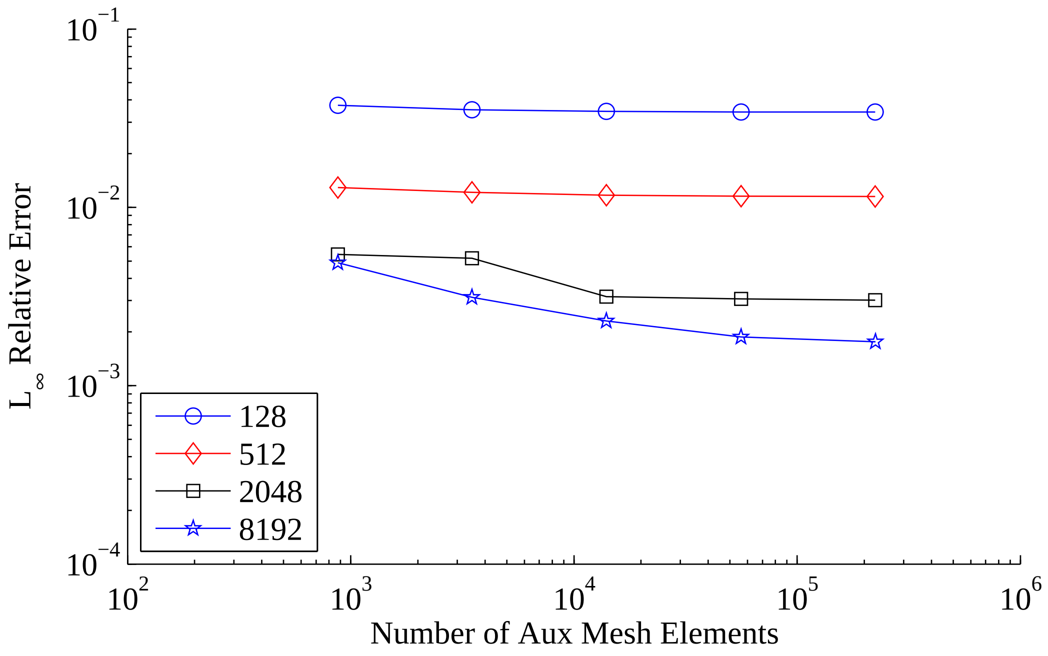
As with the circular inclusion case, there is a dramatic improvement in the error level between the two methods as the auxiliary mesh is refined in Fig. 5.11. Successive refinements of the harmonic map problem shown in Fig. 5.11b that the accuracy of the solution is improved for the new method. However, there appears to be some degree of stagnation for the finest mesh. Notice that the error for the non-conforming method does not decrease monotonically in Fig. 5.11a as it does for the conforming method. In some cases, the error actually increases as the auxiliary mesh is refined.

This behavior is also apparent when looking at the convergence rate as the coarse mesh is refined. The estimated rate of convergence for the new method is $\mathcal{O}(h^{1.5})$ for the data shown in Fig. 5.12. However, the rate of convergence for the first three meshes is $\mathcal{O}(h^{1.75})$. The full rate may be pessimistic based on the apparent stagnation for auxiliary mesh convergence on the finest coarse mesh. This slow-down in convergence is likely due to floating point limitations when computing the fine mesh solution for comparison.

In $\ell^2(\Omega)$, the relative error for the new method is solidly second-order for this complex problem. The non-conforming method has an estimated rate of convergence that is $\mathcal{O}(h^{1.5})$. Either method out-performs the unfitted method which has a rate of $\mathcal{O}(h^{1.3})$.



(a) Non-conforming method using Galerkin localized elements.



(b) New HCFEM method using polygon intersection algorithm.

Figure 5.11: Error improvement in $L^\infty(\Omega)$ error as the auxiliary mesh is refined from 800 elements to $220,000$ elements. The error is shown for three coarse meshes.

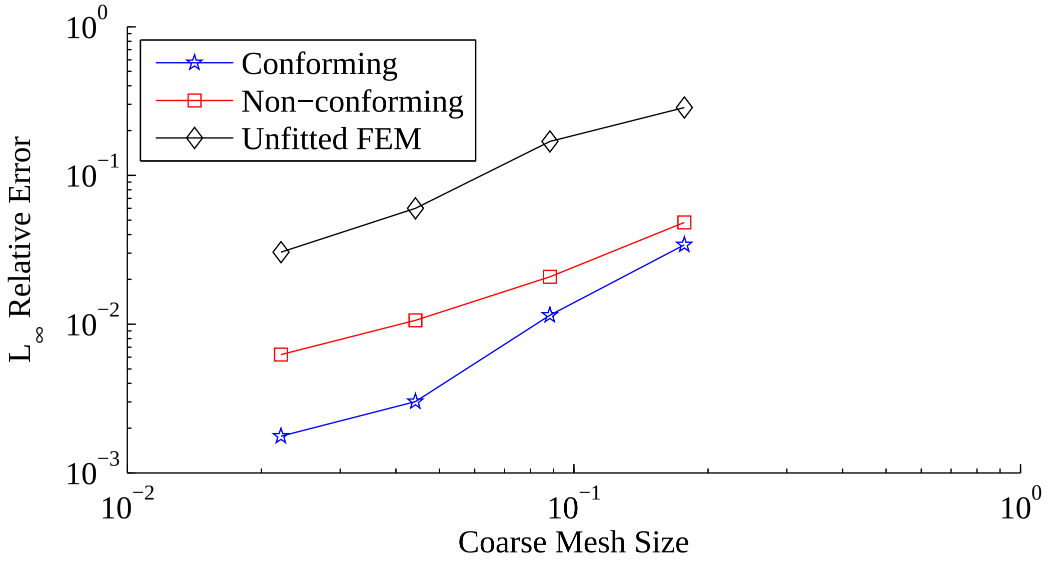


Figure 5.12: Plot showing $\ell^\infty(\Omega)$ convergence for non-conforming HCFEM, conforming HCFEM, and an unfitted finite element method on the multi-layerd model. All methods solutions were computed on the same family of coarse meshes. The HCFEM methods are shown using a highly refined auxiliary mesh (220k elements). A least-squares estimate for the convergence rate is $\mathcal{O}(h)$ for the unfitted and non-conforming methods. The new method is $\mathcal{O}(h^{1.75})$ for the first three meshes and $\mathcal{O}(h^{1.5})$ for the whole section.

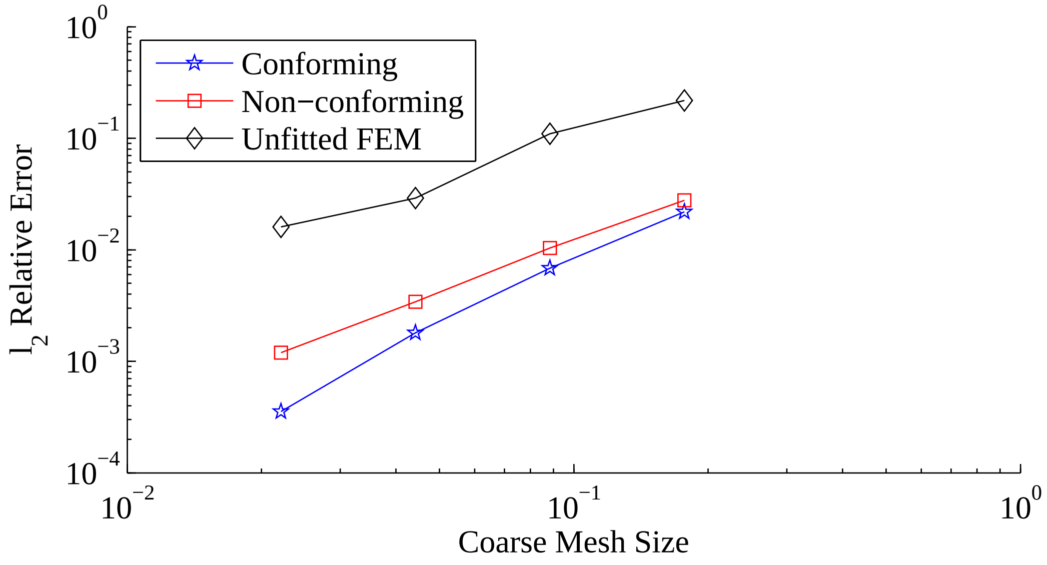


Figure 5.13: Plot showing relative ℓ_2-norm convergence for non-conforming HCFEM, conforming HCFEM, and an unfitted finite element method on the multi-layerd model. All methods solutions were computed on the same family of coarse meshes. The HCFEM methods are shown using a uniform auxiliary mesh (220k elements). A least-squares estimate for the convergence rate is $\mathcal{O}(h\,)$ for the unfitted FEM and $\mathcal{O}(h^{1.5})$ for the non-conforming method. The new method is $\mathcal{O}(h^{1.98})$.

Chapter 6

Conclusions

In this thesis, I developed a new method for constructing conforming high-resolution basis functions suitable for solving problems with piecewise constant coefficients on regular grids. The key to the assembly algorithm is the intersection algorithm used to determine which auxiliary mesh elements make up the support for the composite basis functions. Using these mesh-element intersections, I demonstrated that the finite element mass and stiffness matrices can be constructed without a quadrature scheme.

The computational results show that the new method is capable of solving the elliptic model problem more accurately than the non-conforming localized Galerkin method on which it is based. I demonstrated that a single error norm may not provide enough information to evaluate the performance of a given method for elliptic interface problems. Further, I show that even simple interface problems, such as the circular inclusion, require a highly resolved solution to the harmonic coordinate problem.

While I show only single examples of elliptic problems, I should note that the power of this method is in the assembly process. The discrete operators, the mass and stiffness matrices, need to be computed only once per simulation. In the case of elliptic problems, a large number of source functions may be used without reassembling the finite element matrices. Additionally, the solution is well-resolved at the coarse mesh level. This means that the overall computational cost of modeling complex media can be reduced dramatically. As shown in the examples, the conforming HCFEM method produces nodal values of the solution on a coarse structured mesh are far more accurate than the associated unfitted finite element solution. Thus, one obtains a better solution with the only additional cost being the solution of a simple elliptic problem and mesh-element intersections.

6.1 Future Work

My data encapsulation model for the intersection algorithm leads to duplicate calculations because the primitive objects are triangles. A superior method data model may be applying the intersections to a set of edge objects. Since each edge in a mesh is unique, once an edge intersection is computed for a particular triangle the work is done for its neighbor. This model requires more planning and design than simple triangle intersections. Importantly, an extension to higher dimensions may be simplified greatly since the intersections would follow a hierarchy of tetrahedra, triangular faces, and edges. Exploring this issue is worth consideration since the problem is not trivial. Furthermore, the high degree of optimization and parallelization in current solvers for elliptic problems means the harmonic map calculation is quite inexpensive. The intersection algorithm is the primary cost for my implementation.

References

Alessandrini, G. and Nesi, V. (2001). Univalent σ-Harmonic mappings. *Arch. Rational Mech. Anal.*, 158:155–171.

Alessandrini, G. and Nesi, V. (2003). Univalent σ-Harmonic mappings: Connections with quasiconformal mappings. *Journal D'Analyse Mathematique*, 90:197–215.

Allaire, G. and Brizzi, R. (2004). A multiscale finite element method for numerical homogenization. Technical Report R.I. No. 545, Centre De and Mathématiques Appliqués.

Anton, H. (1988). *Calculus*. John Wiley & Sons, New York, 3rd edition.

Bensoussan, A., Lions, J.-L., and Papanicolaou, G. (1978). *Asymptotic Analysis for Periodic Structures*. Number 5 in Studies in Mathemtics and its Applications. North-Holland Publishing Company, Amsterdam, New York, Oxford.

Bochev, P., Ridzal, D., and Peterson, K. (November 2009). Intrepid - Home. http://trilinos.sandia.gov/packages/intrepid.

Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., and Levy, B. (2010). *Polygon Mesh Processing*. A K Peters Ltd., Massachusetts.

Brenner, S. C. and Scott, L. R. (2002). *The Mathematical Theory of Finite Element Methods*. Springer Verlag, Berlin, Heidelberg, New York, second edition.

Briane, M., Milton, G., and Nesi, V. (2004). Change of Sign of the Corrector's Determinant for Homogenization in Three-Dimensional Conductivity. *Archive for Rational Mechanics and Analysis*, 173(1):133–150.

Brown, D. L. (1984). A Note on the Numerical Solution of the Wave Equation with Piecewise Smooth Coefficients. *Mathematics of Computation*, 45(166):369–391.

Cattani, C. and Paoluzzi, A. (1990). Boundary Integration Over Linear Polyhedra. *Computer Aided Design*, 22(2):130–135.

Chu, C.-C., Hou, T., and Graham, I. (2010). A New Multiscale Finite Element Method for High-Contrast Elliptic Interface Problems. *Mathematics of Computation*, 79(272):1915–1955.

Ciarlet, P. G. (2002). *The finite element method for elliptic problems*, volume 40 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA. Reprint of the 1978 original [North-Holland, Amsterdam; MR0520174 (58 #25001)].

Cohen, G. C. (2002). *Higher Order Numerical Methods for Transient Wave Equations*. Scientific Computing. Springer, Berlin.

Efendiev, Y. and Hou, T. Y. (2000). *Multiscale Finite Element Methods*, volume 4 of *Surveys and Tutorials in the Applied Mathematical Sciences*. Springer, New York.

Ern, A. and Guermond, J.-L. (2004). *Theory and Practice of Finite Elements*, volume 159 of *Applied Mathematical Sciences*. Springer-Verlag, New York.

Evans, L. C. (1998). *Partial Differential Equations*, volume 19 of *Graduate Texts in Mathematics*. American Mathematical Society, Providence, Rhode Island.

Geuzaine, C. and Remacle, J.-F. (2009). Gmsh: A Three-dimensional Finite Element Mesh Generator with Built-in Pre- and Post-processing Facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331.

Golshtein, E. G. and Tretyakov, N. V. (1996). *Modified Lagrangians and Monotone Maps in Optimization*. Wiley-Interscience, New York.

Haroske, D. and Triebel, H. (2008). *Distributions, Sobolev Spaces, and Elliptic Equations*. Textbooks in Mathematics. European Mathematical Society, Z urich.

Heroux, M., Bartlett, R., Hoekstra, V. H. R., Hu, J., Kolda, T., Lehoucq, R., Long, K., Pawlowski, R., Phipps, E., Salinger, A., Thornquist, H., Tuminaro, R., Willenbring, J., and Williams, A. (2003a). An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories.

Heroux, M. A. and Willenbring, J. M. (2003). Trilinos Users Guide. Technical Report SAND2003-2952, Sandia National Laboratories.

Heroux, M. A., Willenbring, J. M., and Heaphy, R. (2003b). Trilinos Developers Guide. Technical Report SAND2003-1898, Sandia National Laboratories.

Heroux, M. A., Willenbring, J. M., and Heaphy, R. (2003c). Trilinos Developers Guide Part II: ASCI Software Quality Engineering Practices Version 1.0. Technical Report SAND2003-1899, Sandia National Laboratories.

Herrmann, F. J. (1997). *A Scaling Medium Representation: A Discussion on Well-logs, Fractals and Waves*. PhD thesis, Delft University, The Netherlands.

Hou, T., Wu, X., and Cai, Z. (1999). Convergence of a Multiscale Finite Element Method for Elliptic Problems with Rapidly Oscillating Coefficients. *Mathematics of Computation*, 68(227):913–943.

Hou, T. and Wu, X. H. (1997). A Multiscale Finite Element Method for Elliptic Problems in Composite Materials and Porous Media. *J. Comput. Phys.*, 134(1):169–189.

Hou, T. Y., Wu, X.-H., and Zhang, Y. (2004). Removing the Cell Resonance Error in the Multiscale Finite Element Method Via a Petrov-Galerkin Formulation. *Comm. Math. Sci*, 2(2):185–205.

Kafafy, R., Lin, T., Lin, Y., and Wang, J. (2005). Three-dimensional immersed interface finite element methods for electric field simulation in composite materials. *International Journal for Numerical Methods in Engineering*, 64:940–972.

Kozlov, S. M. (1980). Averaging of Random Operators. *Mathematics of the USSR-Sbornik*, 37(2):167–180.

Leveque, R. J. and Li, Z. (1994). The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM Journal on Numerical Analysis*, 31:1019–1044.

Li, Z. (1998). The Immersed Interface Method Using a Finite Element Formulation. *Applied Numerical Mathematics*, 27:253–267.

Li, Z. and Ito, K. (2006). *The immersed interface method: numerical solutions of PDEs involving interfaces and irregular domains*, volume FR33 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

Maugeri, A., Palagachev, D. K., and Softova, L. G. (2000). *Elliptic and Parabolic Equations with Discontinuous Coefficients*, volume 109 of *Mathematical Research*. Wiley-VCH, Berlin, New York.

Owhadi, H. and Zhang, L. (2006). Metric-Based Upscaling. *Communications on Pure and Applied Mathematics*, 60(5):675–723.

Papanicolaou, G. (1998). Mathematical Problems in Geophysical Wave Propagation. In *Proceedings of the International Congress of Mathematicians*, volume I, pages 403–427. Documenta Mathematica.

Peskin, C. S. (1972). Flow Patterns Around Heart Valves: A Numerical Method. *Journal of Computational Physics*, 64:252–271.

Sala, M., Heroux, M. A., and Day, D. M. (2004). Trilinos Tutorial. Technical Report SAND2004-2189, Sandia National Laboratories.

Strang, G. and Fix, G. (1973). *An Analysis of the Finite Element Method*. Prentice-Hall Series in Automatic Computation. Prentice-Hall, Englewood Cliffs, New Jersey.

Symes, W. W. and Vdovina, T. (2009). Interface Error Analysis for Numerical Wave Propagation. *Computational Geosciences*, 13:363–370.

Wang, X. (2009). Discontinuous Galerkin Time Domain Methods for Acoustics and Comparison with Finite Difference Time Domain Methods. Master's thesis, Rice University, Houston, Texas.