An Adaptive Mixed Finite Element Method using the Lagrange Multiplier Technique

by

Michael Gagnon

A Project Report

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Applied Mathematics

by

May 2009

APPROVED:

Professor Marcus Sarkis, Capstone Advisor

Professor Bogdan Vernescu, Department Head

Abstract

Adaptive methods in finite element analysis are essential tools in the efficient computation and error control of problems that may exhibit singularities. In this paper, we consider solving a boundary value problem which exhibits a singularity at the origin due to both the structure of the domain and the regularity of the exact solution. We introduce a hybrid mixed finite element method using Lagrange Multipliers to initially solve the partial differential equation for the both the flux and displacement. An a posteriori error estimate is then applied both locally and globally to approximate the error in the computed flux with that of the exact flux. Local estimation is the key tool in identifying where the mesh should be refined so that the error in the computed flux is controlled while maintaining efficiency in computation. Finally, we introduce a simple refinement process in order to improve the accuracy in the computed solutions. Numerical experiments are conducted to support the advantages of mesh refinement over a fixed uniform mesh.

Acknowledgments

I would like to express my gratitude to my advisor, Professor Sarkis for all his dedication, patience, and advice. Also, I would like to thank all my professors for their motivation and guidance during the past two years at WPI. Of course, I can't forget to thank my family and friends for all their support.

Contents

List of Figures

Chapter 1

Introduction

1.1 A Model Problem and its Weak Formulation

In this paper, we are interested in solving the following two dimensional boundary value problem for $u \in H^1(\Omega)$:

$$-\Delta u = f \text{ in } \Omega \subset R^2$$

$$u = u_D \quad on \quad \Gamma_D \tag{1.1}$$

$$\nabla u \cdot v = g \text{ on } \Gamma_N$$

where $f \in L^2(\Omega)$, $g \in L^2(\Gamma_N)$, and $u_D \in H^1(\Omega) \cap C(\bar{\Omega})$ are all known functions. We consider both Neumann and Dirichlet boundary conditions, denoted respectively as Γ_N and Γ_D, in all our numerical experiments. Here the polygonal boundary, Γ of Ω is represented as the following disjoint union, $\Gamma = \Gamma_N \cup \Gamma_D$. By defining $p = \nabla u$, i.e., setting p equal to the flux of u, we are able to develop the following mixed formulation of (1.1):

$$-\nabla \cdot p = f \quad and \quad p = \nabla u \quad in \quad \Omega \tag{1.2}$$

The problem now involves solving for two unknowns, the displacement u and the flux, p. The flux, p lives in the space, $L^2(\Omega)^2$, since $u \in H^1$ implies that u and it's partial derivatives, u_x and u_y live in $L^2(\Omega)$. We now introduce the following function spaces,

$$H(div, \Omega) = \{q \in L^2(\Omega)^2 : \nabla \cdot q \in L^2(\Omega)\}$$

$$H_{g,N}(div, \Omega) = \{q \in H(div, \Omega) : q \cdot \upsilon = g \quad on \quad \Gamma_N\} \tag{1.3}$$

In order to construct a weak formulation of (1.2), we consider for any test function $q \in H_{0,N}(div, \Omega)$ which is zero on the Neumann boundary,

$$\int_\Omega p \cdot q dx = \int_\Omega \nabla u \cdot q dx = -\int_\Omega u \nabla \cdot q dx + \int_{\Gamma_D} u_D q \cdot \upsilon ds \tag{1.4}$$

Note that when integrating by parts, the Neumann boundary term, $\int_{\Gamma_N} gq \cdot \upsilon ds$ is equal to zero, since q is zero on Γ_N. Also, note that since the divergence of q, denoted as $\nabla \cdot q$, lives in $L^2(\Omega)$, it is natural to now consider finding the displacement, u such that $u \in L^2(\Omega)$.

Thus, the variational formulation of the mixed problem (1.2) is to find the unknowns, $p \in H_{g,N}(div, \Omega)$ and $u \in L^2(\Omega)$, such that for any $q \in H_{0,N}(div, \Omega)$ and $v \in L^2(\Omega)$

$$\int_\Omega p \cdot q dx + \int_\Omega u \nabla \cdot q dx = \int_{\Gamma_D} u_D q \cdot \upsilon ds \tag{1.5}$$

$$-\int_\Omega v \nabla \cdot p dx = \int_\Omega v f dx \tag{1.6}$$

with the same f,g, and u_D given as before. It is well known that the solution to (1.5)-(1.6) exists, is unique, and also is equivalent to solving the BVP given by (1.1)(cf., e.g.[3; Section 3.5]).

1.2 The Lagrange Multiplier Technique

An essential aspect in the discretization of a mixed finite element, is to find a $p_h \in H(div, \Omega)$. First, we introduce the following Raviart-Thomas functional space,

$RT_0(T) =$

$$\Big\{ q \in L^2(T) : \forall K \in T, \exists a \in \Re^2, \exists b \in \Re, \forall x \in K, q(x) = a + bx, and, \forall E \in \Sigma_\Omega, [q]_E \cdot \upsilon_E = 0 \Big\} \tag{1.7}$$

where T denotes the triangulation with elements, K and E denotes an edge in the set of all interior edges, Σ_Ω. For any two triangles, K_+ and K_- that share an interior edge, E we define the jump of a piecewise continuous function, q across E in the direction of υ_E (i.e., the unit normal vector of E), to be

$$[\mathrm{q}]_E \cdot \upsilon_E = \Big(q|_{K_+} - q|_{K_-} \Big) \cdot \upsilon_E$$

We assume that υ_E always points from K_+ to K_- (See Figure 1.1) and υ_E equals the exterior outward normal, υ when E is a boundary edge.

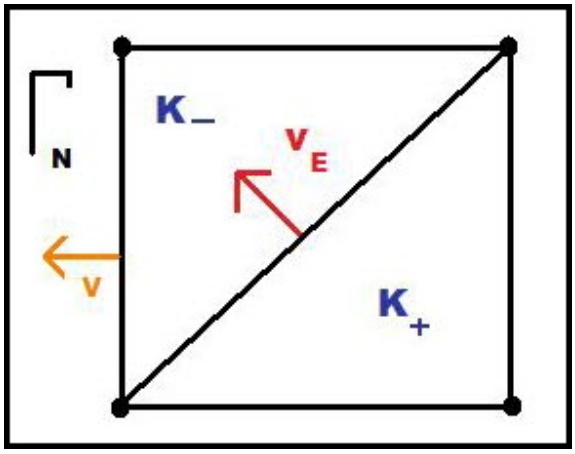

Figure 1.1: Consider an edge, E such that $E = \partial K_+ \cap \partial K_-$. Thus, the jump of a function q across E is in the direction of the normal unit vector, v_E. If E is on the boundary, then $v_E = v$, where v is the exterior unit normal.

In order for $RT_0(T) \subset H(div, \Omega)$, we need to satisfy the continuity condition of the normal components on the boundaries, i.e., the edges of each element in T. One

method is to apply Lagrange multipliers on both the interior edges and Neumann boundary edges. Since our task ahead is primarily a computational one, we refer the interested reader to [2] for the discrete formulation of (1.2) using the Lagrange Multiplier technique. The linear system that results from the discrete formulation via the Lagrange Multiplier technique, is in the following form:

$$Ax = \begin{pmatrix} B & C & D & F \\ C^T & 0 & 0 & 0 \\ D^T & 0 & 0 & 0 \\ F^T & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_\psi \\ x_u \\ x_{\lambda_M} \\ x_{\lambda N} \end{pmatrix} = \begin{pmatrix} b_D \\ b_f \\ 0 \\ b_g \end{pmatrix} \tag{1.8}$$

Here we are given b_D, b_g, and b_f which represent the Dirichlet and Neumann boundary conditions and volume forces, respectively. The unknowns, x_ψ are the components of p_h in the direction of the unit normal vectors, υ_E. Since we are seeking a $u_h \in L^2(\Omega)$, we are not guaranteed that the derivatives of u_h are also $L^2(\Omega)$ functions. Hence, we consider the unknowns x_u in (1.8) to be the T-piecewise constant approximations to the exact solution, u.

As a result of applying the Lagrange multipliers to satisfy the continuity condition, we must now solve for the unknowns, x_{λ_M} and x_{λ_N}. By the construction of $RT_0(T) \subset H(div, \Omega)$, the jumps in the flux, p_h across interior edges in the direction of the normal components, υ_E are all equal to zero. The unknowns, x_{λ_M} found in (1.8) are the Lagrange multipliers corresponding to the restrictions on each interior edge. Finally, x_{λ_N} solves for the unknown Lagrange multipliers on the Neumann boundary, with a given stress represented by the function, g. For the structures of the matrices found in (1.8) we refer the reader to [2]. However, we conclude this section with two important remarks on the computation of the linear system given by the Lagrange Multiplier technique.

An obvious disadvantage of mixed finite element methods compared to the standard finite element method is that the linear system, $Ax = b$ is larger and thus more costly to solve. For instance, the matrix B in (1.8), is a square matrix of size three times the number of elements in the triangulation.

Consider an alternative mixed finite element method, which satisfies the continuity condition of the normal components of p_h by creating an edge basis of the space, $RT_0(T)$ (cf., e.g.[2; Section 4]). The linear system of such a method is in the form,

$$\begin{pmatrix} \tilde{B} & \tilde{C} \\ \tilde{C}^T & 0 \end{pmatrix} \begin{pmatrix} x_\psi \\ x_u \end{pmatrix} = \begin{pmatrix} b_D \\ b_f \end{pmatrix} \tag{1.9}$$

An advantage of the Lagrange Multiplier technique compared to the above method, is that although the linear system of (1.8) is larger than that of (1.9), the matrices B and C consist of 3×3 block matrices. Computationally, it is more efficient to solve systems with block matrices, especially when computing inverses.

Chapter 2

An Example Problem on a Non-Convex Domain

2.1 The Problem

We now use the Lagrange multiplier technique and it's Matlab realization, $LMmfem.m$, both given in [2], to compute the approximate flux and displacement (p_h, u_h) in a model problem of the form (1.1):

$$-\Delta u = 0 \text{ in } \Omega = (-1,1) \times (-1,1) \cap ([0,1] \times [-1,0])^C$$

$$u = 0 \quad on \quad \Gamma_D = \{0\} \times [-1,0] \cap [0,1] \times \{0\} \tag{2.1}$$

$$\nabla \mathrm{u} \cdot \upsilon = g(r,\theta) \; on \; \Gamma_N = (\Gamma \cap \Gamma_D)^C$$

where $g(r,\theta) = \frac{2}{3} r^{-1/3}(-sin(\frac{\theta}{3}), cos(\frac{\theta}{3})) \cdot n$, in polar coordinates (r,θ). The exact solution of (2.1) in polar coordinates is given by,

$$u(r,\theta) = r^{\frac{2}{3}} sin(\frac{2}{3}\theta) \tag{2.2}$$

The computed solution for u_h over an uniform mesh size of $h = \frac{1}{20}$, using $LMmfem.m$ is given in Figure 2.

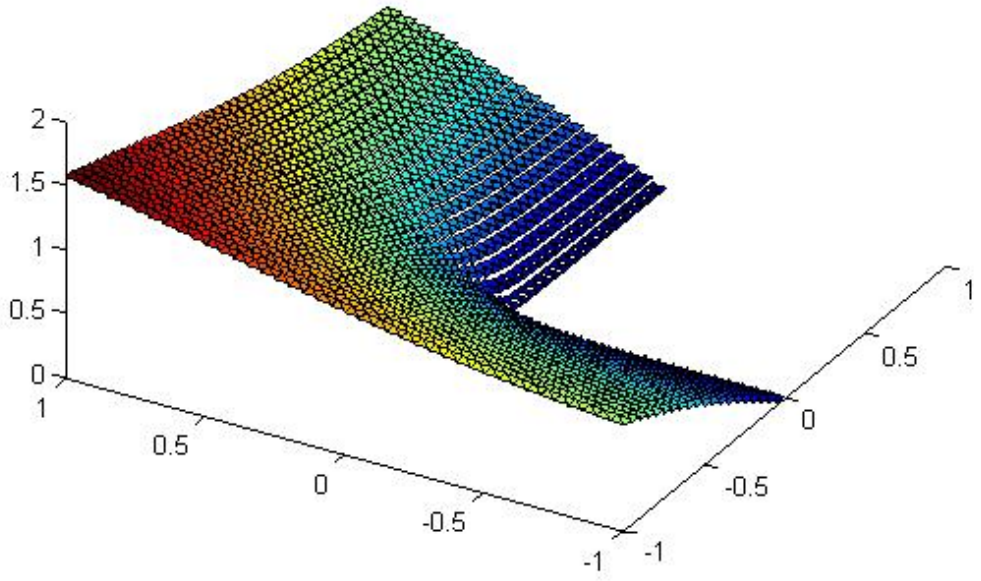

Figure 2.1: The discrete solution, u_h to (2.1) with a uniform step size of $h = 1/20$

Note that $\frac{\partial u}{\partial r}(r, \theta)$ has a singularity at the origin, i.e., when $r = 0$. Since $u = 0$ on Γ_D, we can expect that the computed flux, p_h will not be accurate for non-boundary nodes close to the origin. Also, the L-shape of the domain, Ω forces a singularity at the origin, since Ω is no longer convex. For instance, Figure 2.2 depicts the instabilities of the computed flux, p_h in the x-direction for elements close to the origin.

The reader can clearly see that solving the linear system given in (1.8) can be very costly when dealing with uniform mesh sizes. For instance, A in (1.8), is a 3320×3320 matrix for a uniform mesh size of $\frac{1}{10}$. The size of the system

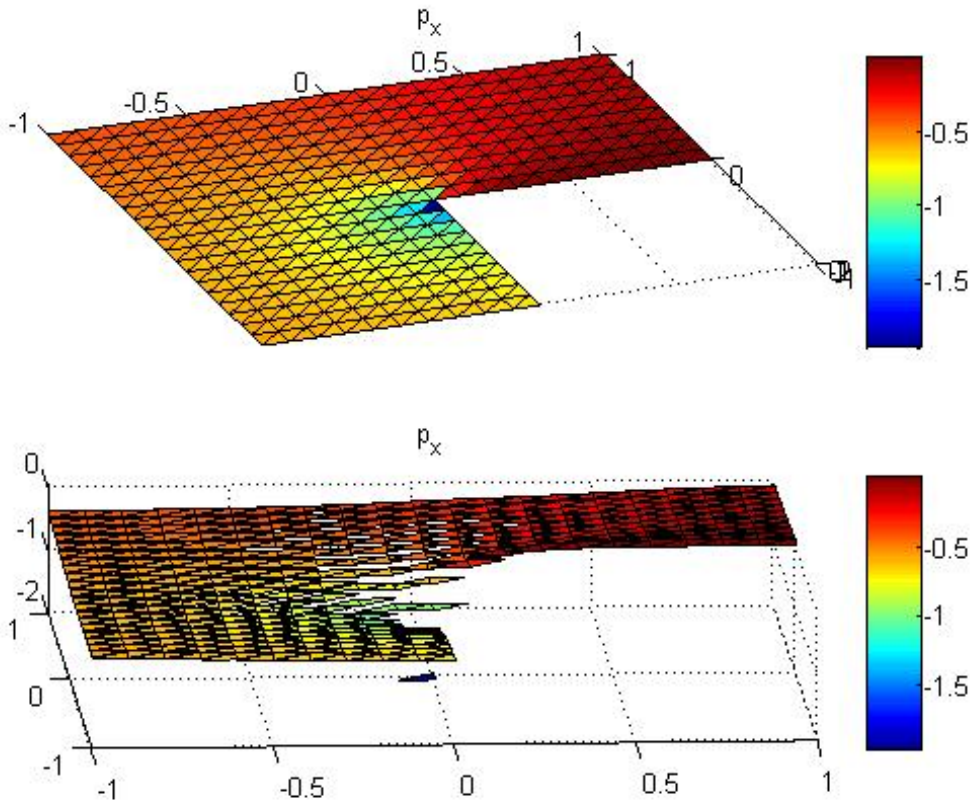

Figure 2.2: The discrete flux in the x direction for an uniform mesh size of $h = 1/10$. The figure shows that the flux, p_h varies greatly for elements close to the origin. This results from the singularity of the exact solution around the origin and the non-convexity of the domain, Ω

quickly becomes an issue and renders any analysis of step sizes greater than $\frac{1}{20}$ as an impractical task.

2.2 An Error Analysis in the $L^2(\Omega)$ Norm

The $L^2(\Omega)$ norm of the error between the discrete and exact solutions, u_h and u is given by,

$$\|u - u_h\|^2_{L^2(\Omega)} = \int_\Omega |u - u_h|^2 \, dx = \sum_{K \in T} \int_K |u - u_h|^2 \, dx \tag{2.3}$$

where K denotes an element (triangle) in the triangulation, T of Ω. Since u_h is piecewise constant on each triangle K, we approximate the exact solution, u over the triangle by computing u at the barycenter of K. See the Appendix for the Matlab realization of (2.3).

As a result of $u \notin H^2(\Omega)$, we expect that the error will no longer decrease by a factor of four when halving the mesh size (cf., e.g.[8; Theorem 5.4]). That is, we can expect the following ratio of errors,

$$\left(\frac{E_1}{E_2}\right) = \frac{\|u - u_h\|_{L_2(\Omega)}}{\left\|u - u_{h/2}\right\|_{L_2(\Omega)}} \tag{2.4}$$

to be less than four, due to the singularity of the exact solution u around the origin. For instance, in our numerical experiments, we have for uniform step sizes, $h = \frac{1}{10}$ and $h = \frac{1}{20}$,

$$\frac{\|u - u_h\|_{L_2(\Omega)}}{\left\|u - u_{h/2}\right\|_{L_2(\Omega)}} = \left(\frac{0.01071900649988}{0.00424245296767}\right) \approx 2.52660585316389 \tag{2.5}$$

Chapter 3

A Posteriori Estimator for the Error in the Flux

We give a brief discussion of the a posteriori error control of the computed flux, p_h and the unknown, exact flux, p of problem (2.1) in the $L^2(\Omega)$ norm. Since our adaptive method, will hinge on the choice of the error estimator, the interested reader should refer to [2] and [4] for an in-depth discussion.

We begin by introducing the following functional space, $S_D^1(T) = P_1(T) \cup C(\Omega)$, where T denotes the triangulation of Ω and $P_1(T)$ denotes the functional space of all piecewise linear functions over T. Also, assume that functions in $S_D^1(T)$ satisfy homogeneous Dirichlet boundary conditions. Hence, for any test function, $v_h \in S_D^1(T)$ we assume that the following Galerkin property holds,

$$\int_\Omega p_h \cdot \nabla v_h dx = \int_\Omega f v_h dx \tag{3.1}$$

The problem of computing $\|p - p_h\|_{L^2(\Omega)}$ now becomes: Seek a $q_h \in S_D^1(T)^2$ such that q_h is a smoother approximation to p_h than the unknown, exact flux, p. Hence, the goal is to approximate $\|p - p_h\|_{L^2(\Omega)}$ by constructing an operator, A that maps

the computed flux, p_h to the test function, q_h, i.e., $Ap_h = q_h$. In the paper [2], the authors defines the operator, A to be the composition of an averaging operator, M_z and an operator, π_z that computes the orthogonal projection of a vector in $\Re^2$ onto an affine subspace of $\Re^2$.

Since $p_h \in P_1(T)^2$, we have that for any node z in the mesh, the operator, $M_z : P_1(T)^2 \to \Re^2$ computes the average of p_h over the patch of z. The patch of a node z, denoted as ω_z, is defined to be the interior of the union of all triangles that contain the node, z. Hence, $M_z(p_h)$ can be defined as,

$$M_z(p_h) = \frac{1}{|\omega|} \int_{\omega_z} p_h dx \tag{3.2}$$

where $|\omega|$ denotes the area of the patch, ω_z. Once the average is computed for a given node, z we then proceed to take the orthogonal projection, π_z of $M_z(p_h)$ onto an affine subspace, X of $\Re^2$ such that,

$X_z =$

$$\{a \in \Re^2 : \forall E \in E_z \cap E_N, g(z) = a \cdot \upsilon_E \; and, \forall E \in E_z \cap E_D, \nabla_E u_D(z) = (a)_E\}$$

where E denotes an edge in the mesh and $\nabla_E u_D(z)$ denotes the tangential derivative along E. The role of π_z is to satisfy the Neumann and Dirichlet boundary conditions for the discrete flux, p_h by mapping to the affine space, X. Hence, if we define the functional space, $S^1(T) = span\{\phi_z : z \in N\}$, where N is the set of all nodes in the triangulation and ϕ_z are nodal basis functions, we can represent any $q_h \in S^1(T)^2$ as,

$$Ap_h = \sum_{z \in N} A_z(p_h|_{\omega_z})\phi_z \tag{3.3}$$

Here we have that, for any node z in the mesh, $A_z = \pi_z \circ M_z$.

Now that we have constructed the operator, A which maps the computed flux, p_h to a piecewise continuous linear function, q_h we can compute the global and local

error estimators given by,

$$\eta^2 = \|p_h - Ap_h\|^2_{L^2(\Omega)} \qquad and \qquad \eta^2_K = \|p_h - Ap_h\|^2_{L^2(K)} \tag{3.4}$$

Bahriawati and Carstensen showed that the error in the exact flux, $\|p - p_h\|^2_{L^2(\Omega)}$ can be bounded by $\eta = \|p_h - Ap_h\|_{L^2(\Omega)}$ that is,

$$\tilde{C}\eta - h.o.t \leq \|p - p_h\|^2_{L^2(\Omega)} \leq C\eta + h.o.t \tag{3.5}$$

where $\tilde{C}$ and C are constants that depend only on the triangulation, T and $h.o.t$ denotes higher order terms (cf., e.g.[2; Theorem 8.1]). This result is essential for adaptivity in the sense that we can now control the error in the flux by estimating locally as well as globally. Also, in practice the exact flux is usually unknown and therefore one must use an a posteriori error estimator to approximate $\|p - p_h\|^2_{L^2(\Omega)}$. In all our numerical experiments, we use the m-file, *Aposteriori.m* given in [2] to compute the error estimators η^2_K for all $K \in T$.

Using the software, *Aposteriori.m* to compute the global error estimate, η for uniform mesh sizes $h = \frac{1}{10}$ and $h = \frac{1}{20}$, we found the ratio of the error estimates is approximately $\frac{3}{2}$ that is,

$$\left(\frac{\eta}{\tilde{\eta}}\right) = \frac{\|p_h - Ap_h\|_{L_2(\Omega)}}{\left\|p_{h/2} - Ap_{h/2}\right\|_{L_2(\Omega)}} = \left(\frac{0.15376438745761}{0.09726532431751}\right) \approx 1.58087569785576 \tag{3.6}$$

Chapter 4

An Adaptive Method

As discussed in the previous section, the example given in (2.1) involves a singularity at the origin which affects the solutions of both the flux and displacement for nodes located around the origin. Also, in order to avoid solving expensive linear systems, such as the Lagrange Multiplier technique given in (1.8), one must consider utilizing an adaptive mesh as opposed to a uniform mesh.

In the subsequent sections, we introduce a simple adaptive method to help control the error in the computed flux, p_h close to the origin. All numerical experiments use the mfile, $LMmfem.m$ given in [2], to solve for both p_h and u_h and also to compute η^2 and η_K^2 by calling $Aposteriori.m$ as a function in Matlab. Depending on the accuracy of the solution, we then use the m-files, $bisection.m$, $label.m$, and $getmesh.m$ provided in [6], to refine the mesh according to the given tolerance, tol and marking parameter, θ. The interested reader should refer to [5],[6], and [7] for a brief background in adaptivity, numerical demos, and adaptive software.

4.1 The Marking Strategy

In this section, we discuss a method to identify or *mark* certain elements, K in the triangulation where the error in the computed flux, p_h is significant. However, before we can consider the marking process, we must first compute p_h over a given or initial triangulation of Ω. Once, we compute p_h we then use the a posteriori error estimator described in Chapter 3 to compute the global and local errors, denoted respectively as η and η_K such that,

$$\eta^2 = \|p_h - Ap_h\|^2_{L^2(\Omega)} = \sum_{K \in T} \eta_K^2 = \sum_{K \in T} \|p_h - Ap_h\|^2_{L^2(K)} \tag{4.1}$$

Initially, the user must decide the desired accuracy or tolerance, denoted as tol, of the computed flux. After solving the problem and estimating both the local and global error estimates, we check to see if $\eta > tol$. If $\eta \leq tol$, then we are done, since the required accuracy in the computed solution has been achieved. However, if $\eta > tol$ then the problem becomes to select a subset of the triangulation, T which contains the triangles where the error in the flux is relatively large. In the case of (2.1), we expect the triangles close to the origin to be marked for refinement, since this area is where the error in p_h will be the most significant. We introduce the following marking strategy provided by [5] and [6]: Given an initial parameter, θ where $0 < \theta < 1$, find a subset $M \subset T$ such that

$$\sum_{K \in M} \eta_K^2 \geq \theta \eta^2 \tag{4.2}$$

4.2 Conformity of the Triangulation and Elements

A natural question arises from the previous discussion. How does one mark a triangle for refinement and still maintain the overall data structure of the problem and, more importantly, maintain the conformity of the triangulation? Note that in a conforming finite element triangulation, the intersection of any two triangles, K_1 and K_2, is either empty, a node, or an edge.

A finite element, K in the triangulation has a non-conforming edge, E on its boundary, ∂K, if E has a node between its endpoints. Such a node is referred to as a *hanging node*. The question now becomes: How to ensure the non-existence of hanging nodes during the refinement process and therefore preserve the overall conformity of the triangulation? In order to answer this question, we must first introduce a method for refining marked elements.

4.3 Mesh Refinement

In this section, we consider the refinement method, the *newest vertex bisection method*, and it's Matlab implementation, *bisection.m* given in [5] and [6]. First we introduce an important aspect of the data structure, *element.dat* which stores each element with it's corresponding vertices (see [1] and [2] for details on data structures). We must identify one of the vertices of each triangle as the peak of the triangle. In [6], the authors identify the peak of K by labeling it as the first vertex entry in the data file *element.dat*.

Once, we have identified the subset of marked elements we can now refine each one by bisecting the edge opposite (also known as the *base* of K) and connecting the segment to the peak. As a result, the triangle K is now split into two triangles K_1 and K_2. Consequently, we now have an additional node $\tilde{z}$, i.e., the midpoint of

the base of K, which must be added to the data structure *coordinate.dat*. Also, we must add one more element to the array *element.dat* and update the original vertex entries for K with the new vertices of either K_1 or K_2. Note that in order to be able to repeat the refinement stage, we must now reassign the peaks of K_1 and K_2 to be the new node, $\tilde{z}$.

Hence, we now return to the marking stage previously discussed in Section 4.1. Now, not only do we consider marking elements where the error is significant, but also marking elements with non-conforming edges. For instance, consider the triangle(s) K where the error in the flux is the maximum over all triangles in the mesh. Clearly, if the global error estimate, η is greater than the desired tolerance, element K will be marked for refinement. Thus, the base of K will be bisected and the midpoint of the base, $\tilde{z}$ will become an additional node in the triangulation. However, if the neighboring element, $\tilde{K}$ which shares the base with K is not refined also, then $\tilde{z}$ will become a hanging node. Hence, while marking elements and identifying their base edges, we must also mark the neighboring element that shares the base edge for refinement also. Note that the resolution of hanging nodes is addressed during the marking process. Thus, the new triangulation is already conforming, prior to the refinement stage.

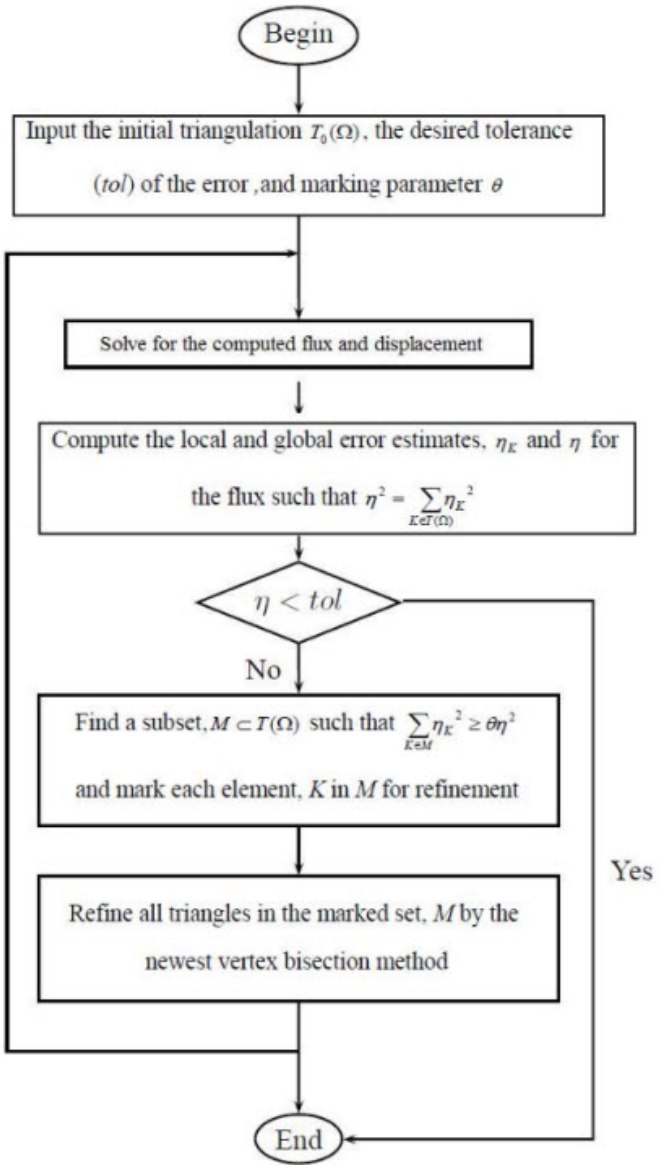

Figure 4.1: An overview of the adaptive process

Chapter 5

Numerical Experiments

We now consider solving the model problem, given by (2.1) adaptively, using the method previously discussed and as shown in Figure 4.1. See the Appendix for the Matlab script, *adaptfem* which is used in the following analysis to compute the solutions of (2.1) adaptively.

First, we begin our analysis by considering a very coarse initial mesh, T_0 such that the uniform mesh size, h is $h = 1/2$. After solving the initial problem and computing the local error estimators, η_K^2 for all $K \in T_0$, the two elements with the largest errors in flux are identified to be triangles 7 and 13 (See Figure 5.1). Both elements have a local error estimate of $\eta_K^2 = 0.0540$. If we choose a marking parameter, θ such that $\theta = 0.3$, elements K_7 and K_{13} will be the only elements marked for refinement based on error reduction. This follows from the fact that the sum, $\eta_7^2 + \eta_{13}^2 = 0.1080$ is greater than $\theta\eta^2 = 0.0605$. Hence, we have the minimum number of marked elements necessary to reduce the global error of the solution over the new mesh.

Also, as mentioned earlier in Sections 4.2 and 4.3 we must also mark the neighboring elements that share the base edges of K_7 and K_{13}. Hence, by also marking

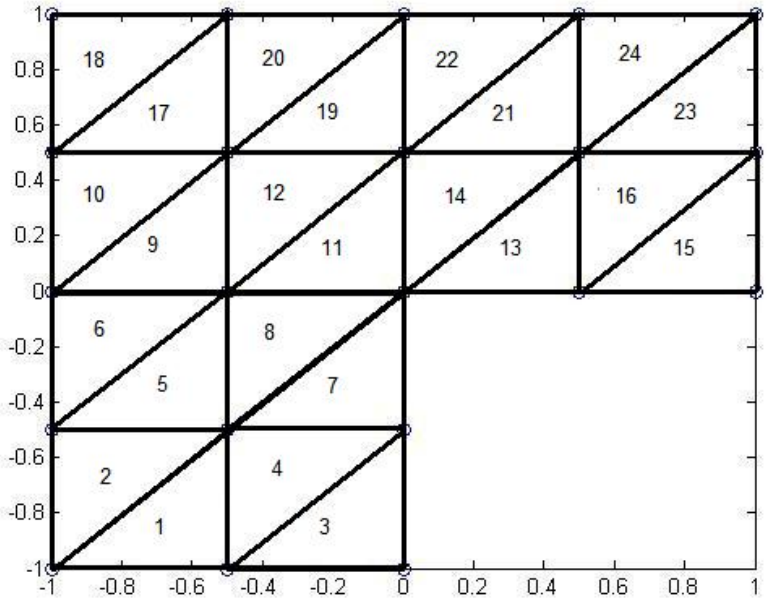

Figure 5.1: An initial mesh, T_0 with the corresponding element numbers. The data structure of each element and its corresponding vertices are stored in the data file, *element.dat*

triangles 8 and 14 for refinement, we maintain the overall conformity of the triangulation and prevent the occurance of hanging nodes. Note that the existence of a hanging node will disrupt the data structures given in [2], which are necessary for solving the discrete solution of (2.1). The computed solution, u_h after one refinement can be seen in Figure 5.2 along with the additional elements and nodes created in the refinement process.

The ability of the adaptive method to control the local error estimates, η_K^2 of the computed flux, p_h is clearly evident from the numerical experiments. The table in Figure 5.3 demonstrates just how significant adaptivity is in controlling large error estimates. For instance, consider an initial uniform mesh of size, h such that $h = \frac{1}{2}$. The initial global error, η is approximately 0.4490. Hence, if the desired tolerance is 0.1, we must apply the adaptive refinement process thirteen times in order to

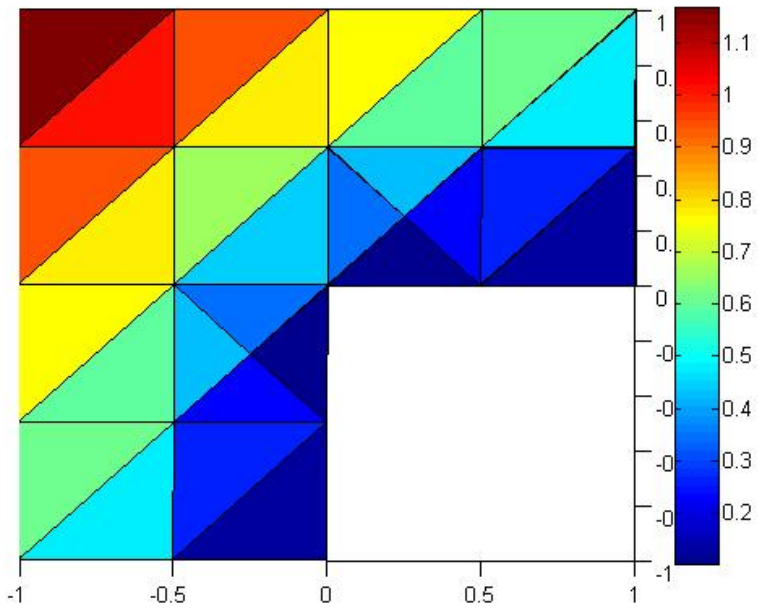

Figure 5.2: The computed solution of u_h for an initial mesh, T_0 of uniform size, $h = \frac{1}{2}$. The global error estimate, η was equal to 0.4490 after solving over T_0. After one refinement, we now have four new triangles, two new coordinates, and a global error, $\eta = 0.3794$

Initial uniform mesh size, *h*	# of refinements	Final # of elements	Final error, η
1/2	13	176	0.093577893468201
1/4	14	700	0.045595580731194
1/3	11	195	0.089906576040345
1/6	12	668	0.047008229267447

Figure 5.3: The number of refinements necessary for $\eta \leq tol$ for an initial uniform mesh, T_0 of size, h. The marking parameter, θ is equal to 0.35. The table compares the refinement process for T_0 of size h and $\frac{h}{2}$ and given tolerances, 0.1 and 0.05

control the error in the flux. After thirteen refinements, we now have a global error of p_h such that $\eta \approx 0.0936$. However, in order to obtain the same given accuracy in the computed solution of p_h without using an adaptive model, one must consider a mesh-size of $h = \frac{1}{20}$. Recall from Chapter 3 that the global error, η for a uniform mesh of size, $h = \frac{1}{20}$ is approximately 0.0973. Hence, we can achieve better accuracy and avoid costly computations by starting with a coarse uniform mesh and refining where the local error estimates of p_h are significantly large. We conclude this section with two important remarks.

Marking parameter , Θ	# of refinements	# of base edges bisected	Final error, η
0.2	21	70	0.09994079
0.3	14	76	0.097030051
0.4	11	80	0.097153688
0.5	9	97	0.095361603

Figure 5.4: This table demonstrates the effect of the marking parameter, θ on the number of iterations and the total number of refined edges. Each simulation was done with an initial mesh size of $h = \frac{1}{2}$ and a given tolerance, $tol = 0.1$

Note that the total number of refinement iterations and marked edges depends on the choice of the marking parameter, θ. The results given in Figure 8, show that for smaller values of θ, fewer triangles are marked for refinement during a single loop. Thus, smaller inputs for θ will imply more refinement loops in order to achieve the given tolerance. However, a larger valued θ will produce fewer refinement loops while simultaneously increasing the number of marked edges per loop. Since more elements are being marked for refinement in a single step, we expect that the total number of edges refined should be greater than the those of a more precise input, i.e., for $\tilde{\theta} < \theta$.

A natural question one may ask is the following: How to compare the global error estimates, $\eta = \|p_h - Ap_h\|_{L^2(\Omega)}$ for two initial uniform meshes of size, h and $\frac{h}{2}$? Note that we can now consider the total cost in halving the step size of T_0 in terms of the total number of elements. For instance, Figure 5.3 shows that when considering an initial mesh, T_0 with $h = 1/4$, there are 3.9773 times more elements in the final mesh than for the refined mesh of initial size $h = 1/2$. Also, the results in Figure 5.3 demonstrate that starting with a finer mesh of size, $h/2$ requires an extra iteration to achieve the desired tolerance when compared to beginning with a coarser mesh of size h.

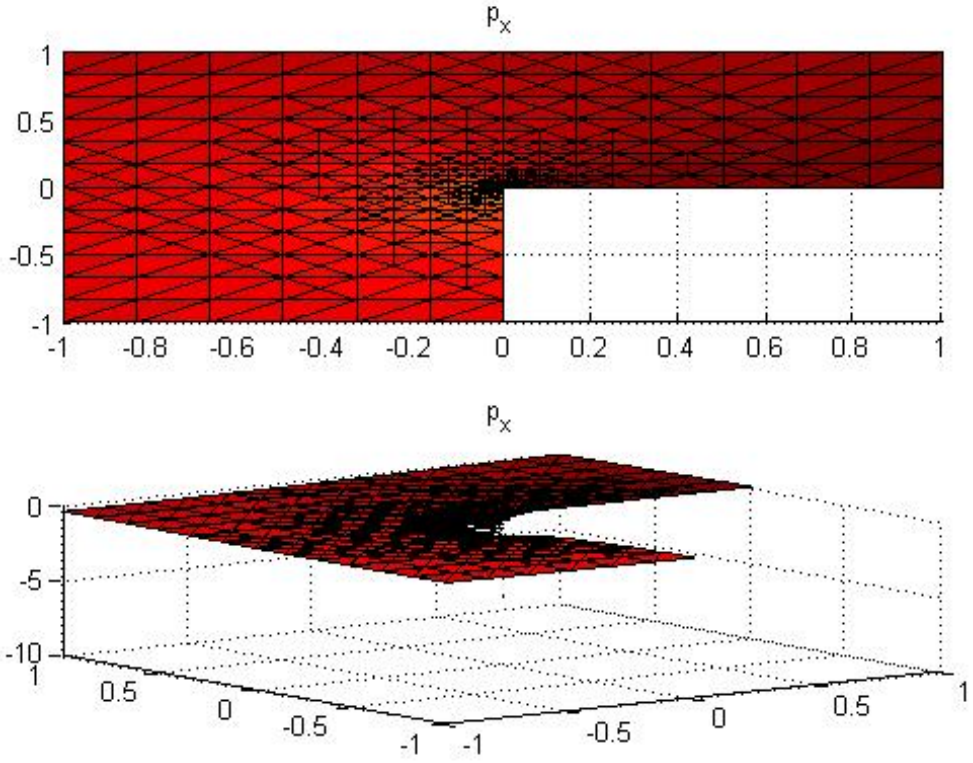

Figure 5.5: Computed flux in the x-direction for an initial mesh of uniform size, $h = 1/6$ with $\theta = .35$ and $tol = 0.05$. Note that in order to control the error estimates, η_K^2 the majority of the refinement is done on elements close to the origin. If we compare the results in this figure with those of Figure 2.2, we see a drastic improvement in the solution of p_h without the addition of expensive computations.

Appendix A

Matlab Code

A.1 An Error Analysis in the $L^2(\Omega)$ Norm

The following Matlab script computes the error in the $L^2(\Omega)$ norm,

```
U = size(element(:,1),1);
L2_norm = zeros(U,1);


for k = 1:U;
    L2_norm(k) = (1/2)*det([1 1 1;coordinate(element(k,:),:)'])*...
        (uexact(sum(coordinate(element(k,:),:))/3) - x(uinit(k)))^2;
end


R = sqrt(sum(L2_norm))
```

where the Matlab function, *uexact* calls the real solution given by (2.2) and $x(uinit)$ is the computed solution, u_h provided by $LMmfem.m$.

A.2 The Adaptive Program used in Chapter 5

The main program, *adaptfem* given below, is used in all the numerical experiments conducted in Chapter 5. The script requires the functions, *biscection.m*, *label.m*, and *getmesh.m*, given in [5] and [6], to perform all the marking and refinement steps. Also, the code calls the mixed finite element solver, $LMmfem$ from [2], as the function, $solve(x)$ where x denotes the data files of the current mesh. The outputs of $solve(x)$ are the local and global error estimates, η_K^2 and η computed by the script, *Aposteriori.m* which is also given in [2].

```
%Load initial mesh
    load coordinate.dat;    load element.dat;
    load dirichlet.dat;     load Neumann.dat;
%Initial solution of the problem via the LMmfem
    [Eta_global, Eta_local] = solve(coordinate,element,dirichlet,Neumann);
%Displaying of the initial error
    disp('Initial Global Error Estimate for the Computed Flux')
    disp(Eta_global)
    disp(Eta_local);
%User input
    tol = input('Input the desired tolerance, tol such that,0 < tol < 1?');
    theta = input('Input the marking parameter theta such that,0 < theta < 1?');
refinement = 0;
%Refinement process and updated solution
    while Eta_global >= tol;
        refinement = refinement + 1     %counter
        current_mesh = getmesh(coordinate,element,dirichlet,Neumann);
```

```
        new_mesh = bisection(current_mesh,Eta_local,theta);
              coordinate = new_mesh.node;   element = new_mesh.elem;
              dirichlet = new_mesh.Dirichlet;   Neumann = new_mesh.Neumann;
    [Eta_global,Eta_local] = solve(coordinate,element,dirichlet,Neumann);
    Eta_global
end
```

Bibliography

[1] Alberty, J., Carstensen, C., Funken, S.A., *Remarks around 50 lines of Matlab: short finite element implementation*, Numerical Algorithms 20 (1999), no. 2-3, pp. 117-137

[2] Bahriawati, C., Carstensen, C., *Three Matlab Implementations of the Lowest-Order Raviart-Thomas MFEM with A Posteriori Error Control*, Computational Methods in Applied Mathematics (2005), no. 4, pp. 333-361

[3] Braess, D., *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics*, Cambridge University Press, Cambridge, UK, 2nd edition, 2001

[4] Carstensen, C., *All First-Order Averaging Techniques for A Posteriori Finite Element Error Control on Unstructured Grids are Efficient and Reliable*, Mathematics of Computation, (2003), Vol. 73, no. 247, pp. 1153-1165

[5] Chen, L., *Short Bisection Implementation in Matlab*, (Submitted to International Workshop on Computational Science and its Education), 2006

[6] Chen, L., Zhang, C-S., *AFEM.at.matlab: A Matlab Package of Adaptive Finite Element Methods*, University of Maryland, 2006

[7] Chen, L., Zhang, C-S., *A Coarsening Algorithm and Multilevel Methods on Adaptive grids by Newest Vertex Bisection*, (Submitted), 2007

[8] Larsson, S., Thomee, V., *Partial Differential Equations with Numerical Methods*, Springer-Verlag, Berlin Heidelberg, 2003, 259 pages