UNIVERSITY OF CALIFORNIA, SAN DIEGO

p-Adaptive and Automatic hp-Adaptive Finite Element Methods for Elliptic Partial Differential Equations

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Mathematics

by

Hieu Trung Nguyen

Committee in charge:

Professor Randolph E. Bank, Chair
Professor Michael Holst
Professor Julius Kuti
Professor Bo Li
Professor Michael Norman

2010

The dissertation of Hieu Trung Nguyen is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____
Chair

University of California, San Diego

2010

DEDICATION

To my dear parents and my loving wife

EPIGRAPH

Some mathematician, I believe, has said that true pleasure lies not in the discovery of truth, but in the search for it.

—Lev Nikolayevich Tolstoy

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This work would not have been completed without the help and support I have received along the way.

I owe my deepest gratitude to my advisor, Dr. Randolph Bank, for his insightful suggestions when I first started working on the project and for the invaluable encouragement and stimulating advices during the whole progress. I am grateful for his share of the PLTMG software package and for all the precious time he spent with me whenever I felt perplexed by the problems.

I am indebted to Dr. Michael Holst and Dr. Bo Li for the valuable comments and suggestions on my presentations of the progressing research during the past three years. They have helped me to see the problem from a much broader view.

I would like to thank Dr. Randolph Bank, Dr. Michael Holst, and Dr. Philip Gill, the directors of the Center for Computational Mathematics. As a member of the center, I have been provided with the access to numerous computing resources, such as the Apple iMac workstations in my offices, the display wall in conference room, and especially the cluster in the server room.

I am indebted to Dr. Yifeng Cui for the guidance and support throughout my work as a research assistant at San Diego Super Computer Center in the summer of 2009. The work was an opportunity for me to see how people from other fields use math and how math people can help them.

I want to thank the Vietnam Education Foundation for introducing me to the opportunity of studying in the U.S and for their financial support during my first two years.

My special thanks go to my parents who always love me and support me unconditionally.

And last but not least, I would like to thank my wife, Diep, for her love, patience and support throughout the years.

VITA

Education

2003	B. S., Major in Numerical Mathematics, Vietnam National University, Hanoi • Honors Thesis: *Shooting methods for two points boundary-value problems* • Advisor: Dr. Pham Ky Anh
2006	M. S. in Mathematics, University of California, San Diego
2010	Ph. D. in Mathematics, University of California, San Diego • Dissertation: *p-Adaptive and Fully Automatic hp-Adaptive Finite Element Methods* • Advisor: Dr. Randolph E. Bank

Work Experience

2003-2005	Junior Lecturer, Vietnam National University, Hanoi
2005-2010	Teaching Assistant, University of California, San Diego
2007-2010	Research Assistant, Center for Computational Mathematics
Jun-Sep, 2009	Research Assistant, San Diego Super Computer Center

PUBLICATIONS

R. E. Bank and H. T. Nguyen, *p and fully automatic hp adaptive finite element methods*, in preparation.

R. E. Bank and H. T. Nguyen, *Domain decomposition and hp-adaptive finite elements*, in Domain Decomposition Methods in Science and Engineering XIX, Lect. Notes Comput. Sci. Eng., Springer, to appear.

H. T. Nguyen, Yifeng Cui, Kim Olsen, Kwangyoon Lee, *Single CPU optimizations of SCEC AWP-Olsen application*, Poster, Southern California Earthquake Center Annual Meeting, 2009.

H. T. Nguyen, *Remark on the shooting methods for nonlinear two-point boundary-value problems*, Journal of Science, Vietnam National University, T. XIX, No 3, 2003.

ABSTRACT OF THE DISSERTATION

p-Adaptive and Automatic hp-Adaptive Finite Element Methods for Elliptic Partial Differential Equations

by

Hieu Trung Nguyen

Doctor of Philosophy in Mathematics

University of California, San Diego, 2010

Professor Randolph E. Bank, Chair

In this dissertation, we formulate and implement p-adaptive and hp-adaptive finite element methods to solve elliptic partial differential equations. The main idea of the work is to use elements of high degrees solely (p-adaptive) or in combination with elements of small size (hp-adaptive) to better capture the behavior of the solution. In implementing the idea, we deal with different aspects of building an adaptive finite element method, such as defining basis functions, developing algorithms for adaptive meshing procedure and formulating a posteriori error estimates and error indicators.

The basis functions used in this work are regular nodal basis functions and special basis functions defined for elements with one or more edges of higher degree (transition elements). It is proved that with our construction of these basis functions, the finite element space is well-defined and $\mathcal{C}^0$.

Several algorithms are developed for different scenarios of the adaptive meshing procedure, namely, p-refinement, p-unrefinement and hp-refinement. They all follow the 1-irregular rule and 2-neighbor rule motivated by [24]. These rules

help to limit the number of special cases and maintain the sparsity of the stiffness matrix, and thus to simplify the implementation and reduce the cost of calculation.

The work of formulating a posteriori error estimates and error indicators is the core of this dissertation. Our error estimates and error indicators are based on the derivative recovery technique proposed by Bank and Xu [27, 28, 29]. Using the information in formulating the error indicators, we define a *hp*-refinement indicator which can be utilized to decide whether a given element should be refined in h or in p. Numerical results show that the combination of the two indicators helps automatic *hp*-refinement to create optimal meshes that demonstrate exponential rate of convergence.

In this dissertation, we also consider *hp*-adaptive and domain decomposition when they are combined using the parallel adaptive meshing paradigm developed by Bank and Holst [18, 19]. Numerical experiments demonstrate that the paradigm scales up to at least 256 processors (maximum size of our experiments) and with nearly 200 millions degrees of freedom.

Chapter 1

Introduction

1.1 Model Problem

In this work, we consider the second order elliptic partial differential equation (PDE):

$$-\nabla \cdot \boldsymbol{a}(x, y, u, \nabla u) + f(x, y, u, \nabla u) = 0 \qquad \text{in } \Omega \tag{1.1}$$

with boundary conditions

$$u = g_2(x, y) \qquad \text{on } \partial\Omega_2 \tag{1.2a}$$

$$\boldsymbol{a}(x, y, u, \nabla u) \cdot \boldsymbol{n} = g_1(x, y, u) \qquad \text{on } \partial\Omega_1 \tag{1.2b}$$

$$u,\ \boldsymbol{a}(x, y, u, \nabla u) \cdot \boldsymbol{n} \quad \text{continuous} \qquad \text{on } \partial\Omega_0 \tag{1.2c}$$

Here $\Omega \in \mathbb{R}^2$ is a bounded domain; $\boldsymbol{n}$ is the unit normal vector; and $\boldsymbol{a} = (a_1, a_2)^t$ and $a_1,\ a_2,\ f,\ g_1,\ g_2$ are scalar functions.

Even though (1.1)-(1.2) is the model actually implemented, in this dissertation we sometimes consider a simpler linear model as the work is either independent of the PDE or easy to generalize to nonlinear problems.

1.2 p- and hp-Adaptive Finite Element Methods

Finite Element Methods (FEMs) have been employed in virtually every area of science and engineering that can make use of models of nature characterized by

partial differential equations. In FEMs, the domain is partitioned into convex subdomains, such as triangles, and the solution is approximated by piecewise smooth polynomials defined on the partition. In practice, most of the FEM packages are available in adaptive version in which the mesh is gradually built to adapt to the projected behavior of the approximate solution. Depending on the mesh adaptivity technique being utilized, adaptive FEMs can be categorized into three versions: h-version, p-version and hp-version. The h-version is the standard version. In the h-version, degree of elements is fixed (usually one or two) and better accuracy is achieved by properly refining the mesh. The p-version, in contrast, fixes the geometry of the mesh and achieves better accuracy by increasing degree of elements uniformly or selectively. The hp-version is the combination of the two.

Babuška and his collaborators introduced the p-version in [11] and, later, the hp-version in [2]. Since then, the analysis of a priori error of these versions has been studied extensively in the literature [11, 2, 36, 39, 40, 9, 10, 5, 3, 37]. These studies show that p-version has comparable rate of convergence with the standard version

$$\|e_p\|_{1,\Omega} \leq C(k,\epsilon) N^{-(k-1)/2+\epsilon} \|u\|_{k,\Omega}, \tag{1.3}$$

while hp-version can achieve exponential rate of convergence

$$\|e_{hp}\|_{1,\Omega} \leq C \exp(-bN^{1/3}). \tag{1.4}$$

Here e_p, e_{hp} are finite element errors; N is number of degree of freedom; and $C(k,\epsilon)$, C are independent of N. These estimates present great potential of p-version and hp-version. However, numerical experiments show that they hold only when size and degree of the elements in the mesh are properly chosen. This makes the mechanism guiding the adaptive meshing procedure the most important part of p- and hp-adaptive FEMs.

In most cases, the adaptive meshing procedure in FEMs is controlled by a posteriori error estimates. These are estimates calculated using information from the approximate solution. They can provide very reliable information on the accuracy of the solution as well as the distribution of the error among elements. Currently, there are several techniques for computing a posteriori errors in FEMs,

such as element residual methods, duality methods, dual weighted residual methods (see [26, 47, 7, 49]). These techniques were first introduced for classical finite element methods (the h-version), and then extended for hp-version (see [46, 42]). Even though they all demonstrate certain successes in providing efficient a posteriori error estimators, they have their own limitations. For example, element residual methods and subdomain-residual methods require special implementation for each problem class; and duality methods and dual weighted residual methods require one to solve another differential equation.

There has been a great desire for a posteriori error estimates for the p- and hp-version of adaptive FEMs that is reliable, independent of the PDE and easy to compute. Especially, we would want that these estimates can be used to formulate a fully automatic hp-version in which the decision whether to refine a given element in h or in p is made efficiently. This dissertation addresses the demand by formulating a posteriori error estimates using the derivative recovery technique proposed by Bank and Xu [27, 28].

To have complete p- and hp-adaptive methods, we also formulate finite element basis functions, develop different adaptive meshing algorithms, then implement them in PLTMG[1] and compare the performance with the standard h-adaptive methods using numerical experiments. In order to solve problems of large scale on parallel machine, the combination of hp-adaptive and domain decomposition methods is studied.

1.3 Contributions of this Dissertation

In chapter 2, we formulate the basis functions for our FEMs. Our work is a bit unconventional with the use of nodal basis functions, rather than a hierarchical family of functions. Here the nodal basis functions are defined using the concept of nodal points which are local degrees of freedom in each element. In order to allow elements of different degrees caused by p-adaptive meshing to exist in the same mesh, we introduce the special sets of basis functions for transition elements.

[1] PLTMG is a FEM package developed by Bank since 1976 [17].

It is shown that with our construction, the finite element space is well-defined and continuous. Moreover, we prove that for each element, its regular finite element space is contained in the transition finite element space. This is important to guarantee that when an element τ is converted to a transition state to adapt with the change in degree of its neighbor(s), the order of accuracy of the approximate solution on τ will not decrease.

In chapter 3, different algorithms for p- and hp-adaptive meshing, namely, p-refinement, p-unrefinement and hp-refinement, are developed. These algorithms are all designed with emphasis on increasing reliability, reducing computational cost and simplifying implementation. One of the highlights of the chapter is the generalization of 1-irregular rule and 2-neighbor rule for p- and hp-versions. These rules ensure that each element is in the support of a bounded number of basis functions. This helps to preserve the sparsity of the resulting system of linear equations as the mesh is refined. Furthermore, applying these rules also helps to limit the number of special cases and thus to simplify the implementation. In addition, we prove that when these rules are applied strictly, an element can be p-refined at most once before the whole problem is resolved. This increases the reliability of the adaptive meshing procedure as a way to approximate error of an element after a p-refinement is currently unavailable.

Chapter 4 is the most important one in which we study the derivative recovery technique and apply it to formulating a posteriori error estimates for the p- and hp-versions of FEMs. The results are extensions of the works of Bank, Xu and Zheng [27, 28, 29]. In Lemma 4.18, the basis functions for the local error space of an element of arbitrary degree p are defined and their coefficients in the error indicator are explicitly computed in Lemma 4.19. In addition, our empirical study shows that using the normalization constant appearing in defining our error indicator provides a simple and effective solution for an important practical problem for automatic hp-meshing. That is, how to decide whether it is better to refine a given element into several child elements (h-refinement), or to increase its degree (p-refinement). As a point of practical interest, we note that our error indicator and hp-refinement indicator are independent of the PDE and thus single implementation can be used

across a broad spectrum of problems.

In chapter 5, we briefly discuss the combination of *hp*-adaptive meshing and domain decomposition using the parallel adaptive meshing paradigm developed by Bank and Holst [18, 19]. Most of our work in this chapter is spent on regularization phase in which meshes from different processors are made conforming in both geometry (in h) and degree (in p). The work was complicated at first but is then simplified with the introduction of a special set of basis functions that allows transition elements to have an arbitrary number of transition edges of arbitrary degree.

In chapter 6 we solve various problems using different versions of adaptive FEMs. The numerical experiments in this chapter show that the p-version and the *hp*-version of FEMs are generally more effective than the standard version. One of the highlights of this chapter is the experiment of a model problem with singularities in the solution. In the experiment, not only is automatic *hp*-version able to recognize the regions of singularities automatically and dominate h-version/p-version in performance, but it also demonstrates exponential rate of convergence as predicted by theory. Another important result of this chapter is the experiment with the lake superior problem in which *hp*-adaptive meshing is used in conjunction with domain decomposition. The experiment shows that the convergence is stable and largely independent of the number of processors and number of degrees of freedom (the biggest run is with 256 processors and has nearly 200 millions degrees of freedom).

Chapter 2

Basis Functions

2.1 Nodal Points

Let Ω in $\mathbb{R}^2$ be the bounded domain of the partial differential equation we are working with. For simplicity of exposition, we assume that Ω is a polygon. Let $\mathcal{T}$ be a triangulation of Ω satisfying the conditions in Definition 3.1 and t be an element (triangle) in $\mathcal{T}$. To define the nodal basis functions associated with t, we begin with the definition of *nodal points*.

Definition 2.1. *Nodal points of an element (triangle) t of degree p are:*

(i) three vertex nodal points at the vertices

(ii) $p-1$ edge nodal points equally spaced in the interior of each edge

(iii) interior nodal points placed at the intersections of lines that are parallel to edges and connecting edge nodal points.

Nodal points of an element of degree p are sometimes referred to as nodal points of degree p. Note that linear elements ($p = 1$) have only vertex nodal points and quadratic elements ($p = 2$) have only vertex and edge nodal points. Figure 2.1 shows examples of nodal points for element of degree for $p = 1, \ldots, 9$.

Definition 2.1 above is a descriptive one. In PLTMG (see [17]), we adopt, for practical purposes, the following result using barycentric coordinates.

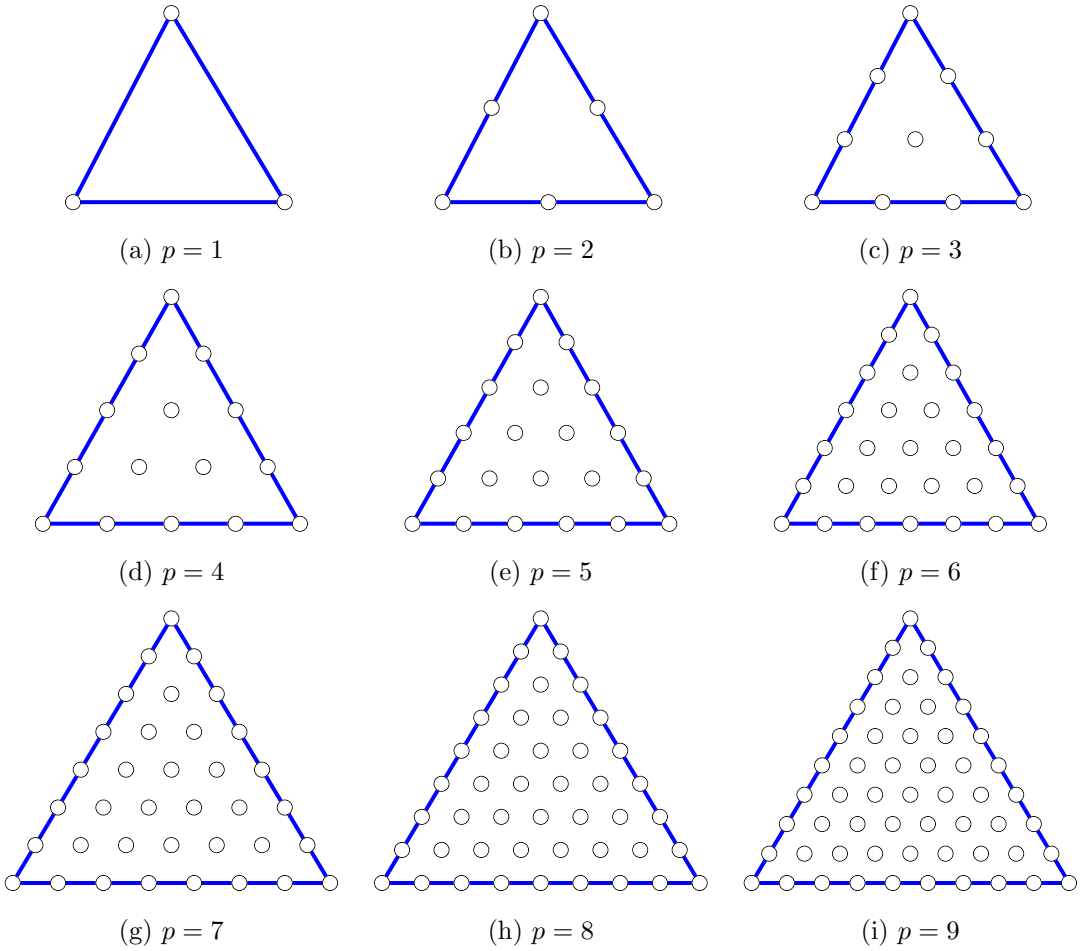


Figure 2.1: Nodal points of elements of degree p.

Proposition 2.2. *Nodal points of degree p are the points with barycentric coordinates $(\frac{i}{p}, \frac{j}{p}, \frac{k}{p})$, where i, j, k are nonnegative integers satisfying $i + j + k = p$.*

Proof. First, the three vertex nodal points are the points with one 1 coordinate and two zero coordinates $((1, 0, 0), (0, 1, 0)$ and $(0, 0, 1))$.

Second, the nodal points in the interior of edges are the ones with one and only one zero coordinate. It is straightforward to see that these points are equally spaced.

Finally, interior nodal points are the points with all three nonzero coordinates. The point with coordinates $(i/p, j/p, k/p)$, where i, j, k are all nonzero, is the intersection of the three lines $c_1 = i/p,\ c_2 = j/p$ and $c_3 = k/p$. Here c_i is the

ith barycentric coordinate and $c_i = c$ is the line consisting of all points that have ith barycentric coordinate equal c. Obviously, the line $c_1 = i/p$ is connecting two edge nodal point $(i/p, 0, (p-i)/p)$ and $(i/p, (p-i)/p, 0)$. Similar statements hold for $c_2 = j/p$ and $c_3 = k/p$. □

Proposition 2.3. *An element of degree p has exactly $N_p = \dfrac{(p+1)(p+2)}{2}$ nodal points.*

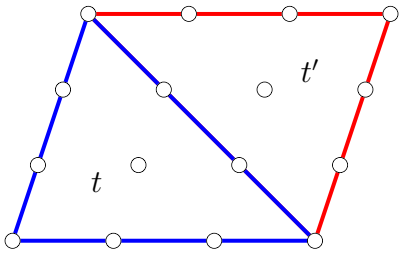


Figure 2.2: Nodal points of t and its reflection t'.

Proof. Let t' be the reflection image of t about one of its edges. Without loss of generality, take it to be edge two. The nodal points of t and t' together create a slanted grid with $p+1$ points on each edge as shown in Figure 2.2. The total number of points in the grid is $(p+1)^2$. Since t and t' have the same number of nodal points and $p+1$ points on their shared edge are counted only once, the number of nodal points of t is

$$\frac{(p+1)^2 + (p+1)}{2} = \frac{(p+1)(p+2)}{2}.$$

□

Corollary 2.4. $\dfrac{(p-1)(p-2)}{2}$ *is the number of interior nodal points of an element of degree p.*

Proof. Since there are three vertex nodal points and each edge has $p-1$ edge nodal points, the number of interior nodal point is

$$\frac{(p+1)(p+2)}{2} - 3 - 3(p-1) = \frac{(p-1)(p-2)}{2}$$

□

2.2 Nodal Basis Functions

Let $\mathcal{P}_p(t)$ be the space of polynomials of degree equal or less than p, restricted on element t. The canonical basis of $\mathcal{P}_p(t)$ is

$$\{1, x, y, xy, \ldots, x^{p-1}y, xy^{p-1}, x^p, y^p\}.$$

Remark 2.5. *The canonical basis of $\mathcal{P}_p(t)$ can be represented as $\{x^i y^j\}_{i,j\geq 0}^{i+j\leq p}$. Therefore the dimension of $\mathcal{P}_p(t)$ is*

$$\sum_{i,j\geq 0}^{i+j\leq p} 1 = \sum_{i=0}^{p}\sum_{j=0}^{i} 1 = \sum_{i=0}^{p}(i+1) = \frac{(p+1)(p+2)}{2} = N_p.$$

This basis is simple but is not convenient to incorporate in finite element methods. In the next few steps, we will prepare for the definition of another basis of $\mathcal{P}_p(t)$ which is usually used in practice.

Lemma 2.6. *Let P be a polynomial of degree $p \geq 1$ that vanishes on the straight line L defined by equation $L(x, y) = 0$. Then we can write $P = LQ$, where Q is a polynomial of degree $p - 1$.*

Proof. Make an affine change of coordinates to $(\hat{x}, y)$ such that $L(x, y) = \hat{x}$ (if $L(x, y) = y$ then no change of coordinates is necessary). Let

$$P(\hat{x}, y) = \sum_{i=0}^{p}\sum_{j=0}^{i} c_{ij}\hat{x}^j y^{i-j}. \tag{2.1}$$

In the new coordinate system, the equation of L is $\hat{x} = 0$. Since $P|_L \equiv 0$, plugging $\hat{x} = 0$ into equation (2.1) we have $\sum_{i=0}^{p} c_{i0}y^i \equiv 0$. This implies that $c_{i0} = 0$ for all $i = 0, \ldots p$. Therefore,

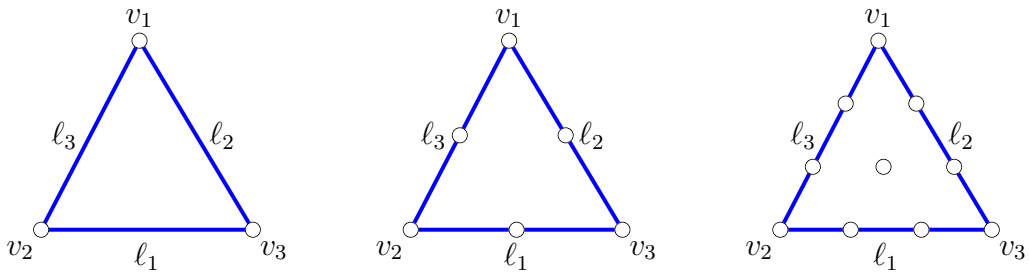
$$\begin{aligned} P(\hat{x}, y) &= \sum_{i=1}^{p}\sum_{j=1}^{i} c_{ij}\hat{x}^j y^{i-j} \\ &= \hat{x}\sum_{i=0}^{p-1}\sum_{j=0}^{i} \hat{x}^j y^{i-j} \\ &= LQ. \end{aligned}$$

Clearly, Q is a polynomial of degree $p - 1$. □

Lemma 2.7. *If $P \in \mathcal{P}_p(t)$ vanishes at all of the nodal points of degree p of t, then P is the zero polynomial.*

Proof. The proof is by induction on p. Denote v_1, v_2, v_3 and ℓ_1, ℓ_2, ℓ_3 respectively be the vertices and edges of t as shown in Figure 2.3. In addition, let L_1, L_2, L_3 be the linear functions that define the lines, on which lie the edges ℓ_1, ℓ_2, ℓ_3.

For $p = 1$, P is a linear polynomial that vanishes at two different points v_2 and v_3 of l_1. Therefore $P|_{\ell_1} \equiv 0$. By Lemma 2.6, $P = cL_1$, where c is a constant (polynomial of degree 0). On the other hand, P equals zero at v_1 and L_1 is nonzero at v_1. This implies that $c = 0$. Hence $P \equiv 0$.



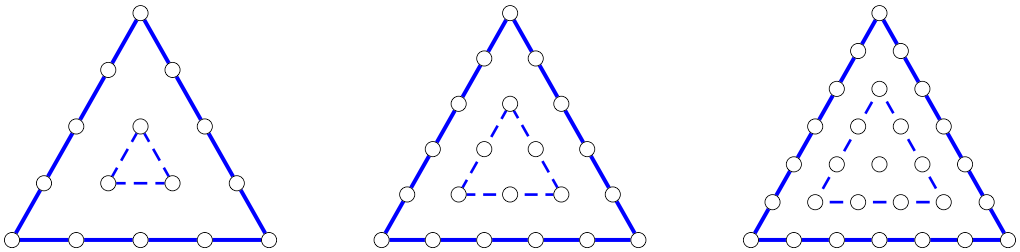
(a) $p = 1$ (b) $p = 2$ (c) $p = 3$

Figure 2.3: Nodal points of elements of degree $p = 1, 2, 3$.

For $p = 2$, P is a quadratic polynomial that vanishes at three different nodal points on ℓ_1. Therefore $P|_{\ell_1} \equiv 0$. Again by Lemma 2.6, $P = L_1Q$, where Q is a linear function (polynomial of degree 1). Since L_1 is nonzero along ℓ_2 except at v_3, Q needs to be zero at least at two points on ℓ_2: v_1 and midpoint of l_2. Hence $Q = cL_2$, where c is a constant. Consequently $P = cL_1L_2$. On the other hand, P needs to be zero at the midpoint of ℓ_3 also. This implies that $c = 0$. Therefore $P \equiv 0$.

For $p = 3$, using a similar argument, we have $P = cL_1L_2L_3$, where c is a constant. In order for P to be zero at the interior nodal point of degree 3, c needs to be 0. Hence $P \equiv 0$.

Assume that the lemma holds for polynomials of degree up to p. For $P \in \mathcal{P}_{p+1}(t)$, again by a similar argument for $p = 1$, 2, 3, we know that $P = L_1L_2L_3Q$, where Q is a polynomial of degree $p - 3$ or less. Furthermore, Q vanishes at all of



(a) $p = 4$ (b) $p = 5$ (c) $p = 6$

Figure 2.4: Nodal points of elements of degree $p = 4, 5, 6$.

the interior nodal points of t. These points can be seen as nodal points of degree $p-3$ of triangle t' laid inside t. Examples for $p = 4, 5, 6$ are illustrated in Figure 2.4. By induction hypothesis, Q is the zero polynomial. Consequently, P is the zero polynomial. □

Remark 2.8. *Even though Lemma 2.7 is stated for $P \in \mathcal{P}_p(t)$, the result still holds for P defined on the whole $\mathbb{R}^2$.*

Now we define nodal basis functions for element t.

Theorem 2.9. *Consider a way of labeling the nodal points of t, an element of degree p, from n_1 to n_{N_p}. Let ϕ_l be the polynomial of degree p that equals 1 at the nodal point n_l and equals 0 at all other nodal points of t. Then $\{\phi_l\}_{l=1}^{N_p}$ is a basis of $\mathcal{P}_p(t)$. This basis is called the nodal basis of t.*

Proof. We first verify that ϕ_l are well defined by showing their existence and uniqueness. Assume $(\hat{i}/p, \hat{j}/p, \hat{k}/p)$ is the barycentric coordinates of $n_{\hat{l}}$. Let P be the polynomial of degree p defined as follows

$$P = \prod_{i=0}^{\hat{i}-1}(c_1 - \frac{i}{p}) \prod_{j=0}^{\hat{j}-1}(c_2 - \frac{j}{p}) \prod_{k=0}^{\hat{k}-1}(c_3 - \frac{k}{p}).$$

Clearly, P is of degree p and is nonzero at $n_{\hat{l}}$. Now we consider a different nodal point n_l which is also of degree p and has barycentric coordinates $(i/p, j/p, k/p)$. Since $i + j + k = p = \hat{i} + \hat{j} + \hat{k}$, either $i < \hat{i}$ or $j < \hat{j}$ or $k < \hat{k}$. Without loss of generality, we can assume that $i < \hat{i}$. Then the formula of P contains the factor

$c_1 - i/p$. This implies that P equals zero at n_l. Therefore P is of degree p and vanishes at all of the nodal points of degree p except for $n_{\hat{l}}$. Consequently, $\phi_{\hat{l}}$ exists and can be written as $k_{\hat{l}}P$, where $k_{\hat{l}}$ is chosen so that $\phi_{\hat{l}}$ equals 1 at $n_{\hat{l}}$.

The uniqueness of ϕ_l comes from Lemma 2.7. Assume that ϕ_l' is another polynomial of degree p that equals 1 at n_l and zero at all other nodal points of degree p. Then $P = \phi_l - \phi_l'$ is a polynomial of degree p (or less) and P vanishes at all of the nodal points of degree p of t. By Lemma 2.7, $P \equiv 0$. Hence $\phi_l \equiv \phi_l'$.

It remains to show that $\{\phi_l\}_{l=1}^{N_p}$ is actually a basis of $\mathcal{P}_p(t)$. Assume that the zero polynomial can be written as a linear combination of ϕ_l, i.e. $\sum_{l=1}^{N_p} \alpha_l \phi_l \equiv 0$. Evaluate both sides of this identity at nodal points of t, we have $\alpha_l = 0$ for all l. This implies that $\{\phi_l\}_{l=1}^{N_p}$ is a linearly independent set. On the other hand, the dimension of $\mathcal{P}_p(t)$ is N_p. Therefore $\{\phi_l\}_{l=1}^{N_p}$ is a basis of $\mathcal{P}_p(t)$. □

Remark 2.10. *Later on, we often refer to the nodal basis functions defined in Theorem 2.9 as standard basis functions of degree p of t.*

Corollary 2.11. *The following statements hold*

(i) A vertex basis function equals zero on the opposite edge.

(ii) An edge basis function equals zero on the other two edges.

(iii) An interior basis function equals zero on all edges.

Proof. The proof of this corollary follows from the fact (shown in the proof of Theorem 2.9) that the basis function associated with nodal points $(\hat{i}/p, \hat{j}/p, \hat{k}/p)$ is uniquely determined by

$$\phi = k \prod_{i=0}^{\hat{i}-1} (c_1 - \frac{i}{p}) \prod_{j=0}^{\hat{j}-1} (c_2 - \frac{j}{p}) \prod_{k=0}^{\hat{k}-1} (c_3 - \frac{k}{p}),$$

where k is a constant. □

Proposition 2.12. *Let e be the shared edge of two elements t and t' in the triangulation $\mathcal{T}$. If $P \in \mathcal{P}_p(t)$ and $Q \in \mathcal{P}_p(t)$ agree at all of the nodal points on e (including the two vertices), then P and Q agree along the whole e.*

Proof. The edge e can be parametrized using one parameter θ. Let $R = P - Q$. Then $R|_e$ is a polynomial of degree p, in variable θ. In addition, $R|_e$ vanishes at $p+1$ different values of θ associated with $p+1$ nodal points on e. Hence $R|_e \equiv 0$. In other words, P and Q agree along the whole edge e. □

So far we have been focusing on basis functions defined on each element. Now we extend the definition to the whole triangulation.

Let $\mathcal{P}_p(\mathcal{T})$ be the space of C^0 (continuous) piecewise polynomials of degree p, namely, the space of continuous functions that are polynomials of degree p on each element of triangulation $\mathcal{T}$. Each element of $\mathcal{T}$ is equipped with a set of nodal points of degree p. Note that some of the vertex and edge nodal points are shared by more than one element. Similar to Theorem 2.9, we will define basis functions associated with these nodal points.

Theorem 2.13. *Consider a way of labeling the nodal points of the triangulation $\mathcal{T}$ from n_1 to n_N. Let ϕ_i be the C^0 piecewise polynomial of degree p defined on $\mathcal{T}$ that equals 1 at the nodal point n_i and equal 0 at all other nodal points of $\mathcal{T}$. Then $\{\phi_i\}_{i=1}^N$ is a basis of $\mathcal{P}_p(\mathcal{T})$. This basis is called the nodal basis of $\mathcal{T}$.*

Proof. We first verify that ϕ_i are well defined by showing their existence and uniqueness. It is sufficient to show that such ϕ_i are uniquely defined on each element and smooth along shared edges of elements since they are C^0 piecewise polynomials.

Let t be an element in $\mathcal{T}$. If n_i does not belong to t, then by definition ϕ_i should be zero at all of the nodal point of degree p of t. By Lemma 2.7, $\phi_i|_t \equiv 0$. If n_i does belong to t, then ϕ_i equals 1 at n_i and equals zero at all other nodal points of degree p of t. By Theorem 2.9, ϕ_i is the basis function of $\mathcal{P}_p(t)$ associated with the nodal point n_i.

The smoothness (continuity) of ϕ_i along the shared edges of elements is obtained by using Proposition 2.12 and noting that two neighboring elements of the same degree share the same set of nodal points along the common edge.

It remains to show that $\{\phi_i\}_{i=1}^N$ is actually a basis of $\mathcal{P}_p(\mathcal{T})$. First, an argument similar to the one used in the proof of Theorem 2.9 shows that $\{\phi_i\}_{i=1}^N$ are

linearly independent. Now let P be an arbitrary function in $\mathcal{P}_p(\mathcal{T})$. Second, we will show that P can be written as a linear combination of $\{\phi_i\}_{i=1}^N$. Let $P' = \sum_{i=1}^N c_i\phi_i$, where c_i is the value of P at nodal point n_i. Because $\{\phi_i\}_{i=1}^N$ are C^0 piecewise polynomial of degree p, so is P'. Furthermore, from definition of P', $P - P'$ equals zero at all of the nodal points of $\mathcal{T}$. By Lemma 2.7, $P - P'$ is zero on each element of $\mathcal{T}$. Therefore, $P - P'$ is zero on the whole triangulation $\mathcal{T}$. In other words, $P = \sum_{i=1}^N c_i\phi_i$. This completes our proof. □

A nodal basis function can be referred to as a vertex, edge, or interior nodal basis function depending on the nodal point associated with it. However, in practice, they are usually called hat functions, bump functions and bubble functions respectively due to their shapes.

In the proof of Theorem 2.13, we observe that $\phi_i|_t \equiv 0$ for almost all elements $t \in \mathcal{T}$, except the ones that touch the nodal point n_i. In other words, these basis functions have compact support. Figure 2.5 illustrates three different kinds of support associated with different types of basis functions.

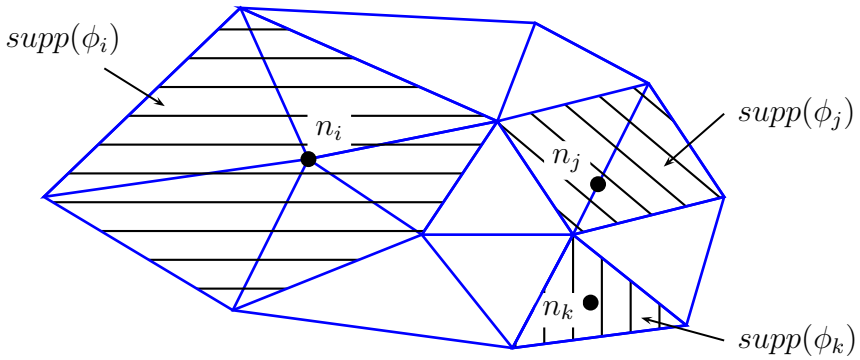


Figure 2.5: Supports of different kinds of basis functions.

In finite element method, solution is sought as a linear combination of basis functions of finite element space. If the space of piecewise polynomial of degree p, $\mathcal{P}_p(\mathcal{T})$, equipped with nodal basis function defined in Theorem 2.13 is chosen to be the finite element space, then the coefficients c_i in the expression of the finite element solution $f_{f.e} = \sum_{i=1}^N c_i\phi_i$ is actually an approximation of the exact

solution at the nodal point n_i. Because of this, c_i are called *degree of freedom* and the number of nodal points in $\mathcal{T}$ is called *number of degree of freedom*. Sometimes, the term "degree of freedom" is also used to refer to nodal points in a triangulation.

2.3 Transition Elements

In the previous section, we have studied the uniform case, in which all elements in triangulation $\mathcal{T}$ have the same degree. In this section, we will establish a foundation for p-adaptivity which allows elements of different degrees be in the same mesh (triangulation). This flexibility helps p-adaptive finite element methods to better capture the exact solutions' behaviors by adaptively choosing degrees for elements.

In a mesh with variable degrees, along the interfaces separating elements of different degrees, it is natural to use nodal points of higher degree for shared edges. Therefore, along degree interfaces, only elements of lower degrees need new sets of basis functions. These elements are called *transition elements*.

2.3.1 The first approach

Consider an admissible mesh, where there is no violation of 1-irregular rule and 2-neighbor rule (these rules are discussed in Chapter 3). In this mesh, a *transition element* is an element of degree p having one and just one neighbor of degree $p+1$ (the other neighbors if exist are of degree p). Figure 2.6 represents a transition element t with its neighbors. The edge shared by t and its neighbor of higher degree $(p+1)$ is called *transition edge*.

To define a set of basis functions for t we will follow the same idea of the previous section by making each basis function equal 1 at one nodal point and equal 0 at all other nodal points.

Assume edge two is the transition edge of t. By Corollary 2.11, all of the basis functions ϕ_j in $\mathcal{P}_p(t)$ that are not associated with edge two equal zero on the whole of edge two. In particular, these functions equal zero at nodal points of degree $p+1$ on the transition edge. Therefore, we can use these functions in the

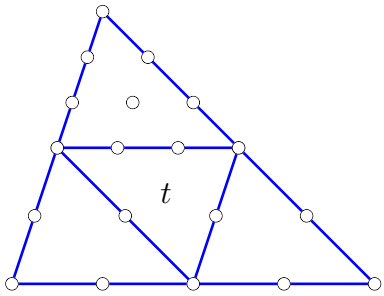


Figure 2.6: A transition element (t in the middle).

set of basis functions for transition element t without modification.

It now remains to define basis functions associated with nodal points of degree $p+1$ on the transition edge (including the two vertex nodal points). Label these points $n_{v_1}, n_{v_3}, n_{e_1}, \ldots, n_{e_{p-1}}$, where the first two are vertex nodal points and the rest are edge nodal points. Denote θ_j the function of the straight line perpendicular to edge two at n_{e_j} and S_{int} the set of interior nodal points of t. Let

$$\psi^{(e_i)} = C_i c_1 c_3 \prod_{\substack{j=1 \\ j\neq i}}^{p-1} \theta_j - \sum_{j\in S_{int}} \alpha_{ij}\phi_j. \tag{2.2}$$

Here c_1 and c_3 are equations of edge one and edge three; C_i and α_{ij} are chosen so that $\psi^{(e_i)}$ equals 1 at n_{e_i} and equals 0 at all the other nodal points of degree $p+1$ on edge two, as well as all interior nodal point of t. Clearly, $\psi^{(e_i)}$ also equals zero at nodal points of degree p on edge one and edge three. Hence $\psi^{(e_i)}$ are the transition edge basis functions for t.

Now we define the basis functions associated with the two vertices on the transition edge. We begin with standard vertex basis functions and use $\psi^{(e_i)}$ to modify them to have right values on the transition edge.

$$\psi^{(v_i)} = \phi_{v_i} - \sum_{j=1}^{p-1} \beta_{ij}\psi^{(e_j)}, \tag{2.3}$$

where $i = 1, 3$ and b_{ij} are chosen such that $\psi^{(v_i)}$ equals zero at all edge nodal points of degree $p+1$ on the transition edge. Obviously, $\psi^{(v_i)}$ equals 1 at the vertex v_i.

In summary, with two vertex basis functions defined by equation (2.3), $p-1$ edge basis functions defined by equation (2.2), and standard nodal basis functions

of degree p not associated the transition edge, we have a set of N_p+1 functions that equal 1 at one nodal point and equal 0 at all other nodal points of t. An argument similar to the one in Theorem 2.9 shows that the set is linearly independent. Let $\mathcal{P}_{p+1/2}(t)$ be the space spanned by that set of basis functions. Then $\mathcal{P}_{p+1/2}(t)$ is a polynomial space for the transition element t. Naturally we would want this newly defined space to contain the regular space of polynomials of degree p restricted on t. The following theorem ensures that desire.

Theorem 2.14. *$\mathcal{P}_p(t)$ is a subset of $\mathcal{P}_{p+1/2}(t)$.*

Proof. Since $\mathcal{P}_{p+1/2}(t)$ includes all of the basis functions in $\mathcal{P}_p(t)$ that are not associated with nodal point on the transition edge, it suffices to show that basis functions of degree p associated with the transition edge is contained in $\mathcal{P}_{p+1/2}(t)$.

From equation (2.3), we can write the standard vertex basis functions ϕ_{v_i} as a linear combination of functions in $\mathcal{P}_{p+1/2}(t)$:

$$\phi_{v_i} = \psi^{(v_i)} + \sum_{j=1}^{p-1} \beta_{ij}\psi^{(e_j)}, \text{ for } i = 1, 3.$$

Therefore, ϕ_{v_1} and ϕ_{v_3} are contained in $\mathcal{P}_{p+1/2}(t)$.

Now we show that the standard basis functions of degree p are also a linear combination of functions in $\mathcal{P}_{p+1/2}(t)$. Denote $\hat{\theta}_j$ the function of straight line perpendicular to transition edge at $n_{\hat{e}_j}$, a nodal point of degree p. Let

$$\hat{\psi}^{(\hat{e}_i)} = \hat{C}_i c_1 c_3 \prod_{\substack{j=1 \\ j\neq i}}^{p-2} \hat{\theta}_j - \sum_{j\in S_{int}} \hat{\alpha}_{ij}\phi_j$$

where $\hat{C}_i$ and $\hat{\alpha}_{ij}$ are chosen so that $\hat{\psi}^{(\hat{e}_i)}$ equals 1 at $\hat{e}_i$ and equals 0 at all of the other nodal points of degree p of t. By the uniqueness of basis functions proved in Theorem 2.9, $\hat{\psi}^{(\hat{e}_i)}$ is actually the nodal basis function of $\mathcal{P}_p(t)$ associated with the nodal point $\hat{e}_i$. Because $\phi_j \in \mathcal{P}_{p+1/2}(t)$ for $j \in S_{int}$, it is now sufficient to show that $\{\hat{\Theta}_j\}_{j=1}^{p-2}$ can be written as linear combinations of $\{\Theta_j\}_{j=1}^{p-1}$, where

$$\hat{\Theta}_i = \prod_{\substack{j=1 \\ j\neq i}}^{p-2} \hat{\theta}_j \quad \text{and} \quad \Theta_i = \prod_{\substack{j=1 \\ j\neq i}}^{p-1} \theta_j.$$

Since θ_j and $\hat{\theta}_j$ are lines of the same direction, the problem is reduced to one dimensional case: show that polynomials of degree $p-3$ can be written as linear combinations of basis polynomials of degree $p-2$. This statement is obviously true. □

Remark 2.15. *The set of basis functions defined above is not a unique one. For a transition element, there are more than one set of basis functions that equal 1 at a nodal point and equal 0 at all of the others.*

2.3.2 The second approach

In the previous subsection, we considered only meshes with no violation of 1-irregular rule and 2-neighbors rule. Now we consider more general meshes that might have violations of those two rules. In these meshes, a *transition element* is an element of degree p with at least one of its neighbors of degree $p+1$ or higher.

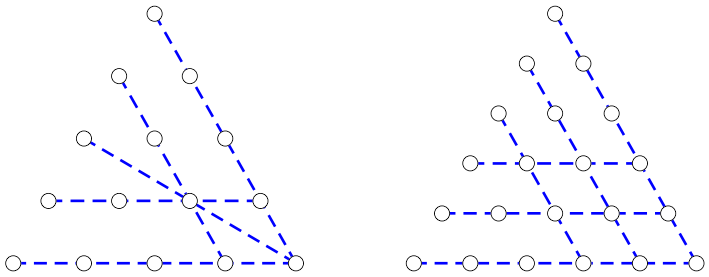
For the sake of clarity, we begin with a transition element t of degree p having one neighbor of degree $p+1$ and no other neighbor of degree higher than p. Without loss of generality, we can assume that the higher degree neighbor is across edge three of t. In other words, edge three is a *transition edge* of t.

Similar to the previous subsection, we can use the standard basis functions of degree p at the nodal points that are not associated with edge three. Again, it remains to define the basis functions associated with the transition edge three.

Define a special polynomial of degree $p+1$, which is zero at all standard nodal points of degree p of t, and identically zero on edges one and two by

$$\tilde{\phi}_{(p+1)} = \begin{cases} \prod_{k=0}^{(p-1)/2}(c_1 - k/p)(c_2 - k/p), & \text{for } p \text{ odd,} \\ \\ (c_1 - c_2)\prod_{k=0}^{(p-2)/2}(c_1 - k/p)(c_2 - k/p), & \text{for } p \text{ even.} \end{cases}$$

This polynomial is actually a product of equal number of lines parallel to edge one and edge two for p is odd, and is that same product multiplied with the median from vertex three for p is even. Figure 2.7 represents the lines in the formula of $\tilde{\phi}_{(p+1)}$ for $p = 4, 5$.



(a) $p = 4$ (b) $p = 5$

Figure 2.7: Lines in the formula of $\tilde{\phi}_{p+1}$ for $p = 4, 5$.

A polynomial space for the transition element is given by $\tilde{\mathcal{P}}(t) = \mathcal{P}_p(t) \oplus \tilde{\phi}_{(p+1)}$. In other words, we form $p+2$ basis functions $\{\psi_i\}_{i=1}^{p+2}$ as linear combinations of $\tilde{\phi}_{(p+1)}$ and N_p standard basis function of degree p of t. Here ψ_i is the basis function associated with the nodal point n_i on the transition edge. Denote S and S_{trans} the set of nodal point of t, and that associated with the transition edge, respectively. We have

$$\psi_i = \sum_{j \in S} \alpha_{ij}\phi_j + c_i\tilde{\phi}_{p+1}. \tag{2.4}$$

Matching both sides of equation (2.4) above at standard nodal points of t that are not associated with the transition edge yields $\alpha_{ij} = 0$ for $j \notin S_{tran}$. Therefore equation (2.4) becomes

$$\psi_i = \sum_{j \in S_{trans}} \alpha_{ij}\phi_j + c_i\tilde{\phi}_{p+1}. \tag{2.5}$$

This equation implies that ψ_i can actually be written as a linear combination of $\tilde{\phi}_{(p+1)}$ and $p+1$ standard basis functions of degree p associated with the transition edge. Since we know values of ψ_i at $p+2$ points on edge three, coefficients α_{ij} and c_i in equation (2.5) can be determined by solving a $(p+2) \times (p+2)$ linear system of linear equations. This approach is considered expensive as we have to solve a system of size $p+2$ for each basis functions ϕ_i.

In our code, PLTMG, we compute coefficients α_{ij} by matching both sides of equation (2.5) at standard nodal points of degree p associated with the transition edges. At each point, $\tilde{\phi}_{p+1}$ equals 0 and all of the ϕ_j equal 0 except one equals

1. As of ψ_i, we do not know its complete formula but its values on the transition edge are defined by its $p+2$ values at nodal points of degree $p+1$. In addition, the coefficient c_i can be computed by taking $(p+1)$th derivative of equation (2.5) in the tangential direction for the transition edge. In this approach of computation, all the coefficients are geometry independent. Therefore, we only need to do the calculation once and use the results for all elements.

Now we consider a more general case, where the higher degree element is of degree $p+k$, for $k>1$. Similarly, we can define a polynomial space for t as

$$\tilde{\mathcal{P}}(t) = \mathcal{P}_p(t) \oplus \{\tilde{\phi}_{(p+1)}(c_1 - c_2)^m\}_{m=0}^{k-1},$$

and the transition basis function ψ_i is given by

$$\psi_i = \sum_{j \in S_{trans}} \alpha_{ij}\phi_j + \sum_{m=0}^{k-1} c_{i,m}\,\tilde{\phi}_{p+1}(c_1 - c_2)^m. \tag{2.6}$$

Here the coefficients $c_{i,m}$ can be consecutively computed by taking $(p+m+1)$st derivative of equation (2.6), and α_{ij} are computed as in the previous case.

In this approach, we can be even more general by allowing one element to have more than one transition edge. The transition basis functions associated with transition edges are defined consecutively and almost independently. Assume edge two is the only transition edge left. Similarly, we can define a polynomial space for t as

$$\tilde{\mathcal{P}}(t) = \mathcal{P}_p(t) \oplus \{\tilde{\phi}^{(3)}_{(p+1)}(c_1 - c_2)^m\}_{m=0}^{k^{(3)}-1} \oplus \{\tilde{\phi}^{(2)}_{(p+1)}(c_3 - c_1)^m\}_{m=0}^{k^{(2)}-1}.$$

After defining the transition basis functions associated with edge three we can define those associated with edge two as in equation (2.6). The only difference is that the basis function ϕ_j associated with vertex one, is now ψ_{v1}, the transition basis function associated with edge three at vertex v_1.

If a third transition edge is present, it is treated analogously.

Theorem 2.16. *The finite element spaces constructed in Subsection 2.3.1 and Subsection 2.3.2 are* C^0.

Proof. Let e be the shared edge of two elements t and t', where t is of degree p and t' is of degree $p+k$, $k \geq 1$. Assume $P \in \tilde{\mathcal{P}}(t)$ and $Q \in \tilde{\mathcal{P}}(t')$ (if t' is not a transition element we can think of $\tilde{\mathcal{P}}(t')$ as $\mathcal{P}_{p+k}(t')$) agree at $p+k+1$ nodal points of degree $p+k$ on e. We will show that P and Q agree along the whole edge e.

Clearly, $\tilde{\mathcal{P}}(t)$ and $\tilde{\mathcal{P}}(t')$ could contain polynomials of degree higher than $p+k$, namely transition basis functions associated with edges other than e. However, in both approaches 2.3.1 and 2.3.2, these basis functions are defined to equal 0 on the whole e. Therefore, the restrictions of P and Q on e are 1-dimensional polynomials of degree $p+k$ or less.

An argument similar to the one in Proposition 2.12 shows that P and Q agree on the whole edge e. □

Chapter 3

Adaptive Meshing

Before we get into discussions of different algorithms of adaptive meshing we define finite element meshes to lay the foundation of this chapter.

A *finite element mesh* is a subdivision of the domain Ω of the differential equation into a number of convex subdomains, usually polygons, called elements. In this dissertation, we restrict the discussion to the case where $\Omega \subseteq \mathbb{R}^2$ and all elements are triangles. For simplicity of exposition, we assume that Ω is, in fact, the union of those elements.[1] In other words, we consider a finite element mesh a triangulation (of the domain Ω) that satisfies certain conditions. In the definition below, we impose Ciarlet's conditions in [32] for triangulations we are working with.

Definition 3.1. *A triangulation $\mathcal{T}$ of the domain Ω is said to be geometrically admissible if it satisfies the following conditions:*

(i) *For each triangle $\tau \in \mathcal{T}$, the set τ is closed and its interior $\mathring{\tau}$ is nonempty.*

(ii) $\bar{\Omega} = \bigcup_{\tau \in \mathcal{T}} \tau$.

(iii) *For two distinct triangles $\tau_1, \tau_2 \in \mathcal{T}$, one has $\mathring{\tau}_1 \cap \mathring{\tau}_2 = \emptyset$*

(iv) *Any edge of any triangle τ_1 in $\mathcal{T}$ is either a subset of the boundary Γ of Ω, or the whole edge of another triangle τ_2.*

[1] Extensions to allow curved edges on the boundary of Ω are implemented in PLTMG but are not discussed in this dissertation.

(v) If an edge of τ belongs to the boundary Γ of Ω, it is either on the Dirichlet boundary or Neumann boundary. No element has an edge containing both Dirichlet and Neumann portions of the boundary.

Remark 3.2. *It is clear from definition 3.1 that two distinct elements in a triangulation can share an edge or a vertex or nothing.*

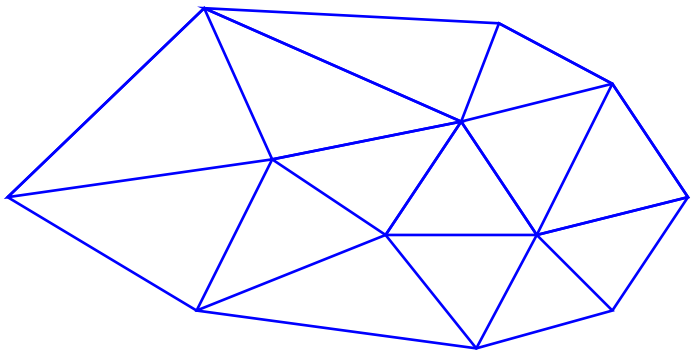


Figure 3.1: An geometrically admissible mesh.

An example of a valid triangulation is showed in Figure 3.1 and an example of a triangulation that violates condition (iv) is showed in Figure 3.2.

3.1 Mesh Smoothing

In this section, we summarize the result of Bank and Smith in [25] about *mesh smoothing* based on geometry.

In finite element methods, the calculation of stiffness matrix and right hand side are done on the reference element using affine maps. In this way, the work is significantly reduced since part of the calculation only need to be done once and used for all elements. However, to guarantee that the calculation is sufficiently

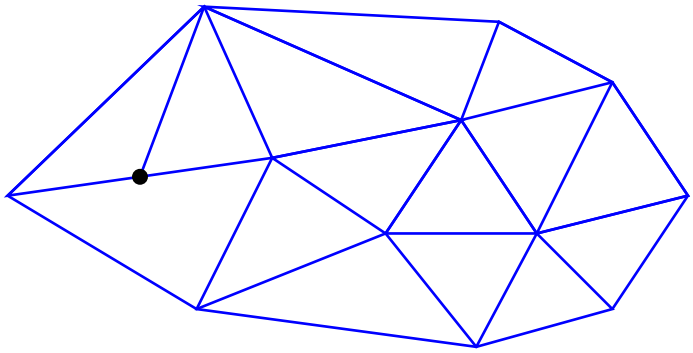
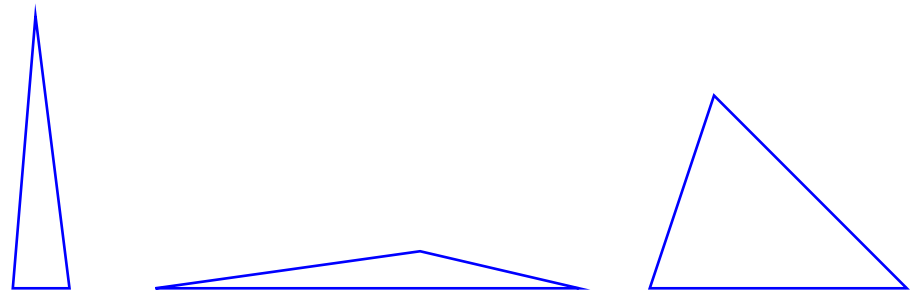


Figure 3.2: A mesh violates condition (iv) in Definition 3.1.

accurate, the geometry of each element (triangle) needs to satisfy a condition called *shape regularity*.

Generally speaking, shape regularity condition requires triangles not to be too "thin" nor too "flat" (see examples in Figure 3.3). In other words, the condition requires triangles' smallest angle or the ratios of their in-radius and circum-radius to be bounded below. There are no fixed values for these lower bounds. However, in practice, one usually requires the smallest angle of an element in finite element mesh is at least 30 degree.



(a) bad (b) bad (c) good

Figure 3.3: Examples of elements with different shape regularity qualities.

Consider an element t with vertices $v_i = (x_i, y_i), 1 \leq i \leq 3$, oriented counter-

clockwise as shown in Figure 3.4. In [25], Bank and Smith defined a *shape regularity quality function* $q(t)$ for t as follows

$$q(t) = \frac{4\sqrt{3}|t|}{|\ell_1|^2 + |\ell_2|^2 + |\ell_3|^2}. \tag{3.1}$$

Here $|\ell_i|$ is the length of edge ℓ_i and $|t|$ is "signed" area computed by

$$|t| = \frac{(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)}{2}. \tag{3.2}$$

This area formula is very convenient in the sense that by using the change of its sign, one can recognize the reorientation of the vertices of t.

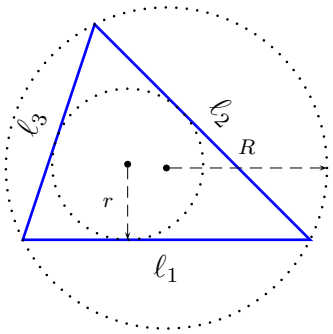


Figure 3.4: An element with its parameters.

Clearly from its definition in Equation 3.1, the shape regularity quality function $q(t)$ has absolute value that equals one for an equilateral triangle and approaches zero for triangles with small angles. In addition, $q(t)$ is independent of the size of t. That is, the shape regularity qualities of two similar triangles are the same no matter how different they are in size.

Definition 3.3. *The shape regularity quality of a mesh is the smallest shape regularity quality of the elements in it.*

It is often helpful to think of a mesh (triangulation) as a grid or a set of vertices and their connectivity structure. In the next step, we discuss a method to improve shape regularity quality of a mesh by moving its vertices around "slightly" and keep the connecting structure (topology) fixed. This method is categorized as a *mesh smoothing* technique.

Let $\mathcal{T}$ be a geometrically admissible triangulation of the domain Ω. For the purposes of mesh smoothing, vertices of $\mathcal{T}$ are decomposed into three separated sets, called *corner, boundary/interface* and *interior* vertices. Roughly speaking, a *corner vertex* is a vertex that is critical for defining the geometry of the region, the boundary conditions, the interfaces, etc., whose movements would compromise the integrity of the domain. Such vertices should remain fixed. *Boundary/interface* vertices are the ones lying along boundaries and interfaces. These vertices can only be moved along the boundaries and interfaces they belong to. All other vertices are interior vertices and they can be moved freely in all directions.

Let $\mathcal{F}$ be a family of triangulations of Ω with the same topology and all members of $\mathcal{F}$ share the same constraints for corner and boundary/interface vertices. We would like to find $\mathcal{T} \in \mathcal{F}$ with the best shape regularity quality. More precisely, we want to seek $\mathcal{T}$ as the solution of the optimization problem: find a triangulation $\mathcal{T} \in \mathcal{F}$ such that

$$\min_{t \in \mathcal{T}} q(t) = \max_{\mathcal{T}' \in \mathcal{F}} \min_{t \in \mathcal{T}'} q(t) \tag{3.3}$$

However, this optimization problem is very expensive to solve especially when the number of vertices becomes large. To keep the cost of the technique reasonable, one could find a triangulation in $\mathcal{F}$ with good shape regularity quality but not necessary the best one. In [25], Bank and Smith proposed a Gauss-Seidel like method, in which they sweep through the vertices, locally optimizing the position of a single vertex while keeping all the others unchanged. During this procedure, after solving a local problem, the quantity

$$\min_{t \in \mathcal{T}} q(t)$$

can only be increased or remain unchanged and the quality of bad elements in the triangulation are generally improved. Now what is left is to solve the local optimization problem effectively.

Assume we want to optimize the position of vertex $v_i = (x_i, y_i)$ while keeping other vertices fixed. For the sake of simplicity, we assume that v_i is an interior vertex. Let Ω_i be the subregion of Ω formed by elements sharing the vertex v_i as shown in Figure 3.5. Let t_1 and t_2 be the two elements in Ω_i with worst shape

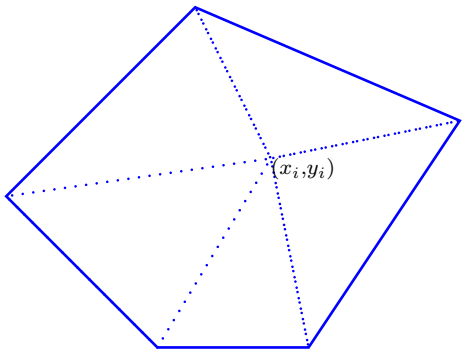


Figure 3.5: Local region Ω_i surrounded vertex v_i.

regularity quality :

$$\alpha = q(t_1) = \min_{t \in \Omega_i} q(t) \quad \text{and} \quad \beta = q(t_2) = \min_{\substack{t \in \Omega_i \\ t \neq t_1}} q(t)$$

Obviously, there is unique point (x_i', y_i'), for which the triangles corresponding to t_1 and t_2 with the vertex (x_i, y_i) replaced by (x_i', y_i') have equal qualities $\alpha' \geq \alpha$. This point is characterized as a point of tangency of the circles for the level curve for the two triangles and can be computed directly from the geometry of the vertices which remain fixed in t_1 and t_2. For the new position of v_i, if the qualities of other triangles in Ω_i are greater than α' then this is the exact solution for our local optimization problem. Fortunately, this is usually the case in practice. When it is not the case, one could use a line search along the line segment connecting (x_i, y_i) and (x_i', y_i') to find a good position for v_i.

Remark 3.4. *The algorithm of mesh smoothing we discuss here is based purely on geometry. In [25], Bank and Smith also proposed two other algorithms of mesh smoothing, one based on local interpolation errors, and the other a posteriori error estimates, where vertices are placed in the ways that minimize approximated errors. In particular, they also pointed out through numerical experiments that mesh smoothing with fixed topology by itself is not necessarily a good strategy for adapting the mesh. However, it is very efficient when used in conjunction with mesh refinement. We shall discuss some mesh refinement strategies in the next section.*

3.2 h-Adaptive Meshing

In h-refinement, one refines an element in two or more children elements of smaller sizes while keeping the degree of the new elements (degrees of the basis functions associated with these elements) the same with their father's.

Based on range of influence, h-refinements can be categorized in three different strategies.

In *global mesh refinement*, every element in the mesh is refined (usually in the same way) to obtain a finer mesh. Clearly, this is the simplest strategy to implement. However, it is also the most expensive strategy since many elements are generated away from the area of interest. Sometimes, global mesh refinement is referred to as uniform refinement.

A variation of global mesh refinement is *semi-global mesh refinement*, in which elements in one or more selected cross-sections of the mesh are refined. In certain cases, this strategy may be implemented as easily as global refinement and may be less wasteful. Nevertheless, this strategy does not always work and is still considered not economical.

In the rest of this section, we discuss two different approaches of (adaptive) *local mesh refinement*, in which only selected elements are refined. This is a very attractive strategy especially for problems with singularities or sharp fronts since the refinement can be restricted to those portions of the domain where it is needed.

Requirement 3.5. *For local mesh refinement to be efficient, it is necessary that:*

(i) we are able to decide what elements to be refined cheaply

(ii) the sparsity of the resulting systems of linear equations is preserved as the mesh is refined.

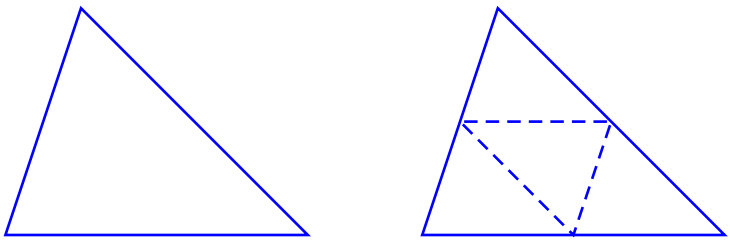
(iii) the adaptive local mesh refinement procedure can be implemented cheaply

Later, in chapter 4, we study a posteriori error estimates, with which we can select the best (or nearly best) elements to refine at low cost. In this chapter, we assume (i) and focus only on (ii) and (iii).

3.2.1 Red-Green Mesh Refinement

In this subsection, we present the work of Bank, Sherman and Weiser in [24]. The ideas of the work later give rise to our extension for p-refinement which is discussed in the next section.

As mentioned in the previous section, shape regularity quality of the mesh is very important in finite element methods. In h-adaptive meshing, to preserve the quality of the current mesh, one can use *red refinement*[2], which is sometimes called *bisection-type* mesh refinement. In this type of refinement, a triangular element t is subdivided into four triangles called *sons* of t, by pairwise connecting the midpoints of the three edges of t. Figure 3.6 illustrates an element t and its children after a red refinement.



(a) before (b) after

Figure 3.6: An element before and after red refinement.

Obviously, in red refinement, the children elements are geometrically similar to their farther. Therefore, they have the same shape regularity quality as their father. This is an advantage of this type of refinement. However, new vertices introduced in red refinement usually break the conformity of the triangulation (violating condition (iv) in definition 3.1 of a geometrically admissible mesh). This can be seen from Figure 3.6 where an element t is red refined several times. In the figure, except for vertices of t, all other vertices are non-conforming.

These non-conforming vertices are usually called irregular vertices and are rigorously defined as follows.

Definition 3.6. *A vertex is said to be regular if it is a corner of each element it*

[2] The name "red refinement" came after the name "green refinement" which is discussed later.

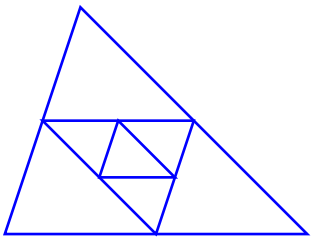


Figure 3.7: Non-conforming vertices created by red refinement.

touches. A vertex is said to be irregular if it is not regular.

Definition 3.7. *The irregular index of a mesh is the maximum number of irregular vertices on a side of any element in the mesh. A k-irregular mesh is a mesh with irregular index k*

Figure 3.8 shows an example of a 2-irregular mesh.

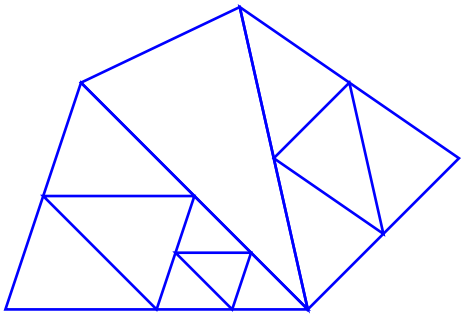


Figure 3.8: A 2-irregular mesh.

Remark 3.8. *Note that all boundary vertices should be regular.*

In general, it is advantageous to "regularize" a mesh by restricting the number of irregular vertices on each edge. There are several reasons for that: simplifying computations such as matrix assembly and mesh refinement, increasing approximation power by insuring that neighboring elements are not too different in sizes, and guaranteeing that each element is in the support of a bounded number of basis functions. There are several ways to achieve this regularization. In [24], Bank et al. suggested to use the following *1-irregular rule* and some of its variants.

Rule 3.9. *1-Irregular Rule: Keeping the number of irregular vertices on any edge of any element in the triangulation be at most one. In other words, refine any element for which any of its edges contains more than one irregular vertex.*

Figure 3.8 can also served as an example of a mesh with a violation of 1-irregular rule. The mesh after fixing the violation is shown in Figure 3.9.

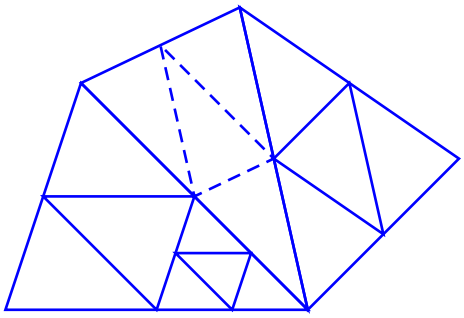


Figure 3.9: Mesh in Figure 3.8 after fixing 1-irregular rule violation.

In order to monitor the number of irregular vertices on an edge of an element, one could use the information of its level and neighbors.

Definition 3.10. *The level ℓ_{t_i} of an element t_i is defined inductively as follows*

$$\ell_{t_i} = \begin{cases} 1 & \text{if } t_i \in \mathcal{T}_0 \\ \ell_{t_f} + 1 & \text{if } t_i \notin \mathcal{T}_0 \end{cases},$$

where t_f is the father of t_i and $\mathcal{T}_0$ is the geometrically admissible initial mesh.

Definition 3.11. *The neighbor t_i^j of element t_i across its jth edge e_i^j is the smallest element with one edge completely overlapping e_i^j.*

Clearly, the number of irregular vertices on an edge of an element is related to the difference of its level and the level of one of its neighbors across that edge.

Let $\mathcal{T}$ be a geometrically admissible mesh. Assume that some elements in $\mathcal{T}$ are selected to be red-refined owing to, for example, having large errors. These refinements, in turn, introduce some irregular vertices. During the refinement process, 1-irregular rule is applied as often as possible to accomplish a regularized mesh which, according to [24], has the following properties:

Proposition 3.12. *Let $\mathcal{T}'$ be the mesh obtained from $\mathcal{T}$ after some red refinements, and regularization using 1-irregular rule. Then*

(i) $\mathcal{T}'$ *has irregular index 1.*

(ii) $\mathcal{T}'$ *uniquely contains the fewest elements of any 1-irregular mesh that can be obtained by refining* $\mathcal{T}$*.*

(iii) $|\mathcal{T}| \leq 13|\mathcal{T}'|$.

Remark 3.13. *The property (iii) of proposition (3.12) is usually pessimistic (see remark 3.16).*

Beside the nice properties above, $\mathcal{T}'$ is still not a geometrically admissible mesh owing to the presence of irregular vertices. In addition, a triangulation of irregular index 1 does not guarantee that the number of nonzero basis functions[3] in each element are exactly three. An example is illustrated in Figure 3.10, where the triangulation satisfies 1-irregular rule , but the four basis functions corresponding to the vertices marked by dots are nonzero in t.

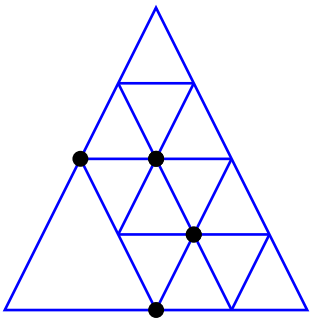


Figure 3.10: Four nonzero basis functions in t.

To fix these issues, Bank et al. proposed using *green refinement*[4], in which a vertex is connected to the midpoint of the opposite edge of the element we want to refine (see Figure 3.11). The use of green refinement is determined by the *green rule* described as follows

[3]Here we only consider linear basis functions.

[4]The name "green refinement" came from graph theory where sometimes special edges are distinguished by color.

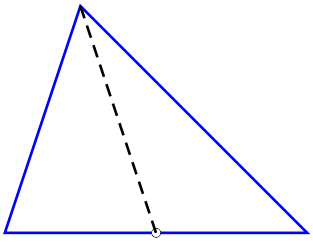


Figure 3.11: Green refinement.

Rule 3.14. *Green Rule: With as few elements as possible , green refine any element with an irregular vertex on one or more of its edges.*

For 1-irregular meshes, there are three cases in which green rule can be applied. These cases are shown in Figure 3.12.

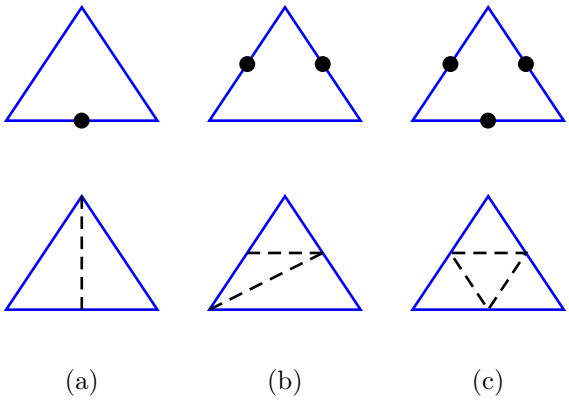


Figure 3.12: Green rule

Proposition 3.15. *Let $\mathcal{T}'$ be an 1-irregular mesh, for example the resulting mesh in Proportion 3.12. Assume that $\mathcal{T}''$ is generated from $\mathcal{T}'$ by applying the green rule wherever possible. Then the following hold:*

(i) For any element t'' in $\mathcal{T}''$, there are at most three basis functions having supports in it. In addition, the restrictions of these basis functions in t'' are linearly independent.

(ii) *In $\mathcal{T}''$, the support of a basis function intersects with those of at most twelve other basis functions.*

(iii) $|\mathcal{T}''| \leq 2|\mathcal{T}'|$.

Remark 3.16. *The properties (i) and (iii) of proposition 3.15 are usually pessimistic. The most common number of non-zeros in a row of the stiffness matrix is seven, and for most meshes encountered in practice $\mathcal{T}''$ contains fewer than twice as many elements as $\mathcal{T}$. Here $\mathcal{T}'$ is obtained from $\mathcal{T}$ after some red refinement and 1-irregular regularization.*

In addition, one could use a more aggressive refinement strategy by applying, in conjunction with 1-irregular rule and green rule, the following 2-neighbor rule.

Rule 3.17. *2-Neighbor Rule: Red refine any element t with two neighbors that have been red refined.*

An example of a mesh with a violation of 2-neighbor rule is shown in Figure 3.13.

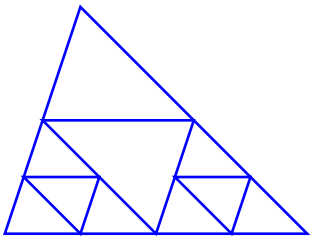


Figure 3.13: A mesh with 2-neighbor rule violation.

When the 2-neighbor rule is used together with the 1-irregular rule, one gets an 1-irregular mesh in which each remaining irregular vertices is located at the midpoint of an edge of a unique element. This implies that for such a mesh, only the case in Figure 3.12a occurs when green rule is applied. For the resulting mesh, analogues of properties (ii) and (iii) in proposition 3.15 hold, but the constants are usually bigger.

Algorithm 1 Local Meshing Procedure For Red Refinement

Procedure REFINE
  while $(i \leq nt)$ **do**
    for $j = 1$ **to** 3 **do**
      if t_i^j *is unrefined* **then**
        if t_i^j *has more than one neighbor* **or** $\ell_i > \ell_{t_i^j} + 1$ **then**
          $DIVIDE(t_i^j)$;
        end if
      end if
    end for
    if $DVTEST(t_i)$ **then**
      $DIVIDE(t_i)$;
    end if
    $i \leftarrow i + 1$;
  end while
End
Procedure DIVIDE
  $s_i \leftarrow nt + 1$; $nt \leftarrow nt + 4$;
  for $j = 0$ **to** 3 **do**
    create t_{s_i+j};
  end for
End

Algorithm 1 above is an algorithm implementing 1-irregular rule in conjunction with 2-neighbor rule.

Here we assume that a Boolean-valued function $DVTEST$, which decides whether an element should be refined, is available. Usually $DVTEST$ is the output of a self-adaptive mechanism within the code that uses local error indicators. Sometimes, $DVTEST$ can be a user specification of a fixed refinement pattern. An element in the mesh may be refined either because $DVTEST$ indicates it should be refined, or because it violates the 1-irregular rule or 2-neighbor rule. It is also possible that an element satisfies both rules at the beginning but violate one of

them later in the refinement process owing to the refinement of one of its neighbors. This implies that an element can be examined by the algorithm more than once.

Note that in Algorithm 1, elements are processed in the order they are created, newly created elements are placed at the end of the working list. In particular, when an element is examined, its neighbors are tested against 1-irregular and 2-neighbor rules before it is checked by $DVTEST$. This guarantees that these rules are satisfied by meshes generated by Algorithm 1, and that remaining irregular vertices are the sole edge midpoints in some elements.

Since a given element has at most three neighbors, we test (and possibly refine) at most four elements at any step of Algorithm 1. Hence, the complexity of Algorithm 1 is linear in the number of elements.

3.2.2 Longest Edge Bisection

As indicated by the title, in this subsection we discuss *longest edge bisection*, another approach of h-adaptive meshing.

In longest edge bisection, an element is refined into two smaller elements by connecting the midpoint of its longest edge with the opposite vertex as shown in Figure 3.14.

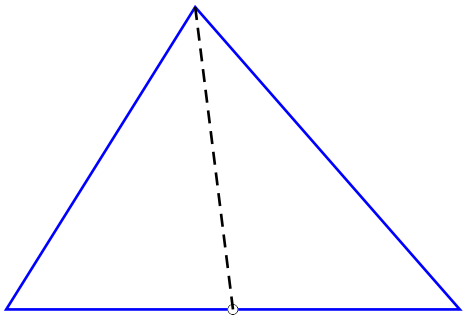


Figure 3.14: Longest edge bisection.

Obviously, one chooses to bisect the longest edge to maintain the shape regularity quality of the mesh. In an element, the angle opposite to the longest edge is the biggest one. Therefore, refinement by dividing that angle would reduce the chance to have elements with small angles. However, bisecting an element

introduces an irregular (nonconforming) vertex. This leads to further refinement. The question is whether the process terminates in finite steps and whether the resulting mesh has some control of the smallest angles.

The following theorem on "a lower bound on the angles of triangulation constructed by bisecting the longest edge" was given by Rosenberg and Stenger in 1975.[5]

Theorem 3.18. *Let α_0 be the smallest interior angle of $\mathcal{T}_0$, a given initial geometrically admissible triangulation. If α_j is the smallest angle of the triangulation $\mathcal{T}_j$ obtained by the jth iterative bisection of all the triangles generated from $\mathcal{T}_0$, then $\alpha_j \geq \alpha_0$, for all j.*

Later, in 1984, Rivara introduced several algorithms using longest edge bisection and gave a proof of their finiteness. The following is the simplest version of her algorithms for local refinement discussed in [50]. Figure 3.15 shows an

Algorithm 2 Local Mesh Refinement Using Longest Edge Bisection

For each $t \in \mathcal{S}_0$, bisect t by its longest edge.
$k \leftarrow 1$;
while $I_k \neq \emptyset$ **do**
  for $t \in I_k$ *with irregular vertex* P **do**
    Bisect t by its longest edge.
    if P *is not on the longest edge* **then**
      Join P with the midpoint of the longest edge of t.
    end if
  end for
  $k \leftarrow k + 1$;
end while

Here S_0 is the set of elements to be refined and I_k is the set of elements with irregular vertices at step k.

example of using longest edge bisection algorithm, in which newly created edges

[5]The original result of Rosenberg and Stenger in [52] was stated slightly different. Here we use the version of Rivara used in [50].

are labeled in the order they are created.

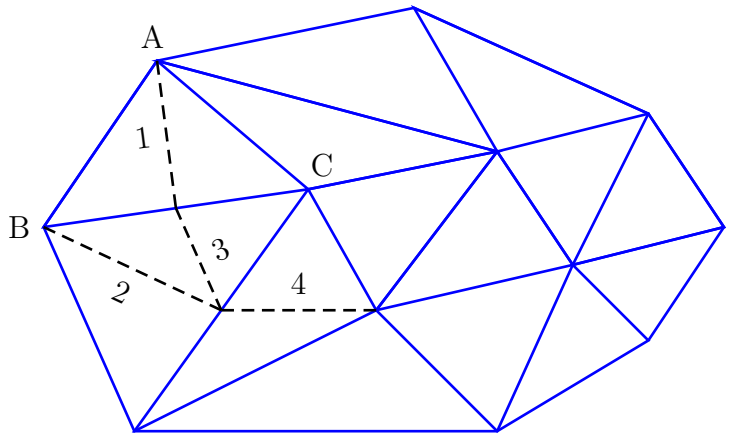


Figure 3.15: Local application of Algorithm 2 for refining element ABC.

In PLTMG, we apply Algorithm 2 with some modifications. First, elements are considered to be refined one at a time, according to their error indicators. Second, a weaker condition for choosing the longest edge is also used. That is, if the irregular vertex P is not on the longest edge but the length of the edge containing P is, for example, 90% of that of the longest one, then we bisect the element by the edge containing P. The reason for doing this is that we want to focus more on refinement by error estimate, rather than on refinement by geometry non-conformity. Outline of the refinement procedure used in PLTMG is described in Algorithm 3 below.

Here the STOP_COND is based on a threshold of number of vertices or number of degree of freedom.

Remark 3.19. *Even though theorem 3.18 guarantees a lower bound on the angles of triangulation constructed by bisecting the longest edge, the shape regularity quality of the mesh could be reduced significantly after several refinements. To deal with this issue, in* PLTMG*, we apply the mesh smoothing technique (described in section 3.1), edge flipping, etc., to improve the quality of the mesh generated from*

Algorithm 3 Local Mesh Refinement in PLTMG

Calculate error estimates.

Build a max-heap H of elements according to their errors.

repeat

$t \leftarrow H(1)$

Bisect t and its neighbors as in Algorithm 2.

Update error estimates.

Update heap H.

until STOP_COND

h-adaptive meshing.

Remark 3.20. *One should notice from Algorithm 3 that an element could be refined more than once before the approximated solution is recalculated. This is possible because there exists one way to approximate errors for children elements in a refinement using information of their father. However, the accuracy of these approximated errors degenerates after each refinement. Therefore, in* PLTMG*, we limit the maximal number of refinements of an element before the whole problem is resolved.*

Remark 3.21. *In* PLTMG*, h-unrefinement is also implemented. Originally, in h-unrefinement, vertices with small surrounding errors are gradually removed from the mesh. In order to have the h-version and the p-version of unrefinement compatible with each other, we currently implemented h-unrefinement as a process of removing elements with small errors step by step. In both cases, when a vertex/an element is removed from the mesh, the connecting structure is updated and edge flipping and mesh smoothing are used to improve the shape regularity property of the mesh.*

3.3 *p*-Adaptive Meshing

In p-adaptive meshing, we fix the geometry of mesh and achieve a better solution approximation by modifying (usually increasing) the degrees of the elements

in the mesh. This is a different approach to get better approximation spaces. In h-refinement, we use elements of small sizes, while in p-refinement, we use elements with higher degree.

Since the geometry of a mesh is unchanged after a p-refinement, we assume, in this section, that all the meshes we working with are geometrically admissible. In addition, in this section, when we talk about a mesh we refer to both its topology and the set of degrees of freedom associated with elements in the mesh.

Similar to h-refinement, there are global and local p-refinements.

In global p-refinement, also called uniform p-refinement, the degree of every element in the mesh is increased by the same quantity. This strategy of refinement appears to be very effective for a class of problem where the exact solutions are very smooth and can be well approximated by polynomials.

In practice, however, we usually encounter problems with singularities, sharp fronts, or rapid change in part of the solutions. For these classes of problem, using uniform p-refinement is very expensive since many elements away from critical area also use high degrees. Because of this reason, local p-refinement, where only selected elements are refined, is very attractive.

The rest of this section is devoted to local p-refinement and other local p-adaptive meshing.

3.3.1 Refinement Rules

For convenience, in this section, "refine" is understood as "p-refine", and refinement as p-refinement unless otherwise specified. Also for now, we restrict our discussion to cases in which the degree of an element is increased by one in a p-refinement.

Similar to the previous section, we assume that we have a way to determine cheaply what elements should be refined. We now focus only on developing an efficient algorithm for p-refinement. The followings are the goals we want to achieve when building such algorithm.

Requirement 3.22. *For local p-refinement to be efficient, it is necessary that:*

(i) Computation is not expensive.

(ii) Neighboring elements are not too different in degree.

(iii) The algorithm can be implemented cheaply.

The requirements (i) and (iii) are natural for any practical algorithm. For p-refinement, we can achieve (i) by ensuring that each element is in the support of a bounded number of basis functions. In doing so, we could preserve the sparsity of the resulting system of linear equation as the mesh is refined. In addition, we could also limit the number of special cases. Not only does this help to simplify computation but it also makes the algorithm simpler to implement.

As of requirement (ii), the purposes are to smooth the changes of approximated solutions from element to element, and to increase approximation power.

To achieve the goals in Requirement 3.22 we propose using the following rules.

Rule 3.23. *1-Irregular Rule (p-version): The difference in degree of neighboring elements can be at most one. Refine any element t of degree p with a neighbor of degree higher than $p+1$.*

Rule 3.24. *2-Neighbor Rule (p-version): For any element, there should be no more than one neighbor of higher degree. Refine any element t of degree p with two or three neighbors of degree $p+1$.*

Remark 3.25. *These rules are inspired by those of the same names in Red-Green (h-) refinement. The only difference is that here the degrees of elements play the role of their levels.*

Definition 3.26. *A mesh is said to be admissible (in degree) if there is no violation of 1-irregular rule and 2-neighbor rule.*

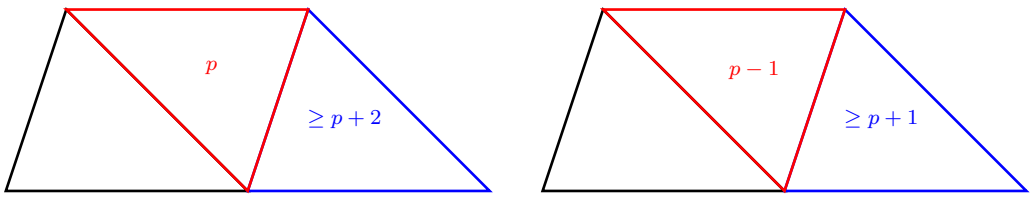
Remark 3.27. *Obviously, if a mesh is admissible, then there is only one special case, in which an element of degree p has one and just one neighbor of degree $p+1$ (the other neighbors if exist are of degree p). Such element is called transition element and their basis functions are especially defined in subsection 2.3.1.*

Now there is only one concern left before we can move on to the next subsection. As far as we know, we cannot recalculate error estimates for an element which has been p-refined before the whole problem is resolved. Therefore, unlike h-refinement, in p-refinement we can only refine an element at most once. The question is whether this is possible when we have both refinement owing to large error estimates, and refinement owing to violations of 1-irregular rule and 2-neighbor rule. The following theorem answers this question.

Theorem 3.28. *If we start with an admissible mesh and no element is required to refine by error more than once, then in a refinement, with enforcement of 1-irregular rule and 2-neighbor rule, each element can be refined at most once.*

Proof. This is a proof by contradiction.
Let t_i be the first element in the mesh to be refined twice. Assume t_i is of degree p right before its second refinement, and therefore of degree $p-1$ in the starting mesh. Now we consider two cases:

Case 1: Violation of 1-irregular rule: Suppose the second refinement of t_i is caused by a refinement of its neighbor t_j. This implies that t_j is of degree $p+2$ or higher right before the second refinement of t_i (see Figure 3.16 on the right). Since t_i is the first element to be refined twice, the refinement of t_j is its first refinement. Therefore, in the starting mesh, the degree of t_j is at least $p+1$. Hence the violation of 1-irregular rule between t_i and t_j in the starting mesh (see Figure 3.16 on the left). This contradicts with the assumption that the starting mesh is admissible.

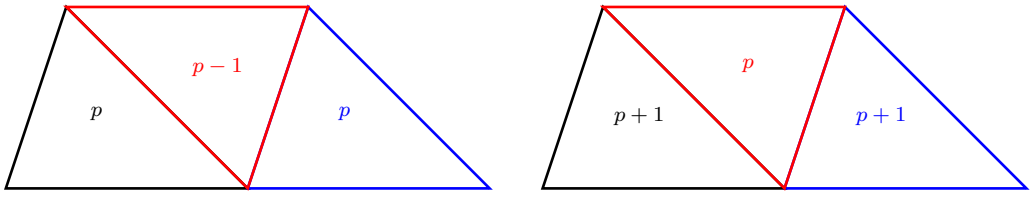


(a) Before 2nd refinement of t_i (b) Starting Mesh

Figure 3.16: Second refinement of t_i caused by a violation of 1-irregular rule.

Case 2: Violation of 2-neighbor rule: Suppose the second refinement of t_i is caused by refinements of its neighbors t_j and t_k. This implies that t_j and t_k is of

degree $p+1$ right before the second refinement of t_i (see Figure 3.17 on the right). Since t_i is the first element to be refined twice, the refinements of t_j and t_k are their first refinements. Therefore, in the starting mesh, t_j and t_k are of degree p. Hence, there is a violation of 2-neighbor rule between t_i, t_j and t_k in the starting mesh (see Figure 3.17 on the left). This contradicts with the assumption that the starting mesh is admissible. □



(a) Before 2nd refinement of t_i (b) Starting Mesh

Figure 3.17: Second refinement of t_i caused by 2-neighbor rule.

3.3.2 Data Structure for p-Adaptive Meshing

In terms of data structure, the major change we made when we incorporated p-adaptive meshing in PLTMG is the reconstruction of data array $ITDOF$. This reconstruction caused a massive changes in the code since $ITDOF$ is used in more than a hundred subroutines.

In PLTMG, GF, short for grid function, is a real array whose Ith column contains all information of approximated solutions (values, derivatives of the exact solution) at the degree of freedom I. On the other hand, $ITDOF$ is an integer array whose Ith column contains pointers of degrees of freedom associated with element I to the grid function array GF.

In the latest version of PLTMG before we incorporated p-adaptive meshing, elements are allowed to have degrees up to three. However, the degrees of elements in the mesh must be the same. In that version, $ITDOF$ stores pointers of all possible dofs in an element (see detail in Table 3.1). If we extended this data structure for the p-version, where elements have variable degrees up to 10, we would need at least 66 entries in each column of $ITDOF$ in order to store pointers of all the dofs of an element of degree up to 10. This would be a huge waste of memory considering

Table 3.1: ITDOF in old versions of PLTMG.

$ITDOF(1, I)$	first vertex dof pointer
$ITDOF(2, I)$	second vertex dof pointer
$ITDOF(3, I)$	third vertex dof pointer
$ITDOF(4, I)$	first edge dof pointer
$ITDOF(5, I)$	second edge dof pointer
$ITDOF(6, I)$	third edge dof pointer
$ITDOF(7, I)$	interior dof pointer
$ITDOF(8, I)$	element degree

the fact that we would want PLTMG to be able to work with meshes having up to three million elements.

To overcome the challenge, we impose some strict rules in ordering dofs locally within each element.

Rule 3.29. *Locally within each element, dofs are labeled consecutively in counter-clockwise direction and in the following decreasing priority: vertex dofs, edge dofs and interior dofs.*

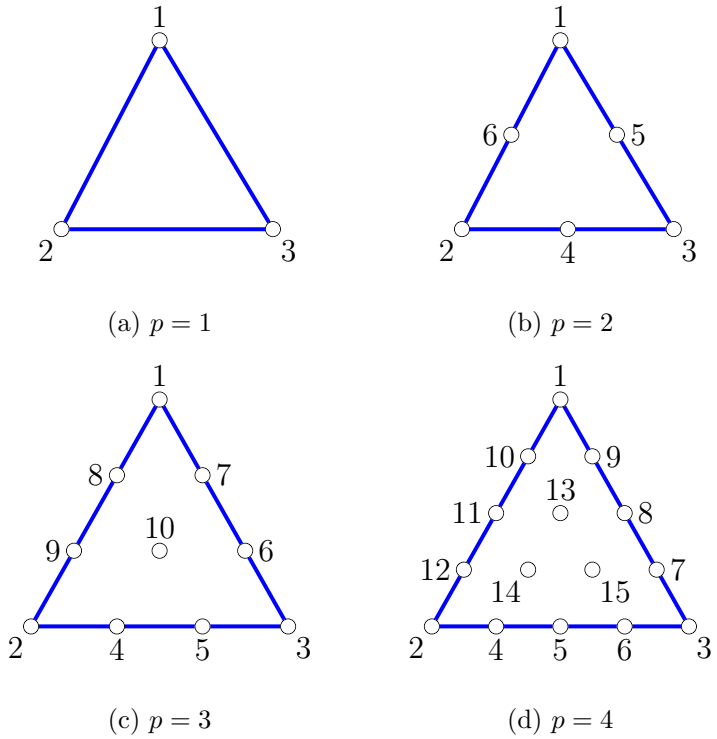
Layouts of dofs for the elements of degree from 1 to 4 are illustrated in Figure 3.18

We also have some rules for pointers to the grid function array.

Rule 3.30. *Pointers to grid function array GF of dofs on the same edge or in interior of the same element must be consecutive.*

With this rule, we can use the smallest (or biggest) index of the dofs to determine pointers of other dofs on the same edge or in the same interior of an element if we know the associated degree. This is the place where the knowledge about the numbers of nodal points on an edge and in interior of an element stated in section 2.1 becomes very useful.

Remark 3.31. *Since a vertex can be shared by more than two elements, we cannot require three vertices of every element having consecutive indices in the grid function array.*



(a) $p = 1$ (b) $p = 2$

(c) $p = 3$ (d) $p = 4$

Figure 3.18: Local ordering of dofs in elements of degree $p = 1, \ldots, 4$.

Currently, in PLTMG the first three entries in each column of $ITDOF$ are still reserved for pointers of vertex dofs. The next three entries are used to store the smallest or biggest pointers dofs of each edge. More specifically, we let $ITDOF(J, I)$, $J = 4, 5, 6$ be the smallest pointer of dofs on edge $J-3$ if the counter-clockwise direction of the edge in the element is the same with the increasing direction of the pointers, and we let $ITDOF(J, I)$ be the negative of the biggest pointer if otherwise. In addition, the seventh entry $ITDOF(7, I)$ is the smallest pointer of interior dofs of element I. Lastly, the eighth entry $ITDOF(8, I)$ contains information about degree and transition status coded together using hexadecimal:

$$ITDOF(8, I) = iord + 16 * iords(1) + 16^2 * iords(2) + 16^3 * iords(3). \quad (3.4)$$

Here $iord$ is the element degree and $iords(j)$ contains degree of jth edge of the element.[6]

Remark 3.32. *If 1-irregular rule and 2-neighbor rule are applied strictly in the*

[6]Note that $iord \leq 10 < 16$.

mesh, then a transition element has only one transition edge. Therefore, we could use

$$ITDOF(8, I) = 4 * iord + iside. \tag{3.5}$$

Here iside indicates transition edge. However, the formula 3.4 used in the current version of PLTMG*, later, allows us to relax the enforcement of 1-irregular rule and 2-neighbor rule in some cases.*

Explanation for $ITDOF$ is summarized in Table 3.2.

Table 3.2: ITDOF in current versions of PLTMG.

$ITDOF(1, I)$	pointer of first vertex
$ITDOF(2, I)$	pointer of second vertex
$ITDOF(3, I)$	pointer of third vertex
$ITDOF(4, I)$	plus-smallest/negative-biggest pointer of the first edge
$ITDOF(5, I)$	plus-smallest/negative-biggest pointer of the second edge
$ITDOF(6, I)$	plus-smallest/negative-biggest pointer of the third edge
$ITDOF(7, I)$	smallest interior pointer
$ITDOF(8, I)$	element degree and transition status

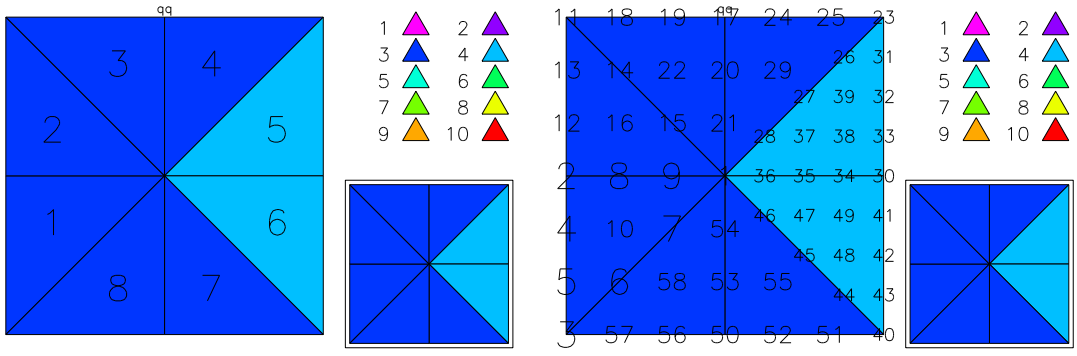


Figure 3.19: Example of dofs in a finite element mesh.

Remark 3.33. *For an edge, its counter-clockwise orientation in an element is opposite with that in the neighboring element. Therefore, if an edge is shared by two elements, then the pointers associated with it in the two elements are different (one is the smallest pointer and the other is negative of the biggest one).*

Table 3.3: ITDOF array.

NT	Vertices			Edges			Interior	Status
1	1	2	3	4	6	-9	10	13107
2	1	11	2	-13	8	-15	16	13107
3	1	17	11	-19	14	-21	22	13107
4	1	23	17	-25	20	-28	29	17203
5	1	30	23	-33	26	-36	37	17476
6	1	40	30	-43	34	-46	47	17476
7	1	50	40	-52	44	-54	55	13363
8	1	3	50	-57	53	-7	58	13107

3.3.3 *p*-Adaptive Refinement

The *p*-adaptive refinement algorithm we are using consists of four phases, namely, *marking*, *laying out new dofs*, *interpolating*, and *updating mesh status*.

Phase 1 - Marking: A posteriori error estimates are calculated based on the current solution on the current mesh. Then, all elements are placed in a max-heap data structure according to the size of their error estimates. The element with the largest error estimate is at the top of the heap. This element is marked for refinement and put in a list. Elements in the list is examined one by one (first in first out) until the list is empty. If refinement of an element in the list causes a violation of 1-irregular rule or 2-neighbor rule, and requires refinement of one of its neighboring element which has not been marked for refinement, then that neighboring element is marked for refinement and put at the end of the list. When the list is empty, the mesh is admissible (having no rule violation).

Phase 2 - Laying out dofs: In this phase, we go over every element in the mesh and redefine their dof pointers. The reason for this is that after some elements are refined the pointers in *ITDOF* no longer follow Rule 3.30. The pointers of dofs of elements are redefined from scratch according to Rule 3.29 and Rule 3.30.

Phase 3 - Interpolating: Since dofs (nodal points) of an element are changed after a p-refinement, the values of grid function associated with new dofs need to be computed in order to provide a good initial guess for iterative solver used to solve the system of linear equations.

Algorithm 4 p-Refinement: Phase 1 - Marking

Phase 1: Marking

Calculate error estimates.

Build a max-heap H of elements according to their error estimates.

$first \leftarrow 1$; $last \leftarrow 1$;

while $ndf < ndftgrt$ **do**

$L(first) \leftarrow H(1)$;

repeat

$t \leftarrow L(first)$

Mark t as a potential element for refinement

Update ndf

for $j = 1$ **to** 3 **do**

if *refinement of t required its neighbor t^j to be refined* **then**

$last \leftarrow last + 1$;

$L(last) = t^j$;

end if

end for

$first \leftarrow first + 1$;

until $first = last$

Set errors of marked element to be zeros and update the heap

end while

End

As stated in Chapter 2, we seek the finite element solution as a linear combination of basis functions:

$$f_{f.e} = \sum_{i=1}^{N} c_i \, \phi_i \tag{3.6}$$

Here the c_i are approximated values of the exact solution at the places of dofs in the mesh. These c_i are actually the values stored in the grid function array GF.

Of course, the identity 3.6 still holds if it is restricted on an element t. Also note that only basis functions associated with dofs of t have supports in t.

Therefore, we can write 3.6 as

$$f_{f.e}|_t = \sum_{i=1}^{N_p} c_{n_i}\ \phi_{n_i}^{(p)}|_t, \tag{3.7}$$

where p is the degree of t. If t is p-refined then we would want to find coefficients $\tilde{c}_{m_i}$ such that

$$\sum_{i=1}^{N_p} c_{n_i}\ \phi_{n_i}^{(p)}|_t = \sum_{i=1}^{N_{p+1}} \tilde{c}_{m_i}\ \phi_{m_i}^{(p+1)}|_t \tag{3.8}$$

(Note that, for the sake of simplicity, we assume t is a regular element before and after the p-refinement.) Clearly we can compute the values of $\tilde{c}_{m_i}$ by evaluating the left hand side of equation 3.8 at the dof m_i.

Phase 4 - Updating Mesh Status: In this phase, using the record of elements marked for refinement and the previous status of the mesh, the new degree and especially new transition status of every element are updated.

3.3.4 p-Adaptive Unrefinement

In a crude way, one could apply Algorithm 4 (of p-adaptive refinement) to marking elements to p-unrefined with only one change: constructing H as a min-heap of errors of elements instead of a max-heap of them. With this approach, elements with small errors are marked for p-refining first. However, comparing approximate quantities of small scale is not very reliable. To avoid this problem, we currently use the same max-heap as in p-refinement. The only difference is that the "p-refinement" is carried on a reduced mesh where degree of elements, except linear ones, are reduced by one. The elements which are not marked in the "p-refinement" are the ones marked for p-unrefinement.

The other phases of p-unrefinement are very much identical to those of p-refinement except for small changes in implementation to reduce degrees of elements instead of increasing them.

3.4 *hp*-Adaptive Meshing

As h-refinement and p-refinement can be used independently, we can perform hp-adaptive refinement by alternating the two versions of refinement in an arbitrary order. Alternating use of the h-version and p-version of refinement usually results in good meshes and shows exponential rate of convergence in many cases (as predicted in [39, 40]). However, the performance of the H^1 error estimate degenerates for problems with singularities or local rapid changes in the solution. This is mainly due to the fact that p-refinement tends to use elements of high degree in critical regions (near singularities or associated with rapid changes of the solution) while using elements of smaller size (h-refinement) usually yields better accuracy in these regions. In addition, alternating h-refinement and p-refinement is expensive since it requires solving the whole problem in between any two refinements. In this section, we discuss how to combine h-refinement and p-refinement to better capture the behavior of the solution and reduce the number of solves before desired accuracy is achieved.

The discussion of how to identify the regions of singularities or with rapid changes in the solution is presented in section 4.4 (hp-refinement indicator). In this section, we assume that a Boolean-valued function $PTEST$, which decides whether an element should be p-refined or h-refined, is available. Similar to the adaptive strategies which have been discussed, local errors for each elements are calculated. According to their local errors, elements are put in a max-heap. Then elements with large errors are selected to be refined first and refinement type is decided using the function $PTEST$.

In standard adaptive meshing, elements are marked for refined/unrefined before they are actually refined/unrefined altogether in the end and mesh information is updated. This is very efficient since mesh data need to be updated only once. Unfortunately, we cannot use the same procedure for automatic hp-refinement since any type of refinement can happen anytime to any element. In automatic hp-refinement, when an element is selected to be refined, its refinement and possibly other subsequent refinements need to be reflected in the mesh before another element can be selected from the heap for the next refinement. Detail of

the strategy for automatic *hp*-refinement is described in Algorithm 5.

Note that since there presently is no way to update error estimates of an element after its *p*-refinement, the elements who have been *p*-refined are not allowed to be refined any further.

Algorithm 5 Automatic *hp*-Refinement

Phase 1: Marking

Calculate error estimates.

Build a max-heap H of elements according to their error estimates.

while $ndf < ndftgrt$ **do**

$t \leftarrow H(1)$;

if $error(t) = 0$ **then**

Stop

end if

if $PTEST(t)$ **then**

p-refine t

Lay out new dofs

Interpolate values for dofs associated with t

Update mesh status

Set $error(t)$ to be zero

else

h-refine t

Update mesh status

Estimate errors for new elements

end if

Update error heap H

end while

End

Numerical results demonstrate that our automatic *hp*-refinement produces optimal meshes which deliver exponential rate of convergence in most cases (see the first three sections of Chapter 6 for examples).

In PLTMG, we also implement automatic *hp*-unrefinement with the same

strategy except that the max-heap of errors of elements is replaced by the min-heap of them. Numerical experiments using automatic *hp*-unrefinement have not been as successful as automatic *hp*-refinement. This is mainly due to the fact that we have to use unreliable information from comparing errors of small scale. More detail about this is discussed previously in subsection 3.3.4 (*p*-adaptive unrefinement).

Chapter 4

Derivative Recovery and Error Estimates

In adaptive FEMs, error estimates play a very important role. They provide FEMs with not only theoretical foundation but also practical techniques. There are two different types of error estimates: a priori error estimates and a posteriori error estimates. A priori error estimates are estimates that can be obtained from the hypotheses of the PDEs problem, even before solving the problem. Although these estimates are usually idealistic, they show potentials of FEMs and provide useful information of the accuracy we should expect when a PDE is solved by FEMs. A posteriori error estimates, on the other hand, are estimates based on information from the currently computed finite element solution. Global versions of these estimates can be used as a measurement for quality of approximate solution while local versions of them can be used to guide adaptive meshing. The main purpose of this chapter is to formulate a posteriori error estimate for finite elements using derivative recovery technique developed by Bank and Xu (see [27, 28]).

The rest of this chapter is organized as follows: Section 4.1 gives a brief overview of error estimates for p- and hp-version of FEMs. Potentials of p- and hp-version of FEMs shown in this section are also the motivations of the dissertation. In section 4.2, we introduce and discuss the derivative recovery technique, first for linear case and then the general case. The superconvergent results in this section is then used to formulate a posteriori error estimates in section 4.3. The last section,

4.4, is devoted to answering a very interesting question in *hp*-adaptive meshing; that is, how to use a posteriori error estimates to decide if it is better to refine a given element into several child element (*h*-refinement), or increasing its degree (*p*-refinement).

4.1 Overview of Error Estimates

According to Babuška, the study of *p*-version of FEMs was initiated at the School of Engineering and Applied Science of Washington University in St. Louis in 1970. However, the first publication on the *p*-version of the FEMs was in 1981 by Babuška, Szabo and Katz (see [11]). In the paper, the authors considered the following model problem

$$-\Delta u + u = f \quad \text{in } \Omega_0, \tag{4.1a}$$

$$\Gamma u = 0 \quad \text{on } \partial\Omega_0, \tag{4.1b}$$

where Ω_0 is a bounded polygonal domain, $f \in H^0(\Omega_0)$, and $\Gamma u = u$ or $\Gamma u = \partial u/\partial n$.

The following is a priori estimate of the *p*-version of FEMs that demonstrates exponential rate of convergence in term of degree p.

Theorem 4.1. *Let $u \in H^k(\Omega_0)$, where $k > 1$, be the solution of the problem 4.1 and let $u_p \in \mathcal{P}_p(\mathcal{T}_h)$ be the finite element approximation. Here $\mathcal{P}_p(\mathcal{T}_h)$ is the space of C^0 piecewise polynomials of degree up to p defined on the triangulation $\mathcal{T}_h$ of Ω_0. Then*

$$\|u - u_p\|_{1,\Omega_0} \leq C(k,\epsilon) p^{-(k-1)+\epsilon} \|u\|_{k,\Omega_0}, \tag{4.2}$$

for any $\epsilon > 0$. If the Neumann boundary conditions are under consideration, namely $\Gamma = \partial u/\partial n$, ϵ can be taken to be zero.

Assume that N is the number of degrees of freedom, in other words, $\mathcal{P}_p(\mathcal{T}_h)$ is of dimension N. Since we are considering the *p*-version of FEMs, the number of elements in the triangulation $\mathcal{T}_h$ is fixed. Therefore, N can be assumed to be proportional to p^2: $N \approx p^2$, and Estimate 4.2 can be rewritten as

$$\|u - u_p\|_{1,\Omega_0} \leq C(k,\epsilon) N^{-(k-1)/2+\epsilon} \|u\|_{k,\Omega_0}. \tag{4.3}$$

While with traditional finite element methods on a quasi-uniform mesh, we have

$$\|u - u_h\|_{1,\Omega_0} \leq Ch^{\min\{k-1,p\}}\|u\|_{k,\Omega_0}. \tag{4.4}$$

In this case, the number of degrees of freedom can be approximated as $N \approx h^{-2}$ (note that our PDE problem is posed in $\mathbb{R}^2$). Hence, Equation 4.4 can be rewritten in the following form:

$$\|u - u_h\|_{1,\Omega_0} \leq CN^{-\min\{k-1,p\}/2}\|u\|_{k,\Omega_0}. \tag{4.5}$$

Since 4.5 is known to be optimal (up to an arbitrarily small $\epsilon > 0$; see [6]), the rate of convergence of the p-version is comparable with that of the h-version. In addition, the convergence rate of the h-version is restricted by p the upper bound of the degree of the elements in the triangulation, while there is no such restriction for the p-version.

In addition, numerical results in [11] show that for problems with smooth solutions, the p-version has at least the same rate of convergence as the h-version. For problems with corner singularities, the rate of convergence of the p-version is twice that of the h-version.

Remark 4.2. *In 1987, Babuška and Suri showed that the term ϵ in 4.2 and 4.3 appears only owing to technicalities in the original proof, and can be removed (see [10]).*

Not long after the first paper on the p-version appeared, in the same year 1981, Babuška and Dorr proved the estimates which show the simultaneous dependence of the order of approximation on both the element degrees and the size of the mesh (see [2]). This result showed the potential of combining the two versions of FEMs to have the so-called hp-FEMs.

Let $\mathcal{T} = \{\tau_i\}_{i=1}^N$ be a triangulation of the domain Ω, where τ_i denote triangles of $\mathcal{T}$. For $\boldsymbol{p} = (p_1, p_2, \ldots, p_N)$ of positive integers, and let $\mathcal{P}(\mathcal{T}, \boldsymbol{p})$ be the maximal set of functions in H_0^1 such that $\mathcal{P}(\mathcal{T}, \boldsymbol{p})|_{\tau_i} \subseteq \mathcal{P}_{p_i}(\tau_i)$ for $1 \leq i \leq N$. The vector $\boldsymbol{p}$ is called the degree of $\mathcal{P}(\mathcal{T}, \boldsymbol{p})$. There is another vector $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ called order of $\mathcal{P}(\mathcal{T}, \boldsymbol{p})$, such that for each $i, 1 \leq i \leq N$,

$$\sigma_i = \max\{p : \ \mathcal{P}(\mathcal{T}, \boldsymbol{p})|_{\tau_i} \supset \mathcal{P}_p(\tau_i) \cap H_0^1(\Omega)\}$$

Now for any $u \in H_0^1(\Omega)$ let

$$Z(\mathcal{T}, \boldsymbol{p}, u) = \inf_{v \in \mathcal{P}(\mathcal{T}, \boldsymbol{p})} \|u - v\|_{H^1(\Omega)}.$$

The following is the main result of [2].

Theorem 4.3. *Denote* $\boldsymbol{p} = (p_1, p_2, \ldots, p_n)$, $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_n)$, *and* $\boldsymbol{m} = (m_1, m_2, \ldots, m_n)$, *where* $p_i > 2$, $2 \le m_i \le m$ *for* $1 \le i \le N$. *Also let* $u \in H_0^1(\Omega)$ *and* $u|_{\tau_i} \in H^{m_i}(\tau_i)$ *for* $i = 1, \ldots, N$. *Then*

$$Z^2(\mathcal{T}, \boldsymbol{p}, u) \le C(m) \sum_{i=1}^{N} h_i^{2[\min(\sigma_i - 1, m_i - 2) - 1]} \sigma_i^{-2(m_i - 2)} \|u\|^2_{H^{m_i}(\tau_i)}$$

In particular, when $p_i = p$ for all i, Babuška and Dorr proved the following result.

Theorem 4.4. *Let* $u \in H_0^1(\Omega) \cap H^m(\Omega)$, $m > 1$. *Then given any* ϵ, $0 < \epsilon < \min(1, m - 1)$,

$$Z(\mathcal{T}, \boldsymbol{p}, u) \le C(m, \epsilon) h^{\min(p, m-1) - \epsilon} p^{-(m-1) + \epsilon} \|u\|_{H^m(\Omega)},$$

where h *is the size of the mesh and* C *is independent of* p, T, *and* u.

Even though results in Theorem 4.3 and Theorem 4.4 are stated for approximation errors $Z(\mathcal{T}, \boldsymbol{p}, u)$, the approximation errors could be converted into finite element error using Céa's Lemma. However, even in the form using finite element error, the result stated in Theorem 4.3 is still difficult to verify by numerical experiments. Therefore, there was a desire for estimates similar to that in 4.3 for the *hp*-version. Such estimates were not available until 1986 when Guo and Babuška published [39, 40]. These papers show that if $u \in \mathcal{B}^2_\beta(\Omega)$ then the *hp*-version leads to the exponential rate of convergence

$$\|e\|_{H^1(\Omega)} \le C \exp(-bN^{1/3}). \tag{4.6}$$

where e is the finite element error, N is the number of degree of freedom, C and b are independent of N, but dependent on the mesh and the solution.[1]

[1] $\mathcal{B}^2_\beta(\Omega)$ is a normed space defined using weighted Sobolev spaces and distance functions (a Besov space). Under normal condition, solutions of elliptic problems have been shown to belong to $\mathcal{B}^2_\beta(\Omega)$.

This result still holds with the presence of singularities as long as these singularities belong to the elements boundary of Ω. In particular, if the singularities are located outside of the domain then better rate of convergence can be achieved:

$$\|e\|_{H^1(\Omega)} \leq C \exp(-bN^{1/2}). \tag{4.7}$$

Numerical results show that 4.6 and 4.7 hold, however, only when the geometry of the mesh and degrees of elements are properly chosen. This drawback poses a very interesting question; that is, how to construct such meshes (using h-adaptive refinements and p-adaptive refinements) to achieve the optimal exponential rate of convergence. We address this issue later in section 4.4.

So far, we have been considered only a priori error estimates. These estimates can give asymptotic rates of convergence as the mesh size h tends to zero or the degrees of the elements in the mesh are sufficiently large. However, these estimates often cannot provide much practical information about the actual errors encountered on a given mesh with size h and degree distribution $\boldsymbol{p}$. A posteriori error estimates, on the other hand, can provide the user of finite element package with such information, enhancing the robustness of FEM package, and the reliability of approximations they produce. In addition, a posteriori error estimates are successfully used as guidance for adaptive meshing.

Ideally, one would like to construct error estimates such that the following conditions hold:

1. the error estimators are computable purely based on given data and an existing approximated (finite element) solution of the problem,

2. the global estimates behave like the global error in a suitable norm, namely there exists constants C_1 and C_2 such that

$$C_1\epsilon \leq \|e\| \leq C_2\epsilon, \tag{4.8}$$

where ϵ is the global error estimator and e is the actual global approximation error. Bounds like the ones in 4.8 are essential in an effective adaptive process. The upper bound provides reliability while the lower one provides efficiency. Together they

ensure that the actual error and the error estimator decrease (or increase) at the same order of the approximation error. In particular, such bounds ensure that if some adaptive procedure is implemented which can be shown to decrease ϵ then $\|e\|$ also decrease as the same rate as ϵ. Also when bounds such as 4.8 are available, one could use ϵ to decide whether the current solution is sufficiently accurate.

Unfortunately, bounds such as 4.8, in general, are not available with constant C_1, C_2 independent of the mesh parameter h and $\boldsymbol{p}$. However, it is possible to establish such bounds asymptotically, for example for sufficient small h and large $\boldsymbol{p}$.

In the following, we list some typical techniques for a posteriori estimation:

(1) *Element residual methods*: were proposed independently by Bank and Weiser [26] and Oden at al. [47]. In this type of method, the residual is used as a mean to measure how much the approximate solution fails to satisfy the governing differential equation and the boundary conditions. Residual of a numerical solution is computed over each element and used as data in special (element-wise) Neumann problem for local error estimator.

(2) *Subdomain residual methods*: is similar to element residual method. The differences are the local Dirichlet problem for error in a given element is formulated over a patch of surrounding elements and local error estimates are computed based on the local residual and the jump in normal derivative of computed solution at inter-element boundaries. These methods were first introduced by Babuška and Rheinboldt in [7].

(3) *Duality methods*: can be used for self-adjoint elliptic problems. These methods exploit the notions of duality theory in convex optimization, in which a primal and a dual problem for element error are solved that provide upper bound and lower bound of local error elemementwise.

(4)*Interpolation methods*: used the interpolation theory of finite element in Sobolev norms to produce rapid (and sometime crude) estimates of the local error for individual elements.

(5) *Dual weighted residual methods*: were introduced by Rannacher in [49]. In this type of method, the adjoint problem is formed and solved together with

the original differential problem. Then, the quantitative information about the global dependence of the error quantity on the local residual associated with each element is obtained from the solution of the adjoint problem.

These approaches were first introduced for classical finite element methods (the h-version), then extended for hp-version (see [46] and [42]). Even though they all demonstrate certain successes in providing an efficient a posteriori error estimates, they also have their own limitations. For example, element residual methods and subdomain-residual methods require special implementation for each problem class; duality methods and dual weighted residual methods require one to solve another differential equation; and interpolation methods might not work well for meshes with high degree elements. In addition, in order to guarantee the accuracy of error estimator, the primal problem in duality methods is required to be solved with higher order of convergence than the primal problem (higher p for the h-version, smaller h for the p-version, or combination of both for the hp-version). Besides, dual weighted residual methods can be used for "goal-oriented error estimate"; however, they require the user to have certain knowledge on setting up the appropriated dual problem and implement it in the code.

In this dissertation, we extend the work of Bank and Xu [27, 28] to use derivative recovery technique to construct a posteriori error estimate especially for the hp-version of FEMs. This approach can be considered as a post-processing method where information from currently computed solution after some post-processing procedures is used to construct error estimates. Before studying in detail this error estimator, we discuss the essential ingredient of our error estimators: derivative recovery technique.

4.2 Derivative Recovery

We consider problem having the following weak form: find $u \in H^1(\Omega)$ such that

$$B(u,v) = \int_\Omega (\mathcal{D}\nabla u + \boldsymbol{b}u) \cdot \nabla v + cuvdx = f(v) \tag{4.9}$$

for all $v \in H^1(\Omega)$. Here $\mathcal{D}$ is a 2×2 symmetric, positive definite matrix function, $\boldsymbol{b}$ is a vector function, c is a scalar function, and $f(.)$ is a linear functional. We assume that all the coefficients functions are smooth. Besides, note that since $H^1(\Omega)$ is chosen as trial space, the Neumann boundary condition is implicitly embedded.

In order to guarantee the uniqueness of the solution of Equation 4.9, we impose the continuity condition on the bilinear form $B(\cdot,\cdot)$:

$$\|B(u,v)\| \leq \nu \|u\|_{1,\Omega}\|v\|_{1,\Omega} \tag{4.10}$$

for all $u, v \in H^1(\Omega)$. We also assume the inf-sup conditions

$$\inf_{v\in H^1}\sup_{u\in H^1}\frac{B(u,v)}{\|u\|_{1,\Omega}\|v\|_{1,\Omega}} = \sup_{u\in H^1}\inf_{v\in H^1}\frac{B(u,v)}{\|u\|_{1,\Omega}\|v\|_{1,\Omega}} \geq \mu > 0 \tag{4.11}$$

Let $\mathcal{T}_h$ be a geometrically admissible triangulation of Ω that has good shape regularity quality and is of size h. Denote $\mathcal{V}_h^{(p)}$ the space of continuous piecewise polynomials of degree up to p associated with $\mathcal{T}_h$.

4.2.1 Gradient Recovery

In this subsection, we limit the discussion in the case of classical finite element methods where all elements are linear. And we would like to approximate the gradients of the exact solution using the information from the currently computed finite element solution.

In this case, we use the space of continuous piecewise linear polynomials associated with $\mathcal{T}_h$, $\mathcal{V}_h^{(1)} \subset H^1(\Omega)$ as the finite element space and consider the approximate problem: find $u_h \in \mathcal{V}_h^{(1)}$ such that

$$B(u_h, v_h) = f(v_h) \tag{4.12}$$

for all $v_h \in \mathcal{V}_h^{(1)}$.

To guarantee Equation 4.12 has unique solution, we assume that the inf-sup condition also holds for the subspace $\mathcal{V}_h^{(1)}$:

$$\inf_{v\in\mathcal{V}_h^{(1)}}\sup_{u\in\mathcal{V}_h^{(1)}}\frac{B(u,v)}{\|u\|_{1,\Omega}\|v\|_{1,\Omega}} = \sup_{u\in\mathcal{V}_h^{(1)}}\inf_{v\in\mathcal{V}_h^{(1)}}\frac{B(u,v)}{\|u\|_{1,\Omega}\|v\|_{1,\Omega}} \geq \mu > 0. \tag{4.13}$$

For a given function $u \in L^2(\Omega)$, we define its L^2 projection $Q_h u \in \mathcal{V}_h^{(1)}$ as the solution of the following variational problem

$$(Q_h u, v_h) = (u, v_h),\ \forall v_h \in \mathcal{V}_h^{(1)}. \tag{4.14}$$

Here $(\cdot, \cdot)$ denotes the inner product on $L^2(\Omega)$.

Now assume that e is an interior edge in $\mathcal{T}_h$, and let τ and τ' be the two elements sharing edge e. We say τ and τ' form an $O(h^2)$ approximate parallelogram if the length of any two opposite edges of them only differ by $O(h^2)$. Now let x be a vertex on $\partial\Omega$, and let e and e' be the two boundary edges sharing x as an end point. Let τ and τ' be the two element having e and e', respectively, as edges, and let $\boldsymbol{t}$ and $\boldsymbol{t'}$ be the unit tangents of e and e'. Take e and e' as one pair of corresponding edges, and make a clockwise traversal of edges of τ and τ' to define two additional corresponding pairs. Here we say that τ and τ' form an $O(h^2)$ parallelogram if $|\boldsymbol{t}-\boldsymbol{t'}| = O(h^2)$, and that the lengths of two corresponding edges differ only by $O(h^2)$.

Definition 4.5. *The triangulation $\mathcal{T}_h$ is $O(h^{2\sigma})$ irregular if:*

1. *The set $\mathcal{E}$ of interior edges in $\mathcal{T}_h$ can be decomposed as $\mathcal{E} = \mathcal{E}_1 + \mathcal{E}_2$ so that for each $e \in \mathcal{E}_1$, τ and τ' form an $O(h^2)$ approximate parallelogram, while $\sum_{e\in\mathcal{E}_2}(|\tau| + |\tau'|) = O(h^{2\sigma})$.*

2. *The set $\mathcal{P}$ of boundary vertices can be decomposed as $\mathcal{P} = \mathcal{P}_1 \oplus \mathcal{P}_2$ so that each pair of elements associated with $x \in \mathcal{P}_1$ forms an $O(h^2)$ approximate parallelogram and $|\mathcal{P}_2| = \kappa$, where κ is fixed and independent of h.*

The set of boundary point P and the decomposition $\mathcal{P} = \mathcal{P}_1 \oplus \mathcal{P}_2$ are used only in the case of Neumann boundary conditions. Generally speaking, we expect $\mathcal{P}_2$ to consist of the geometric corners of Ω and perhaps a few other isolated points.

Now for each element τ, we define a constant matrix $\mathcal{D}_\tau$ as the "average" of the diffusion matrix. More precisely

$$\mathcal{D}_{\tau ij} = \frac{1}{|\tau|}\int_\tau \mathcal{D}_{ij} dx. \tag{4.15}$$

Since $\mathcal{D}$ is positive definite, so is $\mathcal{D}_\tau$.

The following results are proved in [27].

Theorem 4.6. *Assume that the triangulation $\mathcal{T}_h$ is $O(h^2)$ irregular and $\mathcal{D}_\tau$ defined above satisfies*

$$|\mathcal{D}_{\tau ij}| \lesssim 1,$$
$$|\mathcal{D}_{\tau ij} - \mathcal{D}_{\tau' ij}| \lesssim h,$$

for $i = 1, 2,\ j = 1, 2$ and τ and τ' is a pair of triangles sharing a common edge. Assume the solution of 4.9 satisfies $u \in W^{3,\infty}(\Omega)$. Let u_I be the linear interpolant of u associated with $\mathcal{T}_h$ and u_h be the solution of 4.12. Then

$$\|\nabla u_h - \nabla u_I\|_{0,\Omega} \lesssim h^{1+\min(1,\sigma)} |\log h|^{1/2} \|u\|_{3,\infty,\Omega} \tag{4.16}$$
$$\|\nabla u - Q_h \nabla u_I\|_{0,\Omega} \lesssim h^{1+\min(1,\sigma)} |\log h|^{1/2} \|u\|_{3,\infty,\Omega} \tag{4.17}$$
$$\|\nabla u - Q_h \nabla u_h\|_{0,\Omega} \lesssim h^{1+\min(1,\sigma)} |\log h|^{1/2} \|u\|_{3,\infty,\Omega} \tag{4.18}$$

The estimate 4.16 is well-known for the special case $\sigma = \infty$, namely all pairs of adjacent triangles in $\mathcal{T}_h$ satisfy $O(h^2)$ approximate parallelogram property. It is also known for cases where the $O(h^2)$ approximate parallelogram property is satisfied except for triangles along a few lines or except for triangles along the boundary of the domain. Obviously these results are special cases for the results in Theorem 4.6.

In addition, the estimates in Theorem 4.6 are actually superconvergence results since $h^{1+\min(1,\sigma)} |\log h|^{1/2}$ can be thought of as $h^{1+\min(1,\sigma)-\epsilon}$ for any $\epsilon > 0$ arbitrarily small. However, the superconvergence will disappear if σ becomes very close to zero. Intuitively, this is mainly due to high frequency error introduced by the nonuniformities of the mesh. Here σ can be considered a measurement for the portion of unstructured triangles and when σ is close to zero the mesh is unstructured almost everywhere. In [28], Bank and Xu propose to use a multigrid-like operator to smoothen the high frequency terms. In the next step, we define this smoothing operator.

Let $a(\cdot, \cdot)$ be a linear form defined as follows:

$$a(u, v) = (\nabla u, \nabla v) + (u, v),$$

where $(\cdot, \cdot)$ is, as introduced before, the L^2 inner product.[2] By the Riesz represen-

[2]Note that the bilinear form $a(\cdot, \cdot)$ defined here is unrelated to the PDE problem we are solving.

tation theorem , $a(\cdot,\cdot)$ induces a bounded linear discrete operator $A_h : \mathcal{V}_h^{(1)} \to \mathcal{V}_h^{(1)}$ uniquely defined by:

$$(A_h u_h, v_h) = a(u_h, v_h),\ \forall\, u_h, v_h \in \mathcal{V}_h^{(1)} \tag{4.19}$$

Since $a(\cdot,\cdot)$ is symmetric, A_h is also symmetric. We further notice that A_h is positive on $\mathcal{V}_h^{(1)}$ and the spectral radius of A_h:

$$\lambda \equiv \rho(A_h) \simeq h^{-2}$$

Using A_h and λ, we introduce the smoothing operator S as follows:

$$S_h = I - \lambda^{-1} A_h,$$

where I is the identity operator.

We quote the results below from [28] without a proof.

Lemma 4.7. *For any $z \in \mathcal{V}_h^{(1)}$, $m \in \mathbb{Z}$,*

$$\|(I - S_h^m) z\|_{0,\Omega} \lesssim mh \left(\|z - \partial_i u\|_{1,\Omega} + h\|u\|_{3,\Omega} + h^{1/2} |u|_{2,\infty,\partial\Omega} \right)$$

Lemma 4.8. *Let $w \in H^1(\Omega)$ and assume $1/2 < \alpha \leq 1$. Then*

$$\|S_h^m Q_h \partial_i w\|_{0,\Omega} \lesssim \epsilon_m (h^{-1}\|w\|_{0,\Omega} + \|w\|_{1,\Omega} + h^{-\alpha}|w|_{2,\infty,\partial\Omega})$$

where $\epsilon_m = (1 - \kappa^{-1})^m$ with $\kappa = (Ch^2)\lambda$, for some constant C, and $m \in \mathbb{Z}$ satisfying $m < (\kappa - 1)\alpha/2$.

Theorem 4.9. *Let $u \in H^3(\Omega) \cap W^{2,\infty}(\Omega)$. Then for any $v_h \in \mathcal{V}_h^{(1)}$, and $1/2 < \alpha \leq 1$, we have*

$$\begin{aligned} \|\nabla u - S_h^m Q_h \nabla v_h\|_{0,\Omega} \lesssim\ & mh^{3/2} \left(h^{1/2}\|u\|_{3,\Omega} + |u|_{2,\infty,\partial\Omega} \right) \\ & + \epsilon_m \left(h^{-1}\|u - v_h\|_{0,\Omega} + \|u - v_h\|_{1,\Omega} + h^{-\alpha}\|u - v_h\|_{0,\infty,\partial\Omega} \right) \end{aligned}$$

where m, ϵ_m defined as in Lemma 4.8.

The following theorem is the main result of [28]

Theorem 4.10. *Assume $u \in H^3(\Omega) \cap W^{2,\infty}(\Omega)$ and let u_h be the finite element solution. Then*

$$\|\nabla u - S_h^m Q_h \nabla u_h\|_{0,\Omega} \lesssim h\left(mh^{1/2} + \epsilon_m\right)\left(\|u\|_{3,\Omega} + |u|_{2,\infty,\Omega}\right).$$

where m, ϵ_m defined as in Lemma 4.8 and $1/2 < \alpha < 1$.

Proof. For u_h being the finite element solution, we assume the standard estimates

$$\|u - u_h\|_{k,\Omega} \lesssim h^{2-k}|u|_{2,\Omega},\ k = 0, 1, \tag{4.20a}$$

$$\|u - u_h\|_{0,\infty,\Omega} \lesssim h^2 \left|\log h\right| |u|_{2,\infty,\Omega}. \tag{4.20b}$$

Thus for $1/2 < \alpha < 1$

$$h^{-\alpha}\|u-u_h\|_{0,\infty,\partial\Omega} \le h^{-\alpha}\|u-u_h\|_{0,\infty,\Omega} \lesssim h^{1-\alpha}\left|\log h\right|\ h\,|u|_{2,\infty,\Omega} \lesssim h|u|_{2,\infty,\Omega}. \tag{4.21}$$

Theorem 4.10 now follows directly from estimates 4.20, 4.21 and Theorem 4.9. □

Now we state a stronger version of Theorem 4.10 which combines the current result and the earlier result in Theorem 4.6.

Theorem 4.11. *Assume $u \in W^{3,\infty}(\Omega)$ and the hypotheses of Theorem 4.6. Then for m, ϵ_m defined as in Lemma 4.8 and $1/2 < \alpha < 1$, we have*

$$\|\nabla u - S_h^m Q_h \nabla u_I\|_{0,\Omega} \lesssim h\left(\min\left(h^{\min(1,\sigma)}|\log h|^{1/2}, \epsilon_m\right) + mh^{1/2}\right)\|u\|_{3,\infty,\Omega},$$

$$\|\nabla u - S_h^m Q_h \nabla u_h\|_{0,\Omega} \lesssim h\left(\min\left(h^{\min(1,\sigma)}|\log h|^{1/2}, \epsilon_m\right) + mh^{1/2}\right)\|u\|_{3,\infty,\Omega}.$$

Here u_h is the finite element solution and u_I is the Lagrange linear interpolant associated with the triangulation $\mathcal{T}_h$.

Proof. We only give a proof for the estimate with u_h as similar arguments also hold for the estimate with u_I.

By triangle inequality,

$$\|\partial_i u - S_h^m Q_h \partial_i u_h\|_{0,\Omega} \le \|(I-Q_h)\partial_i u\|_{0,\Omega} + \|(I-S_h^m)Q_h\partial_i u\|_{0,\Omega} + \|S_h^m Q_h \partial_i(u-u_h)\|_{0,\Omega}$$

The first term can be bounded by a standard estimation in approximation theory

$$\|(I - Q_h)\partial_i u\|_{k,\Omega} \lesssim h^{2-k}\|u\|_{3,\Omega},\ k = 0, 1. \tag{4.22}$$

The second term is estimated by applying Lemma 4.7 for $z = Q_h\partial_i u$

$$\|(I - S_h^m)Q_h\partial_i u\|_{0,\Omega} \lesssim mh\left(\|(I - Q_h)\partial_i u\|_{1,\Omega} + h\|u\|_{3,\Omega} + h^{1/2}|u|_{2,\infty,\partial\Omega}\right) \quad (4.23)$$

Note that the first term of 4.23 can also be bounded using the standard approximation estimate mentioned above.

As for the third term, we can bound it in two different ways. First, we apply Lemma 4.8 for $w = u - u_h$

$$\|S_h^m Q_h\partial_i(u-u_h)\|_{0,\Omega} \lesssim \epsilon_m(h^{-1}\|u-u_h\|_{0,\Omega}+\|u-u_h\|_{1,\Omega}+h^{-\alpha}|u-u_h|_{2,\infty,\partial\Omega}). \quad (4.24)$$

By arguments similar to those in the proof of Theorem 4.10, 4.24 becomes

$$\|S_h^m Q_h\partial_i(u - u_h)\|_{0,\Omega} \lesssim \epsilon_m h(|u|_{2,\Omega} + |u|_{2,\infty,\Omega}), \quad (4.25)$$

Second, by triangle inequality

$$\begin{aligned}\|S_h^m Q_h\partial_i(u - u_h)\|_{0,\Omega} &\lesssim \|Q_h\partial_i(u - u_h)\|_{0,\Omega} \\ &\lesssim \|\partial u_i - Q_h\partial_i u_h\|_{0,\Omega} + \|(I - Q_h)\partial_i u\|_{0,\Omega} \\ &\lesssim Ch^{1+\min(1,\sigma)}\,|\log h|^{1/2}\,\|u\|_{3,\infty,\Omega} + h^2\|u\|_{3,\Omega} \quad (4.26)\end{aligned}$$

Combining 4.22, 4.24, 4.26 and noting that all the norms in the right hand sides of these estimates can be bounded by $\|\cdot\|_{3,\infty,\Omega}$, we have

$$\|\nabla u - S_h^m Q_h\nabla u_h\|_{0,\Omega} \lesssim h\left(\min\left(h^{\min(1,\sigma)}|\log h|^{1/2}, \epsilon_m\right) + mh^{1/2}\right)\|u\|_{3,\infty,\Omega}$$

□

We have shown in Theorem 4.10 that ∇u can approximately be recovered by $S_h^m Q_h\nabla u_h$. Note ∇u_h is a piecewise constant functions associated with $\mathcal{T}_h$ since u_h is a continuous piecewise linear function (in the current case). In particular, ∇u_h is likely to be discontinuous along the boundaries of the elements in $\mathcal{T}_h$. Denote R the operator defined by

$$R(z) = S_h^m Q_h z$$

for any (discontinuous) piecewise constant function z associated with the triangulation $\mathcal{T}_h$. We call R the *recovery operator*.

The operator R can be used to recover not only the first derivatives but also second derivatives of u. Details are in the theorem below.

Theorem 4.12. *Assume the hypotheses of Theorem 4.10. We have*

$$\|\partial_i(\partial_k u - R(\partial_k u_h))\|_{0,\Omega} \lesssim \left(\min\left(h^{\min(1,\sigma)}|\log h|^{1/2}, \epsilon_m\right) + mh^{1/2}\right)\|u\|_{3,\infty,\Omega}$$

where m, ϵ_m defined as in Lemma 4.8 and $1/2 < \alpha < 1$.

4.2.2 Derivative Recovery

In this subsection, we would like to extend the work in the previous section so that we could recovery the $(p+1)$th derivatives of the exact solution when elements are of degree p arbitrary.

In [29], Bank et al. show that the framework used in subsection 4.2.1 can be generalized for meshes with elements of arbitrary degree without significant changes. We quote the following result from [29] without a proof.

Theorem 4.13. *Let $u \in H^{p+2}(\Omega) \cap W^{p+1,\infty}(\Omega)$ and $u_h \in \mathcal{V}_h^{(p)}$ be an approximation of u satisfying*

$$\|u - u_h\|'_{p-1,\Omega} \lesssim h^2|u|_{p+1,\Omega}, \tag{4.27a}$$

$$\|u - u_h\|'_{p-1,\infty,\Omega} \lesssim h^2\,|\log h|\,|u|_{p+1,\infty,\Omega}. \tag{4.27b}$$

where $\|\cdot\|'_{\cdot,\Omega}$ is the discrete norm defined by $\|\cdot\|'_{\cdot,\Omega} = \sum_{\tau\in\mathcal{T}_h}\|\cdot\|_{\cdot,\tau}$. Then

$$\|\partial^p u - R(\partial^p u_h)\|_{0,\Omega} \lesssim h\left(mh^{1/2} + \epsilon_m\right)\left(\|u\|_{p+2,\Omega} + |u|_{p+1,\infty,\Omega}\right),$$

here m, ϵ_m defined as in Lemma 4.8 and $1/2 < \alpha < 1$.

Remark 4.14. *Note that the estimates in 4.27 are standard for $u_h \in \mathcal{V}_h^{(p)}$ being the finite element solution.*

Using the result from Theorem 4.13, we can derive approximations for the $(p+1)$th derivatives of u.

Theorem 4.15. *Assume the hypotheses of Theorem 4.13. Then*

$$\|\partial(\partial^p u - R(\partial^p u_h))\|_{0,\Omega} \lesssim \left(mh^{1/2} + \epsilon_m\right)\left(\|u\|_{p+2,\Omega} + |u|_{p+1,\infty,\Omega}\right),$$

where m, ϵ_m defined as in Lemma 4.8 and $1/2 < \alpha < 1$.

Proof. Let I_h be the linear approximation operator associated with the triangulation $\mathcal{T}_h$. Put $z = I_h \partial^p u \in \mathcal{V}_h^{(1)}$, then

$$\begin{aligned}
\|\partial(\partial^p u - R(\partial^p u_h))\|_{0,\Omega} &\leq \|\partial(\partial^p u - z)\|_{0,\Omega} + \|\partial(z - R(\partial^p u_h))\|_{0,\Omega} \\
&\lesssim h|u|_{p+2,\Omega} + h^{-1}\|z - R(\partial^p u_h)\|_{0,\Omega} \\
&\lesssim h|u|_{p+2,\Omega} + h^{-1}\left(\|z - \partial^p u\|_{0,\Omega} + \|\partial^p u - R(\partial^p u_h)\|_{0,\Omega}\right) \\
&\lesssim \left(mh^{1/2} + \epsilon_m\right)\left(\|u\|_{p+2,\Omega} + |u|_{p+1,\infty,\Omega}\right)
\end{aligned}$$

□

4.3 A Posteriori Error Estimates

The main purpose of this section is to formulate a posteriori estimates for the H^1 errors of the finite element solution. More precisely, we seek a good approximation for $\|u - u_h\|_{1,\Omega}$ that can be computed cheaply using the information of the current approximate solution[3]. Ideally, we would like to have two different kinds of estimates: global and local ones. Global estimates give us information on how well the current computed solution approximates the exact solution in general. This kind of estimates can be used to decide whether the current approximate solution is accurate enough. On the other hand, local estimates are estimates calculated locally for each element. They can be used as error indicators to guide adaptive mesh refinement to create an efficient refined mesh.

4.3.1 Linear Case

The obvious choice for a global a posteriori error estimate is to approximate $\|\nabla(u - u_h)\|_{0,\Omega}$ by $\|(I - R)\nabla u_h\|_{0,\Omega}$. The following theorem confirms $\|(I - R)\nabla u_h\|_{0,\Omega}$ is indeed a good approximation.

[3]It is well-known that the H^1 norm is comparable with the energy norm.

Theorem 4.16. *Assume the hypothesis of Theorem 4.11, we have*

$$\begin{aligned}\|\nabla(u-u_h)\|_{0,\Omega} \lesssim\ & \|(I-R)\nabla u_h\|_{0,\Omega} \\ & + Ch\left(\min\left(h^{\min(1,\sigma)}|\log h|^{1/2},\epsilon_m\right)+mh^{1/2}\right)\|u\|_{3,\infty,\Omega} \qquad (4.28)\\ \|(I-R)\nabla u_h\|_{0,\Omega} \lesssim\ & \|\nabla(u-u_h)\|_{0,\Omega} \\ & + Ch\left(\min\left(h^{\min(1,\sigma)}|\log h|^{1/2},\epsilon_m\right)+mh^{1/2}\right)\|u\|_{3,\infty,\Omega} \qquad (4.29)\end{aligned}$$

where m, ϵ_m defined as in Lemma 4.8 and $1/2 < \alpha < 1$. Furthermore, if there exists a positive constant $c_0(u)$ independent of h such that

$$\|\nabla(u-u_h)\|_{0,\Omega} \geq c_0(u)h \qquad (4.30)$$

then

$$\left|\frac{\|(I-R)\nabla u_h\|_{0,\Omega}}{\|\nabla(u-u_h)\|_{0,\Omega}}-1\right| \leq \left(\min\left(h^{\min(1,\sigma)}|\log h|^{1/2},\epsilon_m\right)+mh^{1/2}\right) \qquad (4.31)$$

Proof. By triangle inequality,

$$\begin{aligned}\|\nabla(u-u_h)\|_{0,\Omega} &\leq \|(I-R)\nabla u_h\|_{0,\Omega}+\|\nabla u-R(\nabla u_h)\|_{0,\Omega}\\ \|(I-R)\nabla u_h\|_{0,\Omega} &\leq \|\nabla(u-u_h)\|_{0,\Omega}+\|\nabla u-R(\nabla u_h)\|_{0,\Omega}.\end{aligned}$$

Combining these estimates with Theorem 4.11, we obtain (4.28) and (4.29).

Now the estimate (4.31) follows after (4.28), (4.29) and (4.30). □

From (4.28) and (4.29), it follows that $\|(I-R)\nabla u_h\|_{0,\Omega}$ is more than first order approximation of $\|\nabla(u-u_h)\|_{0,\Omega}$. In particular, given a superconvergent approximation to ∇u, one can expect the effectivity ratio $\|(I-R)\nabla u_h\|_{0,\Omega}/\|\nabla(u-u_h)\|_{0,\Omega}$ to be close to unity.

For local error indicator, an obvious choice would be using $\|(I-R)\nabla u_h\|_{0,\tau}$ for local error on given element τ. However, since $R = S_h^m Q_h$ is a global operator (both the L^2 projection operator Q_h and smoothing operator S_h are based on global calculation) we prefer another approach.

The following is the motivation for a more practical approach.

Let u_I be the Lagrange linear interpolant and u_q be the quadratic hierarchical extension.[4] Assume that the triangulation T_h is $O(h^{2\sigma})$, and other hypotheses

[4]Note that for quadratic case, the hierarchical extension is the same as the interpolant.

as in Theorem 4.6. Using the standard estimates in approximation theory and results of Theorem 4.6 we have

$$\begin{aligned}
\|\nabla(u-u_h)\|_{0,\Omega} &\leq \|\nabla(u-u_q)\|_{0,\Omega} + \|\nabla(u_q-u_I)\|_{0,\Omega} + \|\nabla(u_I-u_h)\|_{0,\Omega} \\
&\lesssim h^2|u|_{3,\Omega} + \|\nabla(u_q-u_I)\|_{0,\Omega} + h^{1+\min(1,\sigma)}|\log h|^{1/2}\|u\|_{3,\infty,\Omega} \\
&\lesssim \|\nabla(u_q-u_I)\|_{0,\Omega} + C(u)\, h^{1+\min(1,\sigma)}|\log h|^{1/2} \qquad (4.32a) \\
\|\nabla(u_q-u_I)\|_{0,\Omega} &\leq \|\nabla(u-u_q)\|_{0,\Omega} + \|\nabla(u-u_h)\|_{0,\Omega} + \|\nabla(u_I-u_h)\|_{0,\Omega} \\
&\lesssim h^2|u|_{3,\Omega} + \|\nabla(u-u_h)\|_{0,\Omega} + h^{1+\min(1,\sigma)}|\log h|^{1/2}\|u\|_{3,\infty,\Omega} \\
&\lesssim \|\nabla(u-u_h)\|_{0,\Omega} + \bar{C}(u)\, h^{1+\min(1,\sigma)}|\log h|^{1/2} \qquad (4.32b)
\end{aligned}$$

So $\|\nabla(u_q-u_I)\|_{0,\Omega}$ is at least a first order approximation of $\|\nabla(u-u_h)\|_{0,\Omega}$. In particular, when the mesh is structured ($\sigma \gg 0$), we have superconvergent phenomena and $\|\nabla(u_q-u_I)\|_{0,\Omega}$ is an asymptotically exact approximation of $\|\nabla(u-u_h)\|_{0,\Omega}$. In addition, both functions u_q and u_I can be locally defined on each element. Therefore, for given element τ, we choose to use $\|\nabla(u_q-u_I)\|_{0,\tau}$ as its local error indicator.

In the next step, we formulate a systematic way to estimate $\|\nabla(u_q-u_I)\|_{0,\tau}$ on an arbitrary element τ.

Lemma 4.17. *Assume that $\tau \in \mathcal{T}_h$ is an element having vertices $v_i = (x_i, y_i)$, $1 \leq i \leq 3$, oriented counterclockwise, and corresponding linear nodal basis functions $\{\phi_i\}_1^3$. Let $\{\boldsymbol{t}_i\}_{i=1}^3$ denote the unit tangent vectors (also with counterclockwise orientation), and $\{\ell_i\}_{i=1}^3$ the edge lengths (see Figure 4.1). Also let $(i-1, i, i+1)$ be a cyclic permutation of $(1,2,3)$ and $\psi_i = \phi_{i-1}\phi_{i+1}$ be the bump function associated with edge i, $1 \leq i \leq 3$, of τ. Then*

$$u_q - u_I\big|_\tau = \sum_{i=1}^{3} \ell_i^2 \boldsymbol{t_i}^t M_\tau \boldsymbol{t_i}\, \psi_i, \qquad (4.33)$$

where

$$M_\tau = -\frac{1}{2}\begin{pmatrix} \partial_{11}u_q & \partial_{12}u_q \\ \partial_{21}u_q & \partial_{22}u_q \end{pmatrix}. \qquad (4.34)$$

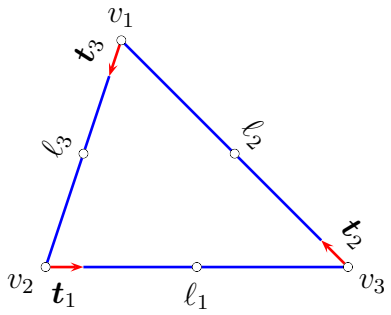


Figure 4.1: Parameters associated with element τ.

Proof. In τ, $u_q - u_I$ is a quadratic polynomial which vanishes at three vertices v_i of of τ. Therefore, we can write $u_q - u_I\big|_\tau$ as a linear combination of ψ_i:

$$u_q - u_I\big|_\tau = \sum \alpha_i \psi_i(x, y) \tag{4.35}$$

Evaluating (4.35) at m_i the midpoint of edge i gives us

$$\begin{aligned}\frac{1}{4}\alpha_i &= u_q(m_i) - u_I(m_i) \\ &= u_q(m_i) - \frac{1}{2}(u(v_{i-1}) + u(v_{i+1})) \quad \text{(by the linearity of } u_I)\end{aligned}$$

Here, we note that u_q has the same values with u at the vertices of τ. Thus

$$\alpha_i = -\ell_i^2 \cdot \frac{u_q(v_{i-1}) - 2u_q(m_i) + u_q(v_{i+1})}{(\frac{\ell_i}{2})^2}. \tag{4.36}$$

Also, the second factor of the right hand side of (4.36) is actually the second order difference quotient of u_q at m_i along the direction $\boldsymbol{t}_i$ of edge i. Since u_q is quadratic polynomial, the difference quotient is the exact value for second derivative of u_q along direction t_i. In other words

$$\frac{u_q(v_{i-1}) - 2u_q(m_i) + u_q(v_{i+1})}{(\frac{\ell_i}{2})^2} = \boldsymbol{t}_i{}^t \begin{pmatrix} \partial_{11}u_q & \partial_{12}u_q \\ \partial_{21}u_q & \partial_{22}u_q \end{pmatrix} \boldsymbol{t}_i \tag{4.37}$$

Now (4.33) and (4.34) follow from (4.35), (4.36) and (4.37). □

In the formula of $u_q - u_I$ (see (4.33)), everything can be computed from the geometric information of the element τ except the Hessian matrix M_τ. Since u_q is a high order approximation of the exact solution we expect its second derivatives

to be close to those of the exact solution. Based on Theorem 4.12, we choose to approximate the second derivatives in the Hessian matrix M_τ using derivatives of $R(\partial u_h)$. In particular, let

$$\tilde{M}_\tau = \begin{pmatrix} \partial_1 R(\partial_1 u_h) & \partial_1 R(\partial_2 u_h) \\ \partial_2 R(\partial_1 u_h) & \partial_2 R(\partial_2 u_h) \end{pmatrix}$$
$$\bar{M}_\tau = \frac{1}{2}\left(\tilde{M}_\tau + \tilde{M}_\tau^t\right).$$

Then, use $\bar{M}_\tau$ as the approximation for the Hessian since Hessian matrix is symmetric for smooth functions. Now the local error estimate ϵ_τ for element τ is given by

$$\epsilon_\tau = \sum_{i=1}^{3} \ell_i^2 \boldsymbol{t}_i^t \bar{M}_\tau \boldsymbol{t}_i \psi_i(x, y) \tag{4.38}$$

We expect this estimate to give good approximation for local error when the Hessian matrix for the true solution is well defined. In practice, we would like to use this local error estimate for a wide range of problems, including ones with singularity whose location is unknown. Since $\|(I-R)\nabla u_h\|_{0,\tau}$ is also a good approximation of $\|\nabla(u-u_h)\|_{0,\tau}$ (see Theorem 4.16), we use it to normalize the local error estimate ϵ_τ. The normalization constant (also called *scaling factor*) α_τ is chosen such that the *local error indicator* η_τ satisfies

$$\eta_\tau \equiv \alpha_\tau \|\nabla \epsilon_\tau\|_{0,\tau} = \|(I-R)\nabla u_h\|_{0,\tau} \tag{4.39}$$

Normally, we expect that $\alpha_\tau \approx 1$, which is likely to be the case where the Hessian matrix of the exact solution is well defined. When the Hessian is not defined, for example near singularities, α_τ provides improvement for the local error indicator ϵ_τ.

The local error estimate (4.38) is very useful in practice. It explicitly shows the dependence on the shape, size and orientation of the element as well as the dependence on the second derivatives of u. It also provides a simple, robust and elegant solution to an important practical problem for adaptive mesh refinement schemes: how to calculate error estimate for the refined elements without immediately resolving the global problem. More precisely, the children elements inherit

only the Hessian matrix from their parents and the geometrical information is derived from the refined elements themselves. With this approach, it is possible to have many levels of refinement before the approximation breaks down and a new global solution is required.

4.3.2 Quadratic Case

In this subsection, we consider quadratic finite elements. Similar to equation (4.33) in the linear case, we seek an expression for $\hat{u}_3 - u_2\big|_\tau$, where u_2 is the quadratic Lagrange interpolant and $\hat{u}_3$ is the cubic hierarchical extension.[5] Assume the same parameters associated with element τ as in Lemma 4.17 and Figure 4.1. Clearly, in τ, $\hat{u}_3 - u_2$ is a cubic polynomial which vanishes at vertices and edge midpoints of τ. Let $\mathcal{E}_3(\tau)$ be the space of cubic polynomials which are zero at vertices and edge midpoints of τ. In [29], Bank et al. propose a basis for $\mathcal{E}_3(\tau)$ given by

$$\psi_0 = \phi_1\phi_2\phi_3$$
$$\psi_i = \phi_{i-1}\phi_{i+1}(\phi_{i+1} - \phi_{i-1})$$

for $1 \le i \le 3$, and $(i-1, i, i+1)$ is a cyclic permutation of $(1, 2, 3)$. Then

$$\hat{u}_3 - u_2\big|_\tau = \frac{1}{12}\prod_{i=1}^{3}(\ell_{i+1}\partial_{\boldsymbol{t}_{i+1}} - \ell_{i-1}\partial_{\boldsymbol{t}_{i-1}})\hat{u}_3\psi_0 + \frac{1}{12}\sum_{i=1}^{3}\ell_i^3\partial_{\boldsymbol{t}_i}^3\hat{u}_3\psi_i, \tag{4.40}$$

where $\partial_{\boldsymbol{t}_i}u$ denotes the directional derivative of u in the direction $\boldsymbol{t}_i$. The directional derivatives in (4.40) can formally be written as standard third derivatives of $\hat{u}_3$. These standard third derivatives are then approximated by

$$\partial_{xxx}\hat{u}_3 \approx \partial_x R(\partial_{xx}u_h), \tag{4.41a}$$
$$\partial_{xxy}\hat{u}_3 \approx \frac{1}{2}\left(\partial_x R(\partial_{xy}u_h) + \partial_y R(\partial_{xx}u_h)\right), \tag{4.41b}$$
$$\partial_{xyy}\hat{u}_3 \approx \frac{1}{2}\left(\partial_x R(\partial_{yy}u_h) + \partial_y R(\partial_{xy}u_h)\right), \tag{4.41c}$$
$$\partial_{yyy}\hat{u}_3 \approx \partial_y R(\partial_{yy}u_h), \tag{4.41d}$$

[5] In cubic case, the hierarchical extension is no longer the same as the interpolant.

Let $\bar{u}$ be any cubic polynomial with third derivatives given by the right-hand sides of (4.41). Then we have *local error estimate* ϵ_τ given by

$$\bar{u}_3 - u_2\big|_\tau = \frac{1}{12}\prod_{i=1}^{3}(\ell_{i+1}\partial_{\boldsymbol{t}_{i+1}} - \ell_{i-1}\partial_{\boldsymbol{t}_{i-1}})\bar{u}_3\psi_0 + \frac{1}{12}\sum_{i=1}^{3}\ell_i^3\partial_{\boldsymbol{t}_i}^3\bar{u}_3\psi_i,$$

The *scaling factor* α_τ is chosen so that the *local error indicator* η_τ satisfies

$$\eta_\tau^2 \equiv \alpha_\tau^2|\epsilon_\tau|_{2,\tau} = \|(I-R)\partial_{xx}^2 u_h\|_{0,\tau}^2 + 2\|(I-R)\partial_{xy}^2 u_h\|_{0,\tau}^2 + \|(I-R)\partial_{yy}^2 u_h\|_{0,\tau}^2.$$

4.3.3 General Case

In this subsection, we formulate error estimates for finite elements of degree p arbitrary. Similar to the linear and quadratic case, we would like to approximate the error $|u-u_h|_{1,\tau}$ by $|\hat{u}_{p+1}-u_p|_{1,\tau}$, where u_p is the interpolant of degree p and $\hat{u}_{p+1}$ is the hierarchical extension of degree $p+1$.

Assume the same parameters associated with element τ as in Lemma 4.17 and Figure 4.1 and let

$$\mathcal{P}_{p+1}(\tau) = \mathcal{P}_p(\tau) \oplus \mathcal{E}_{p+1}(\tau)$$

where $\mathcal{E}_{p+1}(t)$ is the hierarchical extension space consists polynomials in $\mathcal{P}_{p+1}(\tau)$ that are zero at all degrees of freedom associated with $\mathcal{P}_p(t)$. Clearly $\hat{u}_{p+1} - u_p\big|_\tau \in \mathcal{E}_{p+1}$. In order to find an expression for $\hat{u}_{p+1} - u_p\big|_\tau$, we need to find a basis for $\mathcal{E}_{p+1}$. Since $\dim(\mathcal{P}_{p+1}(\tau)) = (p+2)(p+3)/2$ and $\dim(\mathcal{P}_p(\tau)) = (p+1)(p+2)/2$, $\dim(\mathcal{E}_{p+1}) = p+2$. Therefore, we need to find $p+2$ linearly independent polynomials of degree $p+1$ that vanish at all $(p+1)(p+2)/2$ degrees of freedom of degree p.

Intuitively, we would like to find these error basis functions as "product of lines" (each line represents a linear function). Since the basis functions are polynomials of degree $p+1$, we need to find a pattern for $p+1$ lines that going through all of the degrees of freedom of degree p. An example for the case $p=2$ is illustrated in Figure 4.2.

Of course, we do not want to define different patterns for different degrees. Ideally, we would like that the lines to be arranged in a systematic pattern that works for any degree (the pattern in Figure 4.2 is not a good candidate since there is

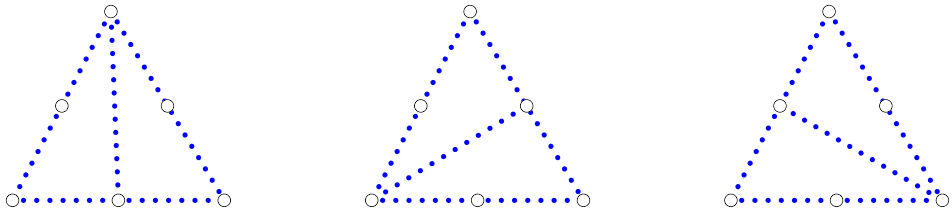


Figure 4.2: A pattern for elements of degree $p = 2$.

no easy way to generalize it for elements of higher degree). In addition, the pattern has to guarantee the independence of the basis functions it produces. Fortunately, we can come up with a magic pattern that fulfills these two requirements.

In our pattern, we choose two arbitrary edges of τ, for example, edge one and edge two, and only use lines in the directions of these edges (these lines also need to contain at least one degree of freedom of degree p). We start with a basis function which is a product of $p + 1$ lines in the direction of edge two. Then, we flip the furthermost line (from edge two) to the direction of edge one and put it on the lowest level available (from edge one). The process can be continued to procedure other functions in the set until all the lines are in the direction of edge one. Illustration for the case $p = 4$ is shown in Figure 4.3.

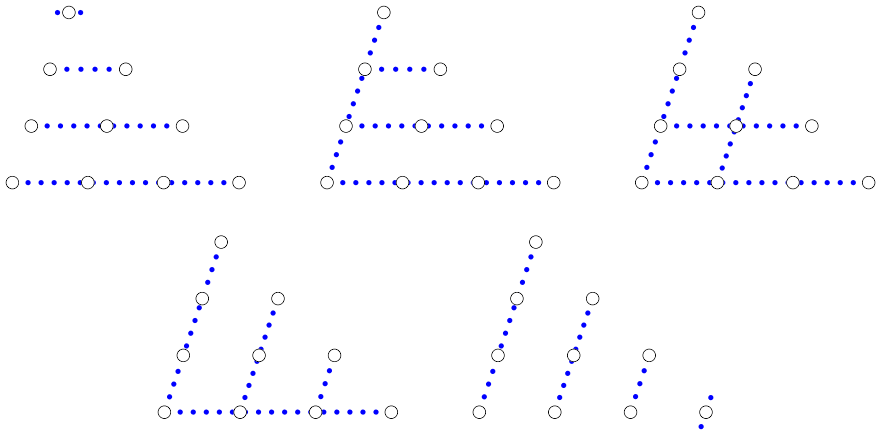


Figure 4.3: The magic pattern for the case $p = 4$.

Lemma 4.18. *For element τ of degree p, the following functions*

$$\psi_{p+1,i} = \prod_{j=0}^{i-1}(\phi_1 - j/p) \prod_{m=0}^{p-i}(\phi_2 - m/p),\ 0 \le i \le p+1$$

form a basis for $\mathcal{E}_{p+1}(\tau)$. Here ϕ_i is the ith linear nodal basis functions of τ (the ith barrycentric coordinate with respect to τ).

Proof. Clearly, the polynomials $\psi_{p+1,i}$ are of degree $p+1$ and vanish at all of the degrees of freedom of degree p. We only need to show that they are linearly independent.

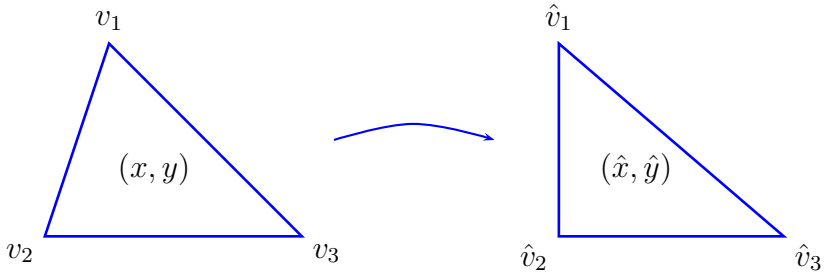


Figure 4.4: Change of coordinates.

Make an affine change of coordinates to $(\hat{x}, \hat{y})$ such that in the new coordinate system, τ is a right angled triangle with edge one on the $\hat{x}$-axis and edge two on the $\hat{y}$-axis (see Figure 4.4). In the new coordinate system, $\phi_1(\hat{x}, \hat{y}) = \hat{y}$ and $\phi_2(\hat{x}, \hat{y}) = \hat{x}$, and $\psi_{p+1,i}$ is a polynomial of degree $p+1$ having $\hat{x}^{p+1-i}\,\hat{y}^i$ as its only term of degree $p+1$.

Assume that there exists $\{c_i\}_{i=0}^{p+1}$ such that $\sum_{i=0}^{p+1} c_i \psi_{p+1,i} = 0$. Then the component of degree $p+1$ of $\sum_{i=0}^{p+1} c_i \psi_{p+1,i}$ (in the new coordinate system) is also zero, namely

$$\sum_{i=0}^{p+1} c_i \hat{x}^{p+1-i}\,\hat{y}^i = 0$$

Since $\{\hat{x}^i\,\hat{y}^{p+1-i}\}_{i=0}^{p+1}$ is known to be linearly independent, $c_i = 0$ for $0 \le i \le p+1$. This implies that $\{\psi_{p+1,i}\}_{i=0}^{p+1}$ is a linearly independent set. Therefore it is a basis of $\mathcal{E}_{p+1}$. □

Lemma 4.19. *Assume that the parameters of τ are labeled as in Lemma 4.17. Let $\{\psi_{p+1,i}\}_{i=0}^{p+1}$ be defined as in Lemma 4.18. Then*

$$\hat{u}_{p+1} - u_p\big|_\tau = \sum_{i=0}^{p+1} \ell_2^i (-\ell_1)^{p+i-1} \frac{\partial_{\boldsymbol{t}_2}^i \partial_{\boldsymbol{t}_1}^{p+1-i} \hat{u}_{p+1}}{i!(p+1-i)!} \psi_{p+1,i}. \tag{4.42}$$

Moreover, $\hat{u}_{p+1} - u_p\big|_\tau$ can be written as

$$\hat{u}_{p+1} - u_p\big|_\tau = \sum_{i=0}^{p+1} \sum_{j=0}^{p+1-i} \sum_{k=0}^{i} (-1)^{p+1-i} \alpha_{ijk} \left(\partial_x^{j+k} \partial_y^{p+1-j-k} \hat{u}_{p+1}\right) \psi_{p+1,i}, \tag{4.43}$$

where

$$\alpha_{ijk} = \binom{p+1-i}{j} \binom{i}{k} (x_1 - x_3)^k (y_1 - y_3)^{i-k} (x_3 - x_2)^j (y_3 - x_2)^{p+1-i-j}. \tag{4.44}$$

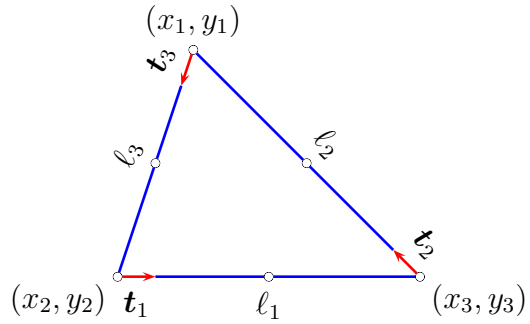


Figure 4.5: Parameters associated with element τ.

Proof. Since $\hat{u}_{p+1} - u_p\big|_\tau \in \mathcal{E}_{p+1}$ and $\{\psi_{p+1,i}\}_{i=1}^{p+1}$ is a basis of $\mathcal{E}_{p+1}$, we can write

$$\hat{u}_{p+1} - u_p\big|_\tau = \sum_{i=0}^{p+1} c_i \psi_{p+1,i} = \sum_{i=0}^{p+1} c_i \prod_{j=0}^{i-1} (\phi_1 - j/p) \prod_{m=0}^{p-i} (\phi_2 - m/p).$$

The coefficient c_i can be computed by taking the $(p+1)$th derivative $\partial_{\boldsymbol{t}_2}^i \partial_{\boldsymbol{t}_1}^{p+1-i}$ of both sides and using the following identity

$$\begin{pmatrix} \ell_1 \boldsymbol{t}_1^t \\ \ell_2 \boldsymbol{t}_2^t \\ \ell_3 \boldsymbol{t}_3^t \end{pmatrix} (\nabla\phi_1 \ \nabla\phi_2 \ \nabla\phi_3) = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}.$$

Then, equations (4.43) and (4.44) follow from equation (4.42), definition of directional derivative and the observation that:

$$\ell_2 \boldsymbol{t}_2 = \begin{pmatrix} x_1 - x_3 \\ y_1 - y_3 \end{pmatrix}; \quad \ell_1 \boldsymbol{t}_1 = \begin{pmatrix} x_3 - x_2 \\ y_3 - y_2 \end{pmatrix}.$$

□

In our *local error estimate* $\epsilon_\tau \approx \hat{u}_{p+1} - u_p\big|_\tau$, we use equations (4.43) and (4.44) and simply approximate the $(p+1)$th derivatives of $\hat{u}_{p+1}$ by

$$\partial_x^i \partial_y^{p+1-i} \hat{u}_{p+1} \approx \begin{cases} \partial_y R(\partial_y^p u_h), & i = 0, \\ (\partial_x R(\partial_x^{i-1} \partial_y^{p+1-i} u_h) + \partial_y R(\partial_x^i \partial_y^{p-i} u_h))/2, & 1 \le i \le p, \\ \partial_x R(\partial_x^p u_h), & i = p+1. \end{cases}$$

Similar to the linear and quadratic cases, we also define *scaling factor* α_τ so that the *local error indicator* η_τ satisfies

$$\eta_\tau^2 = \alpha_\tau^2 \sum_{i=0}^{p} \binom{p}{i} \|\partial_x^i \partial_y^{p-i} \epsilon_\tau\|_{0,\tau}^2 = \sum_{i=0}^{p} \binom{p}{i} \|(I-R)(\partial_x^i \partial_y^{p-i} u_h)\|_{0,\tau}^2 \tag{4.45}$$

Since ϵ_τ is a discontinuous piecewise polynomial (of degree $p+1$) on the whole Ω, we could also formally approximate errors in global norm and other functions using ϵ_τ. More precisely,

$$\|u - u_h\|_{0,\Omega}^2 \approx \sum_{\tau \in \Omega} \|\epsilon_\tau\|_{0,\tau}^2,$$
$$|u - u_h|_{1,\Omega}^2 \approx \sum_{\tau \in \Omega} |\epsilon_\tau|_{1,\tau}^2 = \sum_{\tau \in \Omega} \eta_\tau^2.$$

For $|\cdot|_{1,\Omega}$, we have estimates which are similar to (4.32) in the linear case.

$$|u - u_h|_{1,\Omega} \le |u - \hat{u}_{p+1}|_{1,\Omega} + |\hat{u}_{p+1} - u_p|_{1,\Omega} + |u_p - u_h|_{1,\Omega} \tag{4.46a}$$

$$|\hat{u}_{p+1} - u_p|_{1,\Omega} \le |u_p - u_h|_{1,\Omega} + |u - u_h|_{1,\Omega} + |\hat{u}_{p+1} - u|_{1,\Omega} \tag{4.46b}$$

Under standard conditions, we have $|u - u_h|_{1,\Omega} \ge ch^p$ and $|u - \hat{u}_{p+1}|_{1,\Omega} \le Ch^{p+1}$. Therefore, if $|u_p - u_h|_{1,\Omega}$ can be estimated better than $O(h^p)$ then estimates (4.46) show $|\hat{u}_{p+1} - u_p|_{1,\Omega}$ to be an asymptotically exact estimate for $|u - u_h|_{1,\Omega}$. Such super-approximation estimates for $|u_p - u_h|_{1,\Omega}$ are known for $p = 1, 2$ (see Theorem 4.6 and [43]). However super-approximation estimates for $|u_p - u_h|_{1,\Omega}$ are known not to hold for $p \ge 3$ (see [44]). For general p, estimate (4.46a) can be replaced by

$$|u - u_h|_{1,\Omega} \le C(|u - \hat{u}_{p+1}|_{1,\Omega} + |\hat{u}_{p+1} - u_p|_{1,\Omega})$$

Here we lose asymptotic exactness, but still have a useful upper bound for the H^1 error.

4.4 *hp*-Refinement Indicator

Numerical experiments show that alternating use of the h-version and p-version results in good meshes and shows exponential rate-of-convergence in many cases. However, the performance of the H^1 error estimate degenerates for problems with singularities. This is mainly due to the fact that p-refinement tends to use elements of high degree in the regions near singularities while using elements of smaller size (h-refinement) usually yields better accuracy in these regions. Because of this, we want to use h-refinement near singularities and where the solutions change rapidly, and to use p-refinement elsewhere. The question is how to identify these regions. Our answer come from the consistency-check constant α_τ in (4.45).

Normally, one should expect that $\alpha_\tau \approx 1$ in the region where the solution is smooth and that α_τ is big elsewhere. This suggests using the quantity α_τ to indicate regions with rapid changes or singularity in the solution. In the examples shown in Figure 4.6, α_τ seems to be a good indicator. These example comes from Circle problem whose solution has a singularity at the origin (see Section 6.3 for more details about Circle problem). And we can see that scaling factors of elements near the origin are much larger than other elements.

However, by experimenting different problems with different scenarios, we discover that the scaling factor can be large along interfaces separating elements of different degrees. An example of this behavior is illustrated in Figure 4.7.

Empirical study shows that computing α_t with the assumption that all elements are linear gives a very good indicator. In this approach, elements of high degree are still utilized for approximating the solution and estimating errors. The sole difference is that only vertex degrees of freedom are used in calculating the scaling factor α_τ.

The effectiveness of this indicator in identifying critical regions and guiding automatic hp-adaptive is demonstrated in Sections 6.2.1 and Section 6.2.2.

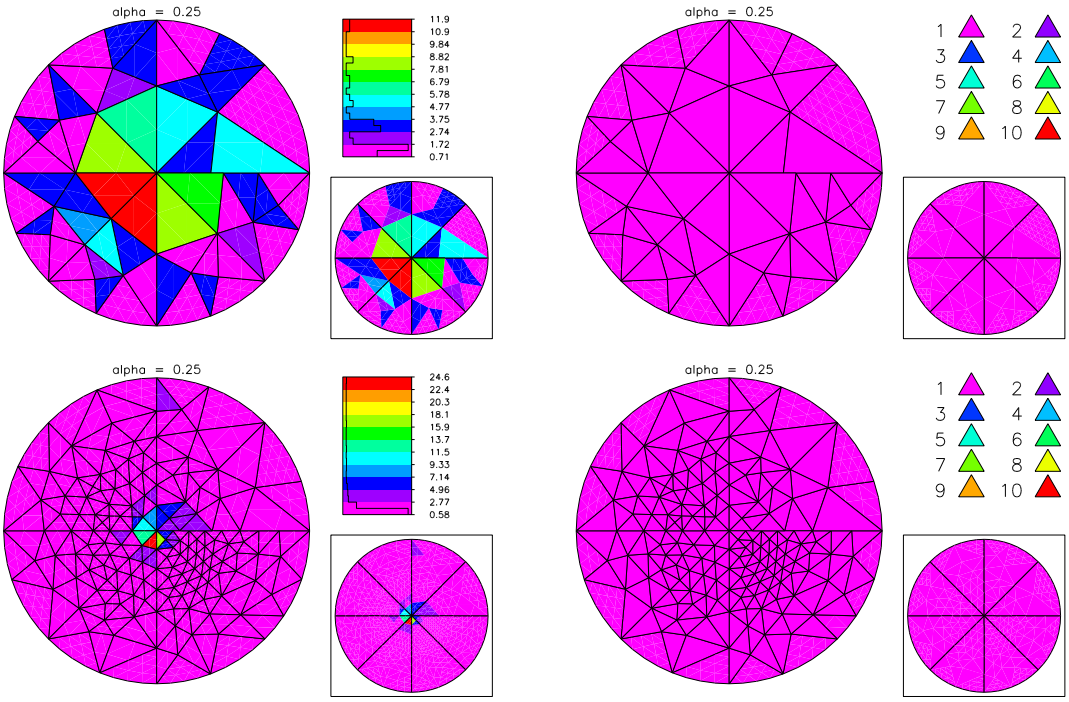


Figure 4.6: Scaling factors of elements (left) and the associated mesh with element degrees (right).

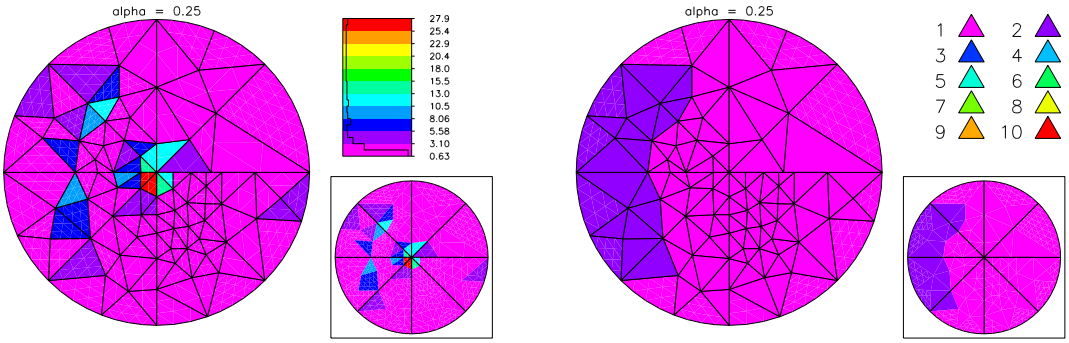


Figure 4.7: Scaling factors of elements (left) and the associated mesh with element degrees (right).

Chapter 5

Domain Decomposition and hp-Adaptive Meshing

5.1 Introduction

In this chapter, we discuss a combination of two techniques: hp-adaptive meshing and domain decomposing on a parallel machine. These two techniques are very effective and popular in scientific computing. However, attempts to combine them together to work on parallel machine usually result in an inefficient method. The main reason is that adaptive meshing proceeds gradually from a small problem (coarse mesh) to a bigger problem (fine mesh), while using a domain decomposition solver on a parallel machine can be effective only when the problem size is sufficiently large. In order to overcome this dilemma, Bank and Hoslt propose a parallel adaptive meshing paradigm in which each processor processes the whole domain but its adaptive enrichment focuses only on its own subregion (see [18, 19]). The bulk of the calculation, then, takes place on each processor. Communication is only needed at the beginning when information of the initial mesh is broadcast to all processors, and in the last step when solutions on subregions are glued together. The algorithm not only helps to keep communication cost low but also allows sequential adaptive finite element packages such as PLTMG to be employed without extensive recoding.

The rest of this chapter is organized as follows: Section 2 presents two different variants of the paradigm. In section 3, we briefly discuss load balancing and adaptive meshing steps and observe how *hp*-refinements affects the paradigm. Section 4 is devoted to a domain decomposition solver which is described in both variational form and matrix form.

5.2 Parallel Adaptive Meshing Paradigm

The original version of the paradigm has three main components:

Step I - Load Balancing: We solve a small problem on a coarse mesh, and use a posteriori error estimates to partition the mesh. Each subregion has approximately the same error although subregions may vary considerably in terms of number of elements, number of degrees of freedom, or polynomial degree.

Step II - Adaptive Meshing: Each processor is provided the complete coarse problem and instructed to sequentially solve the *entire* problem, with the stipulation that its adaptive enrichment (h or p) should be limited largely to its own subregion. The target number of degrees of freedom for each processor is the same. At the end of this step, the mesh is regularized such that the global finite element space described in Step III is conforming in both h and p.

Step III - Global Solve: The final global problem consists of the union of the refined partitions provided by each processor. A final solution is computed using domain decomposition.

There is a variant of the above approach, in which the load balancing occurs on a much finer space (see [15]). For this variant, the motivation was to address some possible problems arising from the use of a coarse grid in computing the load balance. For example, the not-so-accurate solution on a coarse mesh might lead to a bad load balancing. This variant also has three main components.

Step I - Load Balancing: On a single processor we adaptively create a *fine* space of size N_P, and use a posteriori error estimates to partition the mesh such that each subregion has approximately equal error, similar to Step I of the original paradigm.

Step II - Adaptive Meshing: Each processor is provided the complete adaptive mesh and instructed to sequentially solve the *entire* problem. However, in this case each processor should adaptively *coarsen* subregions corresponding to other processors, and adaptively enrich its own subregion. The size of the problem on each processor remains N_P, but this adaptive rezoning strategy concentrates the degrees of freedom in the processor's subregion. At the end of this step, the global space is made conforming as in the original paradigm.

Step III - Global Solve: This step is the same as in the original paradigm.

In the variant, the initial mesh can be of any size. Indeed, our choice of N_P is mainly for convenience and simplicity of notation; any combination of coarsening and refinement could be allowed in Step II.

5.3 Load Balancing and Adaptive Meshing

5.3.1 Load Balancing

One of the most challenging obstacles to overcome in making effective use of parallel computers for adaptive finite element codes is the load balancing problem. The traditional approach of distributing elements fairly among processors could quickly become imbalanced since adaptive meshing guides the mesh enrichment based on error estimates which mimic the solution's (usually nonuniform) behavior. In PLTMG, we use error estimates to do load balancing. In particular, the domain is partitioned in such a way that each subregion has approximately equal error.

Prior to load balancing, the PDE is solved on a single processor, usually the master one. Based on the approximate solution, the local errors estimate for each element are computed. Then we form macro-elements which are patches of elements of small errors. The error associated with each patch is the sum of local error estimates of elements inside the patch. These patches are distributed among processors based on an algorithm similar to the recursive spectral bisection algorithm described in [31]. In the algorithm, the bisection procedure is based on the solution of an eigenvalue problem associated with the adjacency matrix of the dual graph of the macro element mesh, in which the off-diagonal entries are

weighted by the number of overlapping edges in the original triangulation. While this algorithm is considered expensive, it is used only once on a relatively coarse mesh.

5.3.2 Adaptive Meshing

This is the only step in the paradigm that needs special treatment when *hp*-adaptive enrichment is used instead of the classical *h*-adaptive enrichment. This treatments is highly technical. Therefore, we restrict the discussion in this subsection to ideas rather than implementation details (even though the implementation was where we spend most of our efforts).

For the sake of clarity, we split this step into two phases: *adaptive enrichment* and *mesh regularization*.

In adaptive enrichment phase, the mesh on each processor is independently refined/unrefined in both h and p. Since there is no constraint on adaptive enrichment of different processors, we would like to use the same sequential p and *hp*-adaptive refinement/unrefinement strategies we developed in Chapter 3. In order to do that without any modifications in the implementation of these strategies, we weight local error estimates of each element by the "distance"[1] of its subregion to the subregion owned by the processor. With this weighting, error estimates are very likely to be big in the subregion owned by the processor and smaller elsewhere. Therefore, adaptive refinements on each processor appear mainly on its own subregion, and each processor focuses its adaptive unrefinements outside its own subregion.

In the regularization phase, meshes are made conforming in both geometry (in h) and degree (in p). This is necessary as the enrichment phase is carried independently on each processor. And it is possible that some elements along the interface have been refined/unrefined (in both h and p) on a processor, while their neighboring elements have stayed the same or have been refined/unrefined differently on another processor. Naturally, the meshes are made conforming in geometry first and then in degree.

[1]Here the "distance" is topological distance, not the physical one.

In making the meshes geometrically conforming, regiment level and label of the ancestor edge (the edge in the initial mesh which is broadcast) associated with each interface edge are recorded.[2] Before the initial broadcasting, edges on subregion interfaces of the mesh are labeled consecutively starting from zero. The refinement levels of these edges are also set to be zero. Whenever an interface edge is bisected (in PLTMG, we use longest edge bisection for h- refinement), its two children edges inherit the ancestor label, and increase the refinement level by one.

At the beginning of the regularization phase, data describing the interfaces are exchanged among processors. Using the information of refinement level and ancestor label, two copies in two different processors of the same edge can be matched. If an refined edge in a processor has no match then that part of interface (on the processor) is coarser than the corresponding part of interface in the neighboring processor. Less refined interface edges on a processor are h-refined to be compatible with the neighbor in the global mesh. Less refined interface edges on the neighbor's side are refined by the neighbor. After all processors refine their necessary edges, subregions contributed by them are globally compatible (see Figure 5.1)).

In PLTMG, we actually perform another mesh regularization on part of the interface that does not appear in the global mesh. This is to accelerate the domain decomposition solver described in the next section. In this step, local interface edges that are more refined than the global interface system will be coarsened.

When the meshes are geometrically conforming, we move to the next step and make them conforming in degree. In the early version of PLTMG, we tried to keep all the local meshes, and the global mesh admissible. That is, to make sure there is no violation of 1-irregular rule or 2-neighbor rule. This could be very complicated with the possibility of transition edges on the interface system. Therefore, first we use p-refinement to eliminate all of the transition edges existing on the interface of local meshes. Then data of edges on the interface are exchanged between processors. Since these meshes are already geometrically conforming, interfaces edges can be divided in to pairs, where each pair contains edges that

[2]In PLTMG, these two pieces of information are coded using a single number.

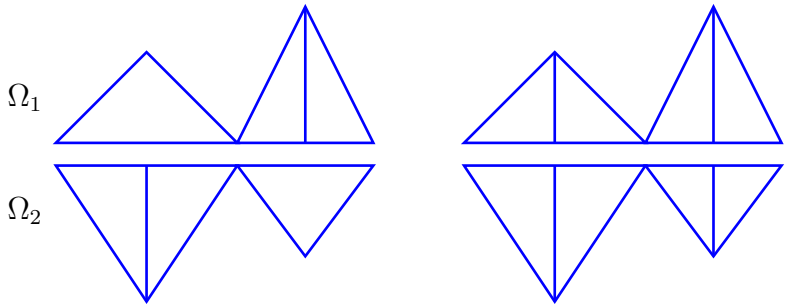


Figure 5.1: The coarse side of a non matching interface (left) is refined to make the global mesh conforming (right).

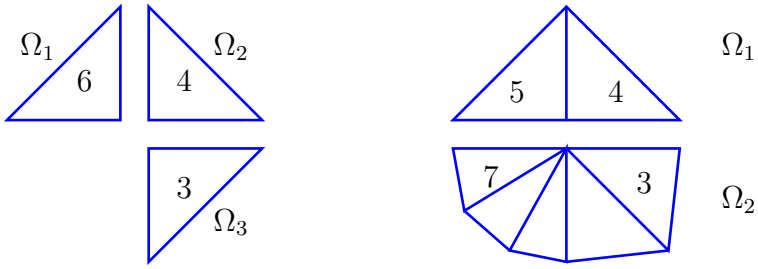


Figure 5.2: Examples require multiple communications.

are copies of a single edge (in the global mesh) in two different processors. If the difference in degree of edges in a pair is greater than one (violation of 1-irregular rule), then the one with lower degree is p-refined.

When 1-irregular rule and 2-neighbor rule are applied strictly, a p-refinement of an interface edge might cause change in degree of another interface edge. A typical case is to have elements with two interface edges that are associated with three different processors (see Figure 5.2). Another (extreme) example is also illustrated in Figure 5.2. In these cases, communication is needed more than once in order to make the meshes conforming in degree.

Taking into consideration the cost of implementing degree regularization code and needed communication, we decide to use the second approach in defining basis functions for transition elements (see Subsection 2.3.2). This approach allows us to make the meshes conforming in degree using only one communication. The edge of element with lower degree in a pair is converted to a transition edge of the higher degree. No subsequent refinements are done to eliminate rule violations. Even though there might be some rule violations in the global mesh, numerical experiments show that they do not affect the accuracy of the solution.

5.4 Domain Decomposition Solver

Let $\Omega = \cup_{i=1}^{P} \Omega_i \subset \mathbb{R}^2$ denote the domain, decomposed into P geometrically conforming subregions. At this step of the paradigm, each processor contain different meshes which are obtained from the same initial mesh after different sequences of adaptive refinement/unrefinement. For processor i, we will refer to the "fine grid" as the partition associated with Ω_i, and the "coarse grid" as the remainder of the mesh.

Let Γ denote the interface system. At this step, the meshes should be conforming in both h (geometry) and p (degree) along Γ. Let x be a degree of freedom lying on Γ then $degree(x)$ is the number of subregions for which $x \in \bar{\Omega}_i$. A cross point is a degree of freedom $x \in \Gamma$ with $degree(x) \geq 3$. Note that a cross point must be a vertex degree of freedom. We assume that the maximal degree at cross points is bounded by the constant δ_0. The connectivity of Ω_i is the number of other regions Ω_j for which $\bar{\Omega}_i \cap \bar{\Omega}_j \neq \emptyset$. We assume that the connectivity of Ω_i is bounded by the constant δ_1.

In our algorithm, we employ several triangulations. The mesh $\mathcal{T}$ is a globally refined, shape regular, and conforming mesh of size h. We assume that the fine mesh $\mathcal{T}$ is aligned with the interface system Γ. The triangulations $\mathcal{T}^i \subset \mathcal{T}$, $1 \leq i \leq P$ are partially refined triangulations; they coincide with the fine triangulation $\mathcal{T}$ within Ω_i, but are generally much coarser elsewhere, although as in the case for the variant paradigm, along the interface system Γ, $\mathcal{T}^i$ may have some

intermediate level of refinement.

Let $\mathcal{S}$ denote the *hp* space of piecewise polynomials, associated with the triangulation $\mathcal{T}$, that are continuous in each of the Ω_i, but can be discontinuous along the interface system Γ. Let $\bar{\mathcal{S}} \subset \mathcal{S}$ denote the subspace of globally continuous piecewise polynomials. The usual basis for $\mathcal{S}$ is just the union of the nodal basis functions corresponding to each of the subregions Ω_i; such basis functions have their support in $\bar{\Omega}_i$ and those associated with nodes on Γ will have a jump at the interface. In our discussion, we will have occasion to consider another basis, allowing us to write $\mathcal{S} = \bar{\mathcal{S}} \oplus \mathcal{X}$, where $\mathcal{X}$ is a subspace associated exclusively with jumps on Γ. In particular, we will use the global conforming nodal basis for the space $\bar{\mathcal{S}}$, and construct a basis for $\mathcal{X}$ as follows. Let z_k be a degree of freedom lying on Γ shared by two regions Ω_i and Ω_j (for now, z_k is not a cross point). Let $\phi_{i,k}$ and $\phi_{j,k}$ denote the usual nodal basis functions corresponding to z_k in Ω_i and Ω_j, respectively. The continuous nodal basis function for z_k in $\bar{\mathcal{S}}$ is $\phi_k \equiv \phi_{i,k} + \phi_{j,k}$, and the "jump" basis function in $\mathcal{X}$ is $\hat{\phi}_k \equiv \phi_{i,k} - \phi_{j,k}$. The direction of the jump is arbitrary at each z_k, but once chosen, will be used consistently. In this example, at degree of freedom z_k we will refer to i as the "master" index and j as the "slave" index. At a cross point where $\ell > 2$ subregions meet, there will be one nodal basis function corresponding to $\bar{\mathcal{S}}$ and $\ell - 1$ jump basis functions. These are constructed by choosing one master index for the point, and making the other $\ell - 1$ indices slaves. We can construct $\ell - 1$ basis functions for $\mathcal{X}$ as $\phi_{i,k} - \phi_{j,k}$, where i is the master index and j is one of the slave indices.

For each of the triangulations $\mathcal{T}^i$, $1 \leq i \leq P$ we have a global nonconforming subspace $\mathcal{S}^i \subset \mathcal{S}$, and global conforming subspace $\bar{\mathcal{S}}^i \subset \bar{\mathcal{S}}$. In a fashion similar to $\mathcal{S}$, we have $\mathcal{S}^i = \bar{\mathcal{S}}^i \oplus \mathcal{X}^i$.

5.4.1 Variational Form

For simplicity, let the continuous variational problem be: find $u \in \mathcal{H}^1(\Omega)$ such that

$$a(u, v) = (f, v) \tag{5.1}$$

for all $v \in \mathcal{H}^1(\Omega)$, where $a(u,v)$ is a self-adjoint, positive definite bilinear form corresponding to the weak form of an elliptic partial differential equation, and $|\!|\!| u |\!|\!|_\Omega^2 = a(u,u)$ is comparable to the usual $\mathcal{H}^1(\Omega)$ norm.

To deal with the nonconforming nature of $\mathcal{S}$, for $u, v \in \mathcal{S}$, we decompose $a(u,v) = \sum_{i=1}^P a_{\Omega_i}(u,v)$. For each node z lying on Γ there is one master index and $\ell - 1 > 0$ slave indices. The total number of slave indices is denoted by K, so the total number of constraint equations in our nonconforming method is K. To simplify notation, for each $1 \le j \le K$, let $m(j)$ denote the corresponding master index, and z_j the corresponding node. We define the bilinear form $b(v,\lambda)$ by

$$b(v,\lambda) = \sum_{j=1}^{K} \{v_{m(j)} - v_j\}\lambda_j \tag{5.2}$$

where $\lambda \in \mathbb{R}^K$. In words, $b(\cdot,\cdot)$ measures the jump between the master value and each of the slave values at each node on Γ. The nonconforming variational formulation of (5.1) is: find $u_h \in \mathcal{S}$ such that

$$\begin{aligned} a(u_h, v) + b(v,\lambda) &= (f,v) \\ b(u_h,\xi) \qquad\quad &= 0 \end{aligned} \tag{5.3}$$

for all $v \in \mathcal{S}$ and $\xi \in \mathbb{R}^K$. Although this is formally a saddle point problem, the constraints are very simple; in particular, (5.3) simply imposes continuity at each of the nodes lying on Γ, which in turn, implies that $u_h \in \bar{\mathcal{S}}$. Thus u_h also solves the reduced and conforming variational problem: find $u_h \in \bar{\mathcal{S}}$ such that

$$a(u_h, v) = (f,v)$$

for all $v \in \bar{\mathcal{S}}$.

Let $\mathcal{K}_i$ denote the index set of constraint equations in (5.2) that correspond to nodes present in $\mathcal{T}^i$. Then

$$b_i(v,\lambda) = \sum_{j \in \mathcal{K}_i} \{v_{m(j)} - v_j\}\lambda_j.$$

We are now in a position to formulate our domain decomposition algorithm. Our initial guess $u_0 \in \mathcal{S}$ is generated as follows: for $1 \le i \le P$, we find (in parallel) $u_{0,i} \in \bar{\mathcal{S}}^i$ satisfying

$$a(u_{0,i}, v) = (f,v) \tag{5.4}$$

for all $v \in \bar{\mathcal{S}}^i$. Here we assume exact solution of these local problems; in practice, these are often solved approximately using iteration. The initial guess $u_0 \in \mathcal{S}$ is composed by taking the part of $u_{0,i}$ corresponding to the fine subregion Ω_i for each i. In particular, let χ_i be the characteristic function for the subregion Ω_i. Then

$$u_0 = \sum_{i=1}^{P} \chi_i u_{0,i}$$

To compute $u_{k+1} \in \mathcal{S}$ from $u_k \in \mathcal{S}$, we solve (in parallel): for $1 \leq i \leq P$, find $e_{k,i} \in \mathcal{S}^i$ and $\lambda_{k,i} \in \mathbb{R}^K$ such that

$$\begin{aligned} a(e_{k,i}, v) + b_i(v, \lambda_{k,i}) &= (f, v) - a(u_k, v) \\ b_i(e_{k,i}, \xi) \qquad\qquad &= -b_i(u_k, \xi) \end{aligned} \tag{5.5}$$

for all $v \in \mathcal{S}^i$ and $\xi \in \mathbb{R}^K$. We then form

$$u_{k+1} = u_k + \sum_{i=1}^{P} \chi_i e_{k,i}.$$

Although the iterates u_k are elements of the nonconforming space $\mathcal{S}$, the limit function $u_\infty = u_h \in \bar{\mathcal{S}}$. In some sense, the purpose of the iteration is to drive the jumps in the approximate solution u_k to zero. Also, although (5.5) suggests a saddle point problem needs to be solved, by recognizing that only $\chi_i e_{k,i}$ is actually used, one can reduce (5.5) to a positive definite problem of the form (5.4). In particular, the Lagrange multipliers $\lambda_{k,i}$ need not be computed or updated.

The information required to be communicated among the processors is only the solution values and the residuals for nodes lying on Γ, which is necessary to compute the right hand sides of (5.5). This requires one all-to-all communication step at the beginning of each DD iteration.

5.4.2 Matrix Form

With an appropriate finite element basis, equation (5.1) can be written in the matrix form as

$$AU = F \tag{5.6}$$

where A is the stiffness matrix, U is the coefficient vector of the finite element solution and F is the right hand side, all with respect to the same basis. Usually (5.6) is solved by iterative methods using the following update scheme

$$R = F - AU \tag{5.7a}$$

$$A\,\delta U = R \tag{5.7b}$$

$$U = U + \delta U \tag{5.7c}$$

In our case, (5.7b) is more complicated since we need to impose the continuity along the boundary.

For the sake of simplicity, we first consider the case of two subregions only. With proper ordering of unknowns, the global system of equations for the update δU has the block 5×5 form

$$\begin{pmatrix} A_{11} & A_{1\gamma} & 0 & 0 & 0 \\ A_{1\gamma} & A_{\gamma\gamma} & 0 & 0 & I \\ 0 & 0 & A_{\nu\nu} & A_{\nu 2} & -I \\ 0 & 0 & A_{2\nu} & A_{22} & 0 \\ 0 & I & -I & 0 & 0 \end{pmatrix} \begin{pmatrix} \delta U_1 \\ \delta U_\gamma \\ \delta U_\nu \\ \delta U_2 \\ \Lambda \end{pmatrix} = \begin{pmatrix} R_1 \\ R_\gamma \\ R_\nu \\ R_2 \\ U_\nu - U_\gamma \end{pmatrix} \tag{5.8}$$

In (5.8), indices 1, 2 are used for quantities associated with fine grid degree of freedom of processor 1 and 2 while indices γ, ν are used for quantities associated with interface of processor 1 and 2 respectively. The reason there is no block crossed between two processors is that each physical node on the interface is associated with a pair of different degrees of freedom: one for processor 1 and one for processor 2. Eventually, we would want the approximate solution U to have the same values at interface degrees of freedom coming from a single pair. The fifth block equation guarantees that by imposing the constraint

$$U_\gamma + \delta U_\gamma = U_\nu + \delta U_\nu.$$

Since setting up an equation like (5.8) on a single processor would require expensive communication and a lot of memory, we set up a similar but "local" formulation in which fine grid and coarse grid of the same processor are "mortared"

to each other. On processor 1, we have

$$\begin{pmatrix} A_{11} & A_{1\gamma} & 0 & 0 & 0 \\ A_{1\gamma} & A_{\gamma\gamma} & 0 & 0 & I \\ 0 & 0 & \bar{A}_{\nu\nu} & \bar{A}_{\nu 2} & -I \\ 0 & 0 & \bar{A}_{2\nu} & \bar{A}_{22} & 0 \\ 0 & I & -I & 0 & 0 \end{pmatrix} \begin{pmatrix} \delta U_1 \\ \delta U_\gamma \\ \delta \bar{U}_\nu \\ \delta \bar{U}_2 \\ \Lambda \end{pmatrix} = \begin{pmatrix} R_1 \\ R_\gamma \\ R_\nu \\ 0 \\ U_\nu - U_\gamma \end{pmatrix} \tag{5.9}$$

where quantities with a bar (e.g. $\bar{A}_{\nu\nu}$) refer to the coarse grid. A similar formulation can be set up for processor 2. Here we note that the residual for the coarse grid is set to be zero. This is a very important assumption as it helps avoiding the need to obtain R_2 via communication and to implement a calculation to restrict R_2 to the coarse grid on processor 1 (in general, the coarse grid on processor 1 is much coarser than the fine grid on processor 2). Besides, R_1 and R_2 are anticipated to be close to zero, especially after few steps of the iterative solve.

Since on processor 1 we only need δU_1 and δU_γ to update the part of global solution belonging to processor 1, we formally reorder (5.9) as

$$\begin{pmatrix} 0 & -I & 0 & I & 0 \\ -I & \bar{A}_{\nu\nu} & 0 & 0 & \bar{A}_{\nu 2} \\ 0 & 0 & A_{11} & A_{1\gamma} & 0 \\ I & 0 & A_{\gamma 1} & A_{\gamma\gamma} & 0 \\ 0 & \bar{A}_{2,\nu} & 0 & 0 & \bar{A}_{22} \end{pmatrix} \begin{pmatrix} \Lambda \\ \delta \bar{U}_\nu \\ \delta \bar{U}_1 \\ \delta \bar{U}_\gamma \\ \delta \bar{U}_2 \end{pmatrix} = \begin{pmatrix} U_\nu - U_\gamma \\ R_\nu \\ R_1 \\ R_\gamma \\ 0 \end{pmatrix}. \tag{5.10}$$

Then we block eliminate the Lagrange multiplier Λ and $\delta \bar{U}_\nu$ to have

$$\begin{pmatrix} A_{11} & A_{1\gamma} & 0 \\ A_{\gamma 1} & A_{\gamma\gamma} + \bar{A}_{\nu\nu} & \bar{A}_{\nu 2} \\ 0 & \bar{A}_{2\nu} & \bar{A}_{22} \end{pmatrix} \begin{pmatrix} \delta U_1 \\ \delta U_\gamma \\ \delta U_2 \end{pmatrix} = \begin{pmatrix} R_1 \\ R_\gamma + R_\nu + \bar{A}_{\nu\nu}(U_\nu - U_\gamma) \\ \bar{A}_{2\nu}(U_\nu - U_\gamma). \end{pmatrix} \tag{5.11}$$

Clearly, the matrix in the left hand side of (5.11) can be locally assembled using regular finite elements discretization on processor 1. On the other hand, in order to compute the right hand side of (5.11) we need data from the interface of processor 2, namely, the residual R_ν and the part of the solution on the interface U_ν. In summary, a single domain decomposition iteration on processor 1 consists of the following steps

1. locally set up the stiffness matrix on the left hand side of (5.11).
2. locally compute R_1 and R_γ.
3. exchange boundary data (send R_γ and U_γ; receive R_ν and U_ν).
4. compute the right hand side of (5.11).
5. locally solve equation (5.11).
6. update U_1 and U_γ using δU_1 and δU_γ.

Here the equation (5.11) can be solved using any sequential solver.

When we have more than two subregions, the fundamental elements of our algorithm remain the same, but description in matrix notation becomes much more complicated.

For an interface degree of freedom z_i with $degree(z_i) = l \geq 2$, we have $l-1$ constraints that equate l different solution values associated with the degree of freedom, e.g., $U_{i1} = U_{i2} = \cdots = U_{il}$. As in the variational form, we choose an index i_m of z_i as the master one and making $l-1$ other indices slaves. Then U_{i_m} is the master value, and the remaining slave values are given by $U_{i_s} = U_{i_m}, 1 \leq s \leq l, s \neq l$.

Now we consider the saddle point formulation for a single processor k, $1 \leq k \leq P$.

$$\begin{pmatrix} \bar{A}_{ss} & \bar{A}_{sm} & \bar{A}_{si} & I \\ \bar{A}_{ms} & \bar{A}_{mm} & \bar{A}_{mi} & -\bar{Z}^t \\ \bar{A}_{is} & \bar{A}_{im} & \bar{A}_{ii} & 0 \\ I & -\bar{Z} & 0 & 0 \end{pmatrix} \begin{pmatrix} \delta\bar{U}_s \\ \delta\bar{U}_m \\ \delta\bar{U}_i \\ \Lambda \end{pmatrix} = \begin{pmatrix} \bar{R}_s \\ \bar{R}_m \\ \bar{R}_i \\ \bar{Z}\bar{U}_m - \bar{U}_s \end{pmatrix}$$

To avoid unnecessary complexity in notation, we do not distinguish subregion k and other subregions. Here we use index i for matrix blocks associated with interior degrees of freedom for all subregions, and indices m and s for matrix blocks associated with master interface degrees of freedom and slave interface degrees of freedom, respectively. Since the adaptive refinement on processor k is focused on subregion k, the portion of matrix $\bar{A}_{ii}$ arising from subregion k is substantially

larger than from other subregions. In addition, as there might be more than one slave degree of freedom associated with a single master degree of freedom, the matrix $\bar{Z}$ is generally not an identity matrix; however, each row of $\bar{Z}$ will be zero except for a single entry associated with the master index.

Similar the case of two subregions, we block eliminate $\delta\bar{U}_s$ and the Lagrange multiplier Λ to get

$$\begin{pmatrix} \bar{A}_{mm} + \bar{A}_{ms}\bar{Z} + \bar{Z}^t\bar{A}_{sm} + \bar{Z}^t\bar{A}_{ss}\bar{Z} + \bar{Z}^t\bar{A}_{ss}\bar{Z} & \bar{A}_{mi} + \bar{Z}^t\bar{A}_{si} \\ \bar{A}_{im} + \bar{A}_{is}\bar{Z} & \bar{A}_{ii} \end{pmatrix} \begin{pmatrix} \delta\bar{U}_m \\ \delta\bar{U}_i \end{pmatrix} = \begin{pmatrix} \bar{R}_m + \bar{Z}^t\bar{R}_s - (\bar{A}_{ms} + \bar{Z}^t\bar{A}_{ss})(\bar{Z}\bar{U}_m - \bar{U}_s) \\ \bar{R}_i - \bar{A}_{is}(\bar{Z}\bar{U}_m - \bar{U}_s) \end{pmatrix} \quad (5.12)$$

The matrix on the left hand side of (5.12) can be assembled locally using regular finite element discretization on processor k. Actually, it is used in the final adaptive refinement step on processor k, with slight modification due to global mesh regularization. To compute the right hand side of (5.12), we need information for parts of $\bar{U}_m$, $\bar{U}_s$, $\bar{R}_m$ and $\bar{R}_s$ which are not associated with subregion k. In summary, the following is the algorithm for a single domain decomposition iteration on processor k.

1. locally set up the stiffness matrix on the left hand side of (5.12).
2. locally compute $\bar{R}_i$ and part of $\bar{R}_m$ associated with subregion k.
3. exchange boundary data (send parts of $\bar{R}_m$ and $\bar{U}_m$ associated with subregion k; receive $\bar{R}_s$, $\bar{U}_s$ and parts of $\bar{R}_m$, $\bar{U}_m$).
4. compute right hand side of (5.12).
5. locally solve equation (5.12).
6. update $\bar{U}_i$ and $\bar{U}_m$ using appropriate parts of $\delta\bar{U}_i$ and $\delta\bar{U}_m$.

Chapter 6

Numerical Experiments

In this chapter, theories studied in the previous chapters are implemented in PLTMG and tested via numerical experiments.

The first three sections of this chapter study the performance of p-adaptive and automatic hp-adaptive in terms of error. In these sections, model problems can have known solutions. Thus exact errors can be computed as the difference between the finite element solutions and the exact solutions. On the other hand, computed errors are predicted (without using the exact solutions) by our error estimators. The reliability of the error estimators are verified by studying the ratio between the two kinds of errors. In addition, the errors are tested against the exponential model discussed in Chapter 4:

$$\|e\|_{H^1(\Omega)} \leq C \exp(-bN^{1/k}), \tag{6.1}$$

Here e is the finite element error; N is the number of degree of freedom (DOF); C and b are independent of N, but dependent on the mesh and the solution; and $k = 3$ for solutions with singularities, $k = 2$ for smooth solutions.

The last section is devoted to solving a problem using the combination of hp-adaptive and domain decomposition. Since the explicit formula for the solution does not exist, we will study only the local and global behaviors of the hp-adaptive meshing and the convergence of the domain decomposition method.

6.1 Problem UCSD Logo

We consider the following problem

$$-\Delta u = f \qquad \text{in } \Omega \tag{6.2a}$$

$$u = g \qquad \text{on } \partial_D\Omega \tag{6.2b}$$

where Ω is the UCSD logo domains shown in Figure 6.1 (left). In addition, f and g are chosen such that

$$u(x, y) = e^{x^5+y^2}$$

is the solution of 6.2. The approximate shape of u is illustrated in Figure 6.1 (right).

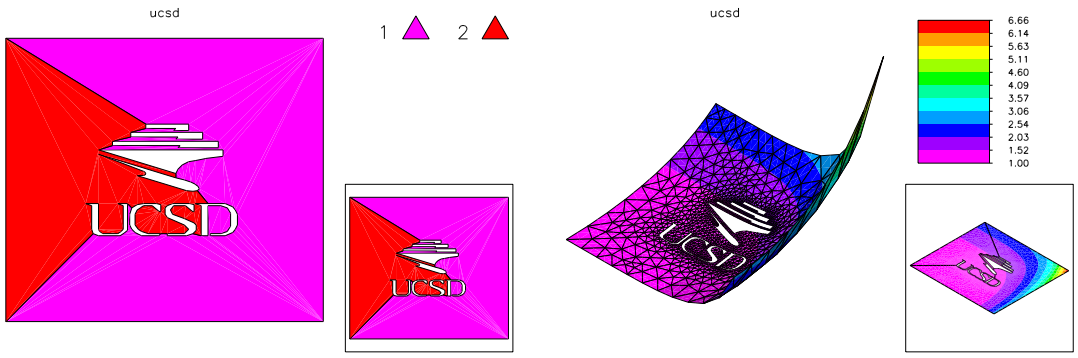


Figure 6.1: The domain (left) and the solution (right) - UCSD logo.

The problem is solved using different adaptive strategies, namely, h-adaptive, alternating h-adaptive and p-adaptive, and automatic hp-adaptive. When the two versions are used alternately, PLTMG is manually instructed which version to use.

As shown in Table 6.1, Table 6.2 and Table 6.3, both ways of combining the two versions of adaptive FEMs produce much better accuracy than the traditional h-version. In this particular problem, manually alternating the two versions of adaptive meshing is even better than automatic hp-adaptive. The reason is that the solution is very smooth and elements of high degree have advantage of capturing more terms in its Taylor expansion. And in the solve which alternates the two versions, we actually use more elements of high degree and fewer elements of small size.

Table 6.1: H^1 seminorm of errors in h-adaptive refinements - UCSD logo.

N	NTF	Exact $\|e\|_{1,\Omega}$	Computed $\|e\|_{1,\Omega}$	Ratio
1381	2174	0.1075307E+01	0.1525668E+01	1.42
5524	10260	0.1030214E+00	0.1247256E+00	1.21
22096	42931	0.4283418E−01	0.4363938E−01	1.02
88385	174601	0.2145246E−01	0.2156134E−01	1.01
353540	702887	0.1085278E−01	0.1086548E−01	1.00
500000	994909	0.8532813E−02	0.8546480E−02	1.00

Table 6.2: H^1 seminorm of errors in alternating hp-adaptive refinement UCSD logo.

N	NTF	Exact $\|e\|_{1,\Omega}$	Computed $\|e\|_{1,\Omega}$	Ratio
1381	2174	0.1075307E+01	0.1525668E+01	1.42
5531	7122	0.7673431E−01	0.9806008E−01	1.28
17483	7122	0.4766941E−02	0.4327741E−02	0.91
34481	7122	0.4186920E−03	0.3479004E−03	0.83
57743	11853	0.3204877E−05	0.4405957E−05	1.37
96180	20914	0.1448857E−05	0.1790322E−05	1.24
162266	20914	0.1036328E−07	0.1702824E−07	1.64

Table 6.3: H^1 seminorm of errors in automatic hp-adaptive refinements UCSD logo.

N	NTF	Exact $\|e\|_{1,\Omega}$	Computed $\|e\|_{1,\Omega}$	Ratio
1381	2174	0.1075307E+01	0.1525668E+01	1.42
5531	7122	0.7673431E−01	0.9806008E−01	1.28
18782	12401	0.5667967E−02	0.9061338E−02	1.60
44819	29822	0.6554077E−03	0.1032255E−02	1.57
107736	60869	0.8044036E−04	0.1486564E−03	1.85
239806	136981	0.8442172E−05	0.1463038E−04	1.73
497232	286457	0.1129567E−05	0.2118481E−05	1.88

Figure 6.2 shows the fitting curves of the H^1 seminorm of errors in alternating hp-adaptive and automatic hp-adaptive. The errors in these two cases does not fit the model (6.1) with $k = 2$ perfectly; and there are small oscillations in both cases. However, they clearly demonstrate exponential rate of convergence and are much smaller than the errors in the traditional h-adaptive. Moreover, in alternating hp-adaptive, errors seems to decrease a bit faster with p-refinement and slower with h-refinement (p-refinement can be recognized from the table when the number of element, NTF, is unchanged).

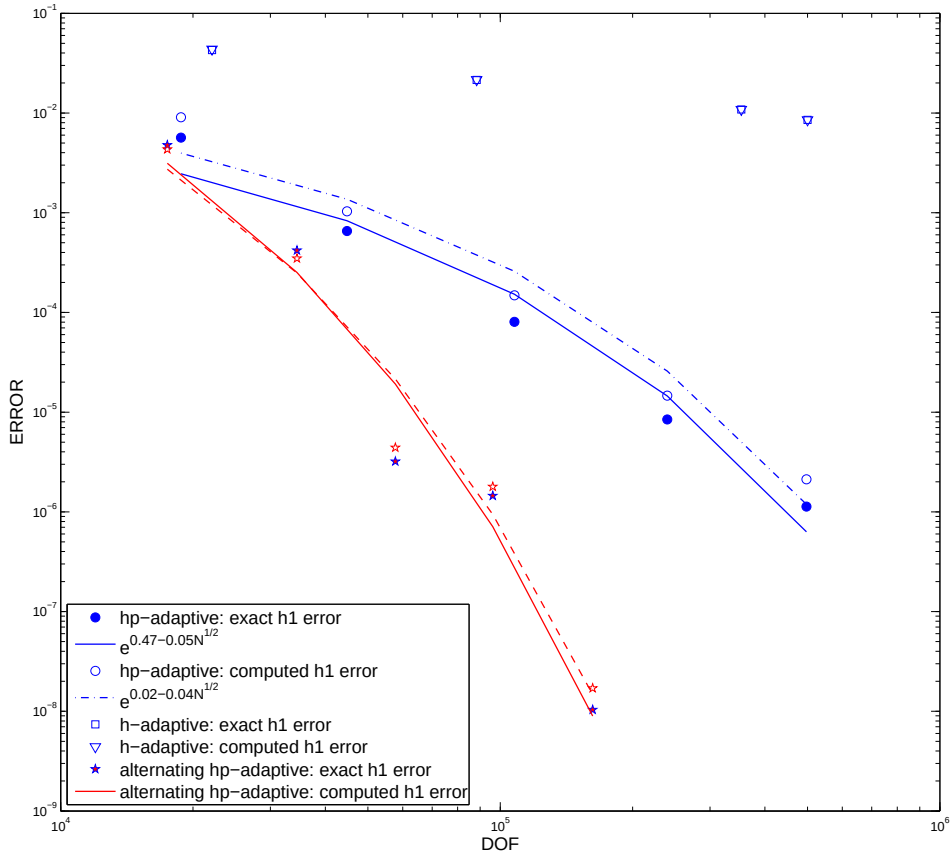


Figure 6.2: Loglog plot of errors and fitting curves - UCSD logo.

We note that the cases in which manually alternating hp-adaptive is better than automatic hp-adaptive are not popular in practice and that the performance

of this approach highly depend on the decision of the users on when to use which version. On the other hand, in automatic *hp*-adaptive the decision whether to refine an element in h or in p is made heuristically. The automatic *hp*-adaptive is even more attractive in the next three experiments where the exact solutions have singularities.

6.2 Problem with Singularities

Consider the following problem

$$-\Delta u + u = f \qquad \text{in } \Omega \tag{6.3a}$$

$$u = g \qquad \text{on } \partial_D\Omega \tag{6.3b}$$

$$\frac{\partial u}{\partial n} = h \qquad \text{on } \partial_N\Omega \tag{6.3c}$$

where Ω is the unit square in the first quadrant of the plane; $\partial_D\Omega$ and $\partial_N\Omega$ can be chosen as any combination of edges of the square as long as $\partial_D\Omega \cap \partial_N\Omega = \emptyset$ and $\partial_D\Omega \cup \partial_N\Omega = \partial\Omega$.

6.2.1 Problem with One Singularity

In this subsection, f and g are chosen so that the exact solution of (6.3) is given by

$$u(x, y) = \left((x - \frac{1}{3})^2 + (y - \frac{2}{3})^2\right)^{\frac{1}{2}} \tag{6.4}$$

We note that $u(x, y)$ has a singularity at $(1/3, 2/3)$ as its derivatives are unbounded there. The shapes of $u(x, y)$ viewed from different angles are illustrated in Figure 6.3.

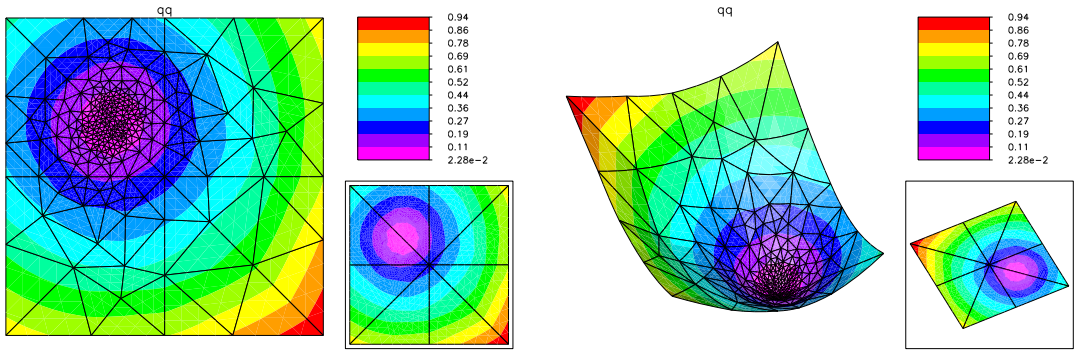


Figure 6.3: The solution viewed from different angles - One singularity.

As shown in Figure 6.4, automatic *hp*-adaptive is able to recognize the region of the singularity and automatically uses small elements of low degrees in that region while uses bigger elements of higher degree elsewhere.

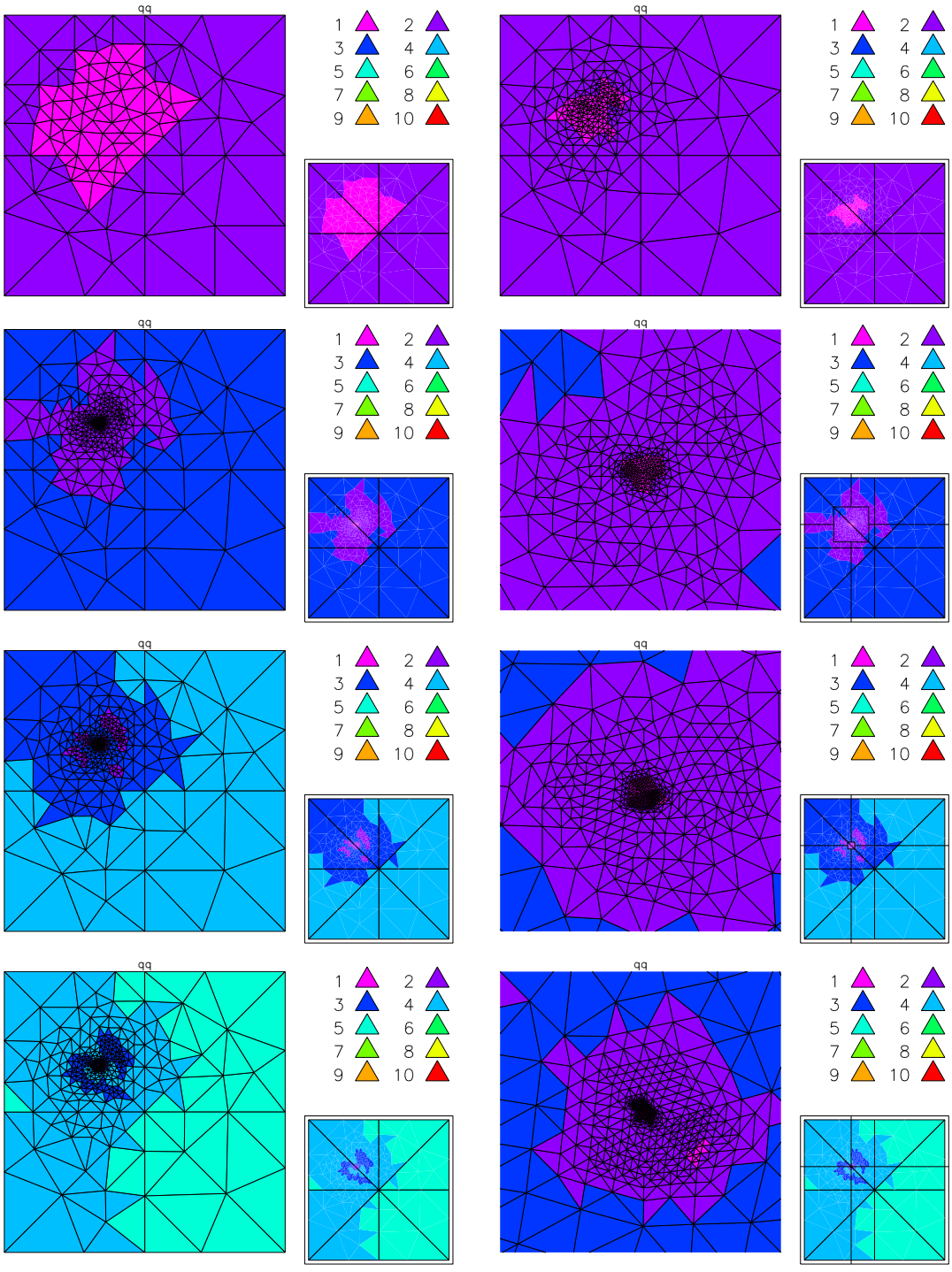


Figure 6.4: Meshes at different states in automatic *hp*-refinements (The last three subfigures on the right are closeups near the singularity of the ones on the right) One singularity.

Table 6.4: H^1 seminorm of errors in automatic *hp*-adaptive refinements
One singularity case.

N	NTF	Exact $\|e\|_{1,\Omega}$	Computed $\|e\|_{1,\Omega}$	Ratio
9	8	0.4881191E+00	0.8271845E+00	1.69
25	32	0.2918504E+00	0.4352211E+00	1.49
103	64	0.1505220E+00	0.3577442E+00	2.38
250	192	0.6215575E−01	0.1265815E+00	2.04
666	488	0.1986211E−01	0.3544746E−01	1.78
1661	856	0.3904102E−02	0.6816039E−02	1.75
3424	1374	0.8140260E−03	0.1437018E−02	1.77
6626	1926	0.1384592E−03	0.2819653E−03	2.04
10097	2378	0.2393775E−04	0.4399026E−04	1.84

Table 6.5: H^1 seminorm of errors in *h*-adaptive refinements - One singularity case.

N	NTF	Exact $\|e\|_{1,\Omega}$	Computed $\|e\|_{1,\Omega}$	Ratio
9	8	0.4881191E+00	0.8271845E+00	1.69
25	32	0.2918504E+00	0.4352211E+00	1.49
100	172	0.1481403E+00	0.2065763E+00	1.39
400	750	0.6383758E−01	0.9344887E−01	1.46
1600	3093	0.2756831E−01	0.3285567E−01	1.19
6401	12565	0.1342222E−01	0.1405042E−01	1.05
25604	50734	0.6693362E−02	0.6860291E−02	1.02
40000	79400	0.5020024E−02	0.5126507E−02	1.02

Not only can automatic *hp*-adaptive detect the singularity, but it also gives much better performance in terms of accuracy. Automatic *hp*-adaptive achieves the exact error of 0.2393775E-04 with a mesh of just 10097 degrees of freedom whereas *h*-adaptive can only achieve the exact error of 0.5020024E-02 with a mesh of 40000 degrees of freedom. Details can be found in Table 6.4 and Table 6.5.

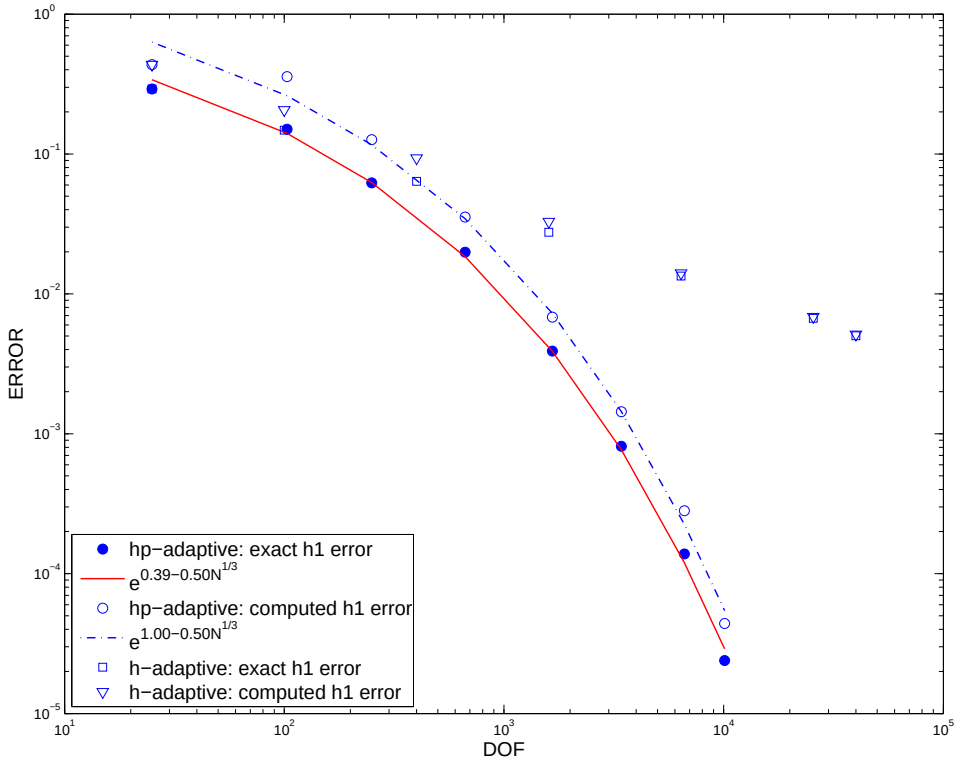


Figure 6.5: Loglog plot of errors with fitting curves - One singularity.

More importantly, Figure 6.5 shows that the rates of convergence of exact and computed errors of automatic *hp*-adaptive are exponential and optimal. They fit the exponential model (6.1) perfectly.

6.2.2 Problem with Two Singularities

In this subsection, f and g are chosen so that the exact solution of (6.3) is given by

$$u(x,y) = \Big(\big((x-a)^2+(y-b)^2\big)\big((x-c)^2+(y-d)^2\big)\Big)^{\frac{1}{2}} \tag{6.5}$$

Clearly, $u(x,y)$ have two singularities at (a,b) and (c,d) as its derivatives are unbounded at these two points. In our experiments, we choose $(a,b) = (1/3, 2/3)$ and $(c,d) = (2/3, 1/3)$. For these values of $a,\ b,\ c,\ d$, the shapes of $u(x,y)$ viewed from different angles are illustrated in Figure 6.6

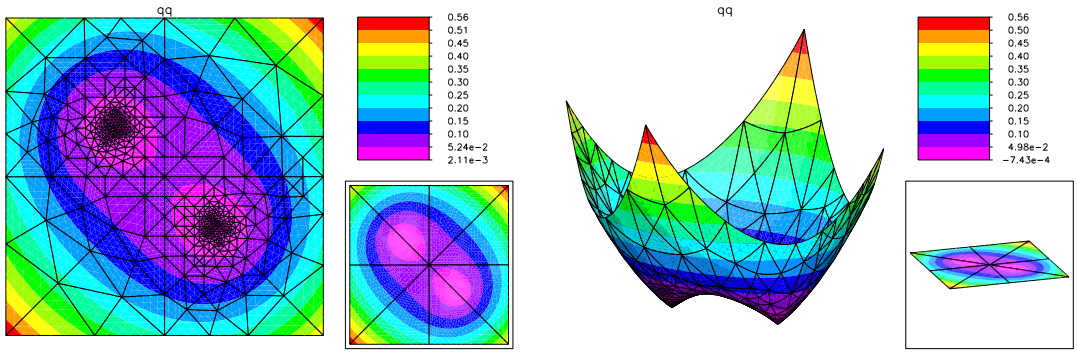


Figure 6.6: The solution viewed from different angles - Two singularities .

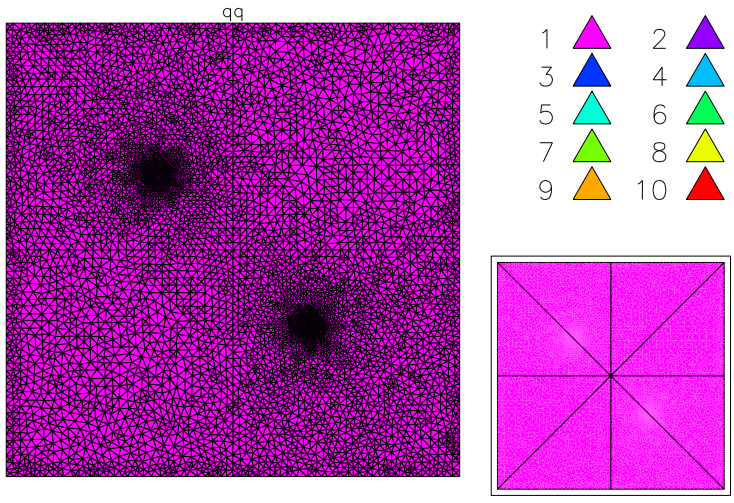


Figure 6.7: An adaptive mesh in h-refinements - Two singularities.

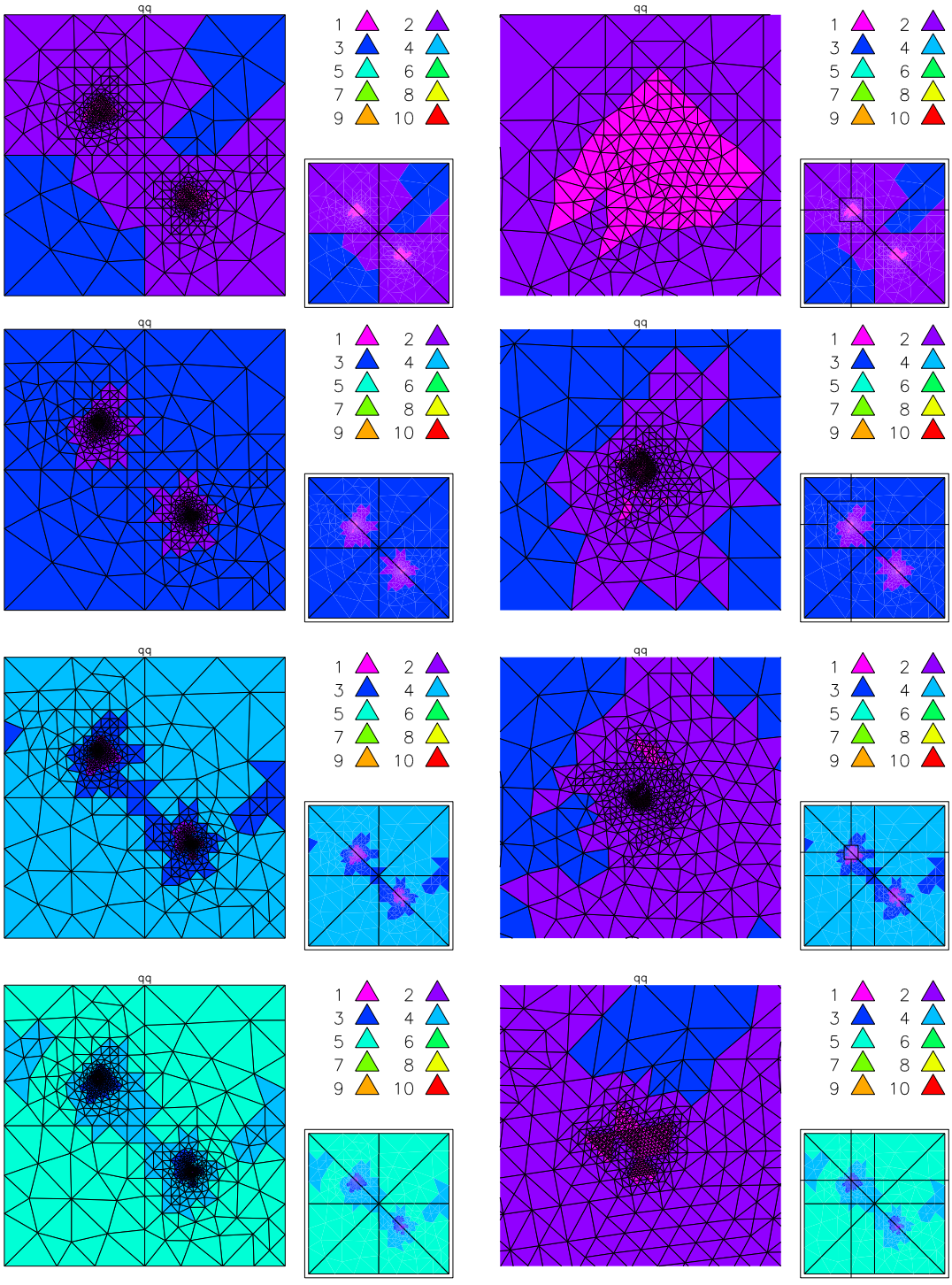


Figure 6.8: Meshes at different states in automatic *hp*-refinements (The subfigures on the right are closeups near the singularity of the ones on the right) Two singularities.

Table 6.6: H^1 seminorm of errors in automatic *hp*-adaptive refinements Two singularities.

N	NTF	Exact $\|e\|_{1,\Omega}$	Computed $\|e\|_{1,\Omega}$	Ratio
9	8	0.4716675E+00	0.7402666E+00	1.57
25	32	0.2469270E+00	0.4012926E+00	1.63
106	62	0.1219589E+00	0.2320611E+00	1.90
252	184	0.6411837E−01	0.1238655E+00	1.93
657	367	0.2462741E−01	0.4045914E−01	1.64
1468	901	0.9661101E−02	0.1454012E−01	1.51
3314	1842	0.2417861E−02	0.3944454E−02	1.63
7246	3031	0.4405109E−03	0.5927462E−03	1.35
14308	4175	0.5967078E−04	0.8137111E−04	1.36

Table 6.7: H^1 seminorm of errors in *h*-adaptive refinements - Two singularities.

N	NTF	Exact $\|e\|_{1,\Omega}$	Computed $\|e\|_{1,\Omega}$	Ratio
9	8	0.4716675E+00	0.7402666E+00	1.57
25	32	0.2469270E+00	0.4012926E+00	1.63
100	166	0.1274365E+00	0.1931106E+00	1.52
400	730	0.6126349E−01	0.9018663E−01	1.47
1600	3055	0.2764871E−01	0.3569039E−01	1.29
6400	12496	0.1276869E−01	0.1395538E−01	1.09
25600	50528	0.6164818E−02	0.6408720E−02	1.04
40000	79120	0.4694955E−02	0.4836192E−02	1.03

As shown in Figure 6.7 and Figure 6.8, both *h*-adaptive and automatic *hp*-adaptive are able to recognize the regions of the singularities. However, while *h*-adaptive can only use small elements in those regions, automatic *hp*-adaptive can use small elements of low degrees in those regions and bigger elements of higher degree elsewhere. In terms of accuracy, automatic *hp*-adaptive has dominant performance. Automatic *hp*-adaptive achieves the exact error of 0.5967078E-04 with a mesh of just 14308 degrees of freedom whereas *h*-adaptive can only achieve the exact error of 0.4694955E-02 with a mesh of 40000 degrees of freedom. More details can be found in Table 6.6 and Table 6.7

Similar to the case of One singularity, automatic *hp*-adaptive is optimal. Figure 6.9 shows that the rates of convergence of exact errors and computed errors of automatic *hp*-adaptive fit the exponential model (6.1) perfectly.

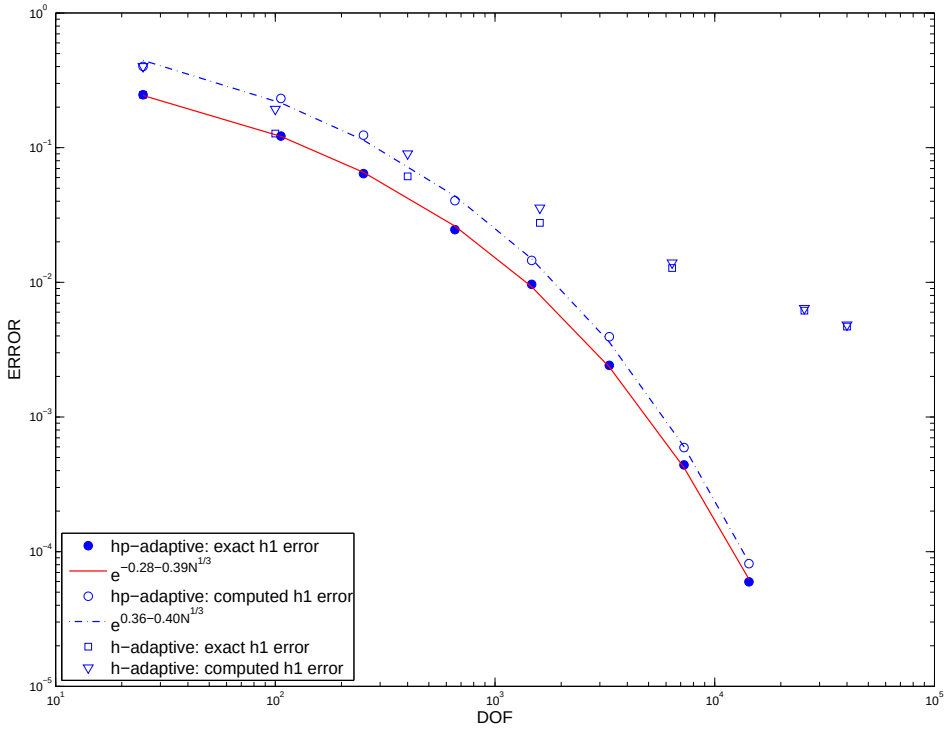


Figure 6.9: Loglog plot of errors with fitting curves - Two singularities.

6.3 Problem Circle

In this experiment, we consider the problem

$$-\nabla \cdot (a\nabla u) = 0 \qquad \text{in } \Omega \tag{6.6a}$$

$$u = f \qquad \text{on } \partial_D\Omega \tag{6.6b}$$

$$\frac{\partial u}{\partial n} = 0 \qquad \text{on } \partial_N\Omega \tag{6.6c}$$

Here Ω is the unit circle with a crack along the positive x axis. $\partial_D\Omega$ is the union of the boundary of the circle and the top of the crack and $\partial_N\Omega$ is the bottom of the crack. The coefficient a is piecewise constant in the eighth sectors

$$\Omega_k = ((r,\theta)|0 \le r \le 1, (k-1)\pi/4 \le \theta \le k\pi/4)$$

For simplicity, we use $a_k = 1$ for all k for computation in this section.

On the top of the crack, we impose homogeneous Dirichlet boundary condition while nonhomogeneous Dirichlet boundary condition is imposed on the boundary of the circle so that the solution is given by

$$u = r^\alpha(\sin\alpha\theta),$$

Here α is chosen to be $1/4$ to correspond to the leading singularity arising from the geometry and the change of boundary condition at the origin. We note that u is not smooth ($u \in H^{5/4-\epsilon}(\Omega)$).

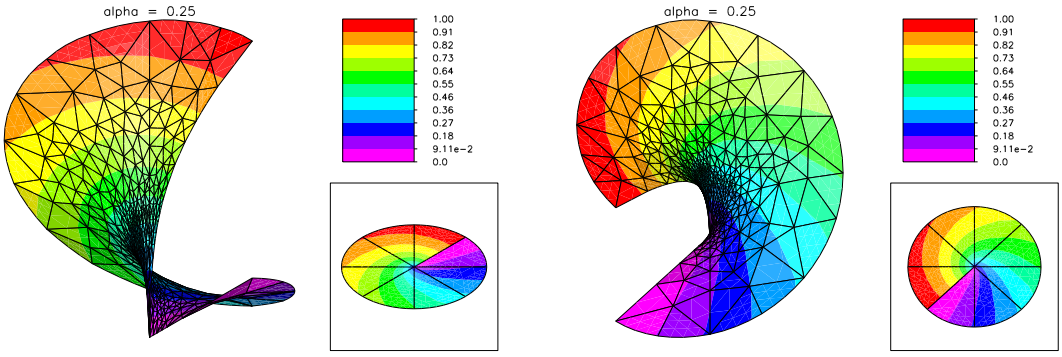


Figure 6.10: The solution viewed from different angles - Problem Circle.

As shown in Figure 6.11 and Figure 6.12, both h-adaptive and automatic hp-adaptive are able to recognize the region of singularity. However, the distribution

of element degree is not as good as in the problem with singularity in section 6.2. Automatic *hp*-adaptive still achieves better exact error of 0.3578414E-02 with a mesh of 184251 degrees of freedom whereas *h*-adaptive can only achieve the exact error of 0.8408233E-02 with a mesh of 348160 degrees of freedom. More details are shown in Table 6.8 and Table 6.9.

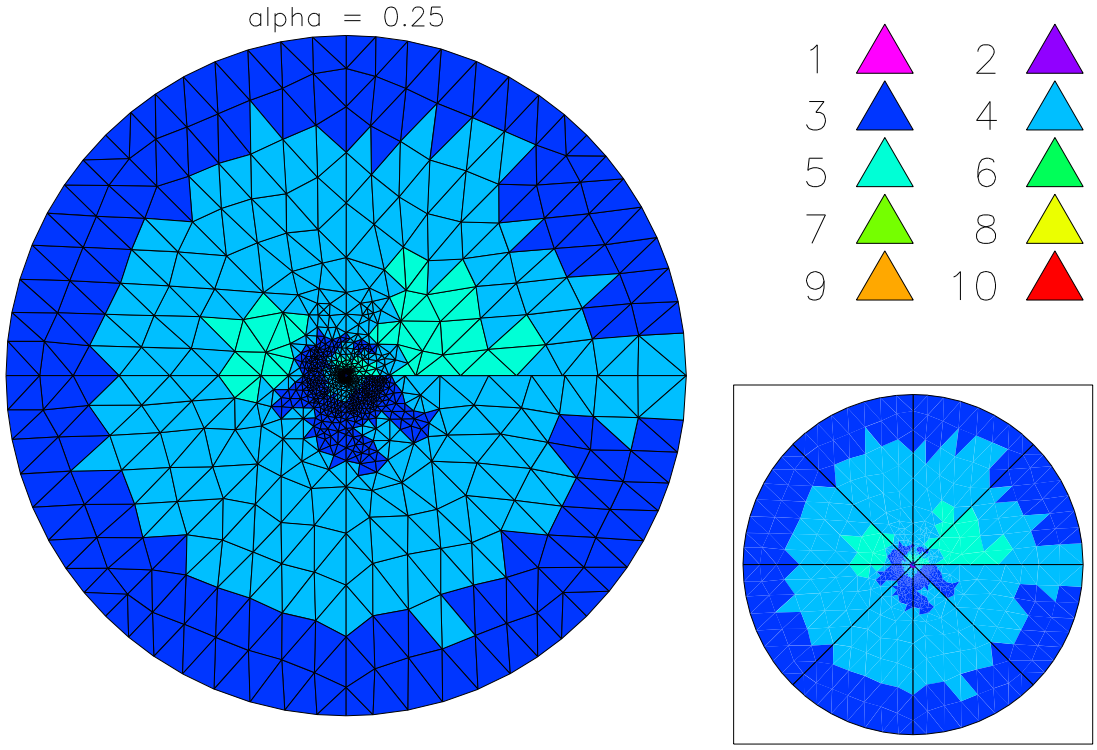


Figure 6.11: An adaptive mesh in automatic *hp*-adaptive ($N = 452220$)
Problem Circle

Even though the final error is still big, all of the exact element errors are very small (around 1e-5) except for few elements near the singularity. This can be verified by Figure 6.13.

As shown in Figure 6.14, the errors of automatic *hp*-adaptive and *h*-adaptive still demonstrate exponential rate of convergence and fit the model (6.1) even though not as well as in the case of the problems with singularity in previous section. In addition, the performance gap between *h*-adaptive and automatic *hp*-adaptive is narrowed in this problem. Probably this is due to the fact that the

Table 6.8: H^1 seminorm of errors in automatic *hp*-adaptive refinements Circle problem.

N	NTF	Exact $\|e\|_{1,\Omega}$	Computed $\|e\|_{1,\Omega}$	Ratio
10	8	0.9366196E+00	0.1194892E+01	1.28
27	32	0.7269343E+00	0.8216088E+00	1.13
85	128	0.5788278E+00	0.6378252E+00	1.10
297	512	0.4690523E+00	0.5058047E+00	1.08
1197	1462	0.2671881E+00	0.2504655E+00	0.94
3902	3700	0.1331213E+00	0.1232696E+00	0.93
10980	9566	0.5765182E−01	0.5695069E−01	0.99
29429	22064	0.2210382E−01	0.2396920E−01	1.08
73383	54562	0.8385920E−02	0.8379619E−02	1.00
184251	132873	0.3578414E−02	0.2799377E−02	0.78

Table 6.9: H^1 seminorm of errors in *h*-adaptive refinements - Problem Circle.

N	NTF	Exact $\|e\|_{1,\Omega}$	Computed $\|e\|_{1,\Omega}$	Ratio
10	8	0.9366196E+00	0.1194892E+01	1.28
27	32	0.7269343E+00	0.8216088E+00	1.13
85	128	0.5788278E+00	0.6378252E+00	1.10
340	627	0.3301023E+00	0.3456646E+00	1.05
1360	2621	0.1845436E+00	0.1828977E+00	0.99
5440	10646	0.9428551E−01	0.9231464E−01	0.98
21760	43001	0.4363149E−01	0.4468371E−01	1.02
87040	172975	0.1912079E−01	0.1949437E−01	1.02
348160	694036	0.8408233E−02	0.8459008E−02	1.01

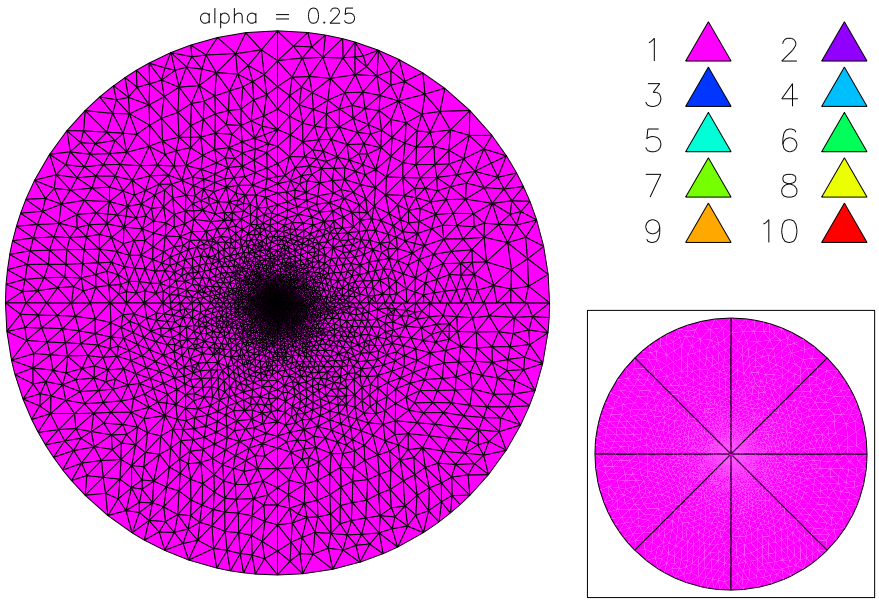


Figure 6.12: An adaptive mesh in h-adaptive ($N = 4753$) - Problem Circle

singular behavior of the problem at the origin is more severe and thus is difficult to capture.

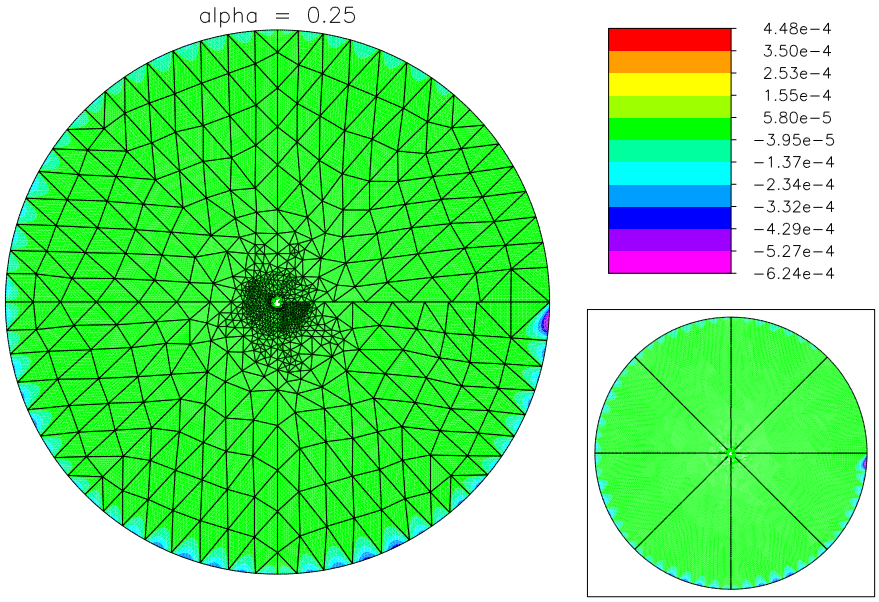


Figure 6.13: Exact element errors in automatic *hp*-adaptive - Problem Cirlce.

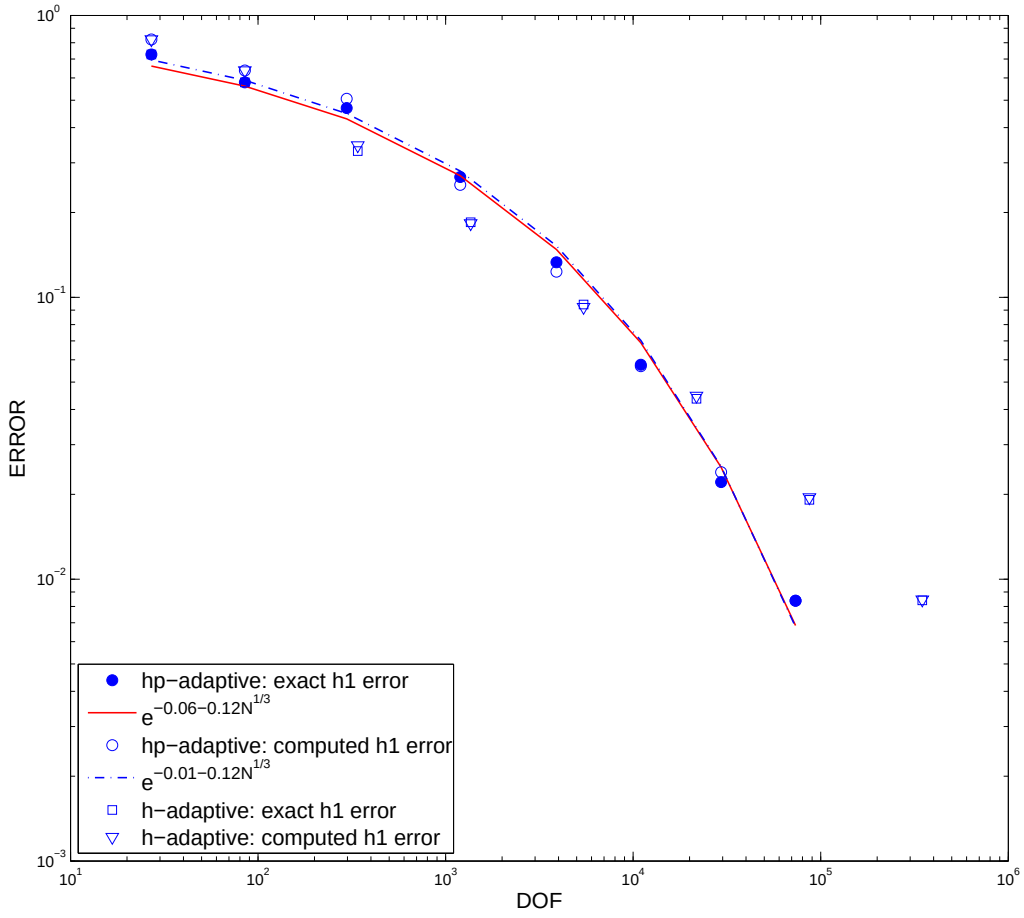


Figure 6.14: Loglog plot of errors with fitting curves - Problem Circle.

6.4 Problem Lake Superior - Domain Decomposition

In this section, we present some numerical results of using *hp*-adaptive meshing in conjunction with domain decomposition. Our examples were run on a LINUX-based Beowulf cluster, consisting of 38 nodes, each with two quad core Xeon processors (2.33GHz) and 16GB of memory. The communication network is a gigabit Ethernet switch. This cluster runs the NPACI ROCKS version of LINUX and employs MPICH2 as its MPI implementation. The computational kernels of PLTMG [17] are written in FORTRAN; the *gfortran* compiler was used in these experiments, invoked using the wrapper *mpif90* and optimization flag *-O*.

In these experiments, we used PLTMG to solve the boundary value problem

$$\begin{aligned} -\Delta u &= 1 \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned}$$

where Ω is a domain shaped like Lake Superior.

In our first experiment, the variant strategy was employed. A mesh of N_P degrees of freedom was created on a single processor using h-adaptive and p-adaptive refinement. Elements on this mesh had different sizes and degrees. This mesh was then broadcast to P processors, where a strategy of combined coarsening and refinement in both h and p was used to transfer approximately $N_P/2$ degrees of freedom from outside Ω_i to inside Ω_i. The global fine mesh was then made h-conforming (geometrically conforming) as described in [18, 19] and p-conforming (degrees agree on shared edges along the interface Γ). Note that the adaptive strategies implemented in PLTMG allow mesh moving and other modifications that yield meshes $\mathcal{T}_i$ that generally are *not* submeshes of the global conforming mesh $\mathcal{T}$ (by definition they are identical on Ω_i and $\partial\Omega_i$). However, PLTMG does insure that the partitions remain geometrically conforming, even in the coarse parts of the domain, and in particular, that the vertices on the interface system in each $\mathcal{T}_i$ are a subset of the vertices of interface system of the global mesh $\mathcal{T}$.

In this experiment, three values of N_P ($400K$, $600K$, and $800K$), and eight values of P (2^k, $1 \le k \le 8$) were used, yielding global fine meshes ranging in size

from about $626K$ to $96.5M$ unknowns. Because our cluster had only 38 nodes, for larger values of P, we simulated the behavior of a larger cluster in the usual way, by allowing nodes to have multiple processes.

In these experiments, the convergence criterion was

$$\frac{\|\delta U^k\|_G}{\|U^k\|_G} \leq \frac{\|\delta U^0\|_G}{\|U^0\|_G} \times 10^{-3}. \tag{6.7}$$

This is more stringent than necessary for purposes of computing an approximation to the solution of the partial differential equation, but it allows us to illustrate the behavior of the solver as an iterative method for solving linear systems of equations.

Table 6.10 summarizes this computation. The columns labeled DD indicate the number of domain decomposition iterations required to satisfy the convergence criteria (6.7). For comparison, the number of iterations needed to satisfy the actual convergence criterion used in PLTMG, based on reducing the error in the solution of the linear system to the level of the underlying approximation error, is given in parentheses. From these results it is clear that the number of iterations is stable and largely independent of N and P over this range of values. The size of the global mesh for the variant strategy can be estimated from the formula

$$N \approx \theta P N_P + N_P \tag{6.8}$$

where $\theta = 1/2$. Equation (6.8) predicts an upper bound, as it does not account for refinement outside of Ω_i and coarsening inside Ω_i, needed to keep the mesh conforming and for other reasons. For $N_P = 800K$, $P = 256$, (6.8) predicts $N \approx 103200000$, where the observed $N = 96490683$.

In our second experiment we solved the same problem using the original paradigm. On one processor, an adaptive mesh of size $N_c = 50K$ was created. All elements on this mesh were linear elements. This mesh was then partitioned into P subregions, $P = 2^k$, $1 \leq k \leq 8$. This coarse mesh was broadcast to P processors (simulated as needed) and each processor continued the adaptive process in both h and p, creating a mesh of size N_P. In this experiment, N_P was chosen to be $400K$, $600K$, and $800K$. This resulted in global meshes varying in

Table 6.10: Convergence Results for Variant Algorithm. Numbers of iterations needed to satisfy (6.7) are given in the column labeled DD. The numbers in parentheses are the number of iterations required to satisfy the actual convergence criterion used by PLTMG.

	$N_P = 400K$		$N_P = 600K$		$N_P = 800K$	
P	N	DD	N	DD	N	DD
2	625949	10 (3)	776381	8 (3)	1390124	12 (4)
4	1189527	13 (4)	1790918	11 (4)	2288587	9 (3)
8	1996139	10 (4)	2990807	13 (4)	3993126	10 (3)
16	3569375	14 (4)	5220706	13 (4)	6920269	12 (3)
32	6723697	13 (3)	9736798	16 (4)	13142670	11 (3)
64	12978568	11 (4)	18905909	14 (4)	25326662	11 (3)
128	25155124	12 (3)	37148571	10 (4)	48841965	10 (3)
256	48874991	11 (3)	72902698	14 (4)	96490683	11 (3)

Table 6.11: Convergence Results for Original Algorithm. Numbers of iterations needed to satisfy (6.7) are given in the column labeled DD. The numbers in parentheses are the number of iterations required to satisfy the actual convergence criterion used by PLTMG.

	$N_P = 400K$		$N_P = 600K$		$N_P = 800K$	
P	N	DD	N	DD	N	DD
2	750225	13 (4)	1150106	13 (4)	1549915	13 (4)
4	1450054	13 (4)	2248841	13 (4)	3047906	13 (4)
8	2846963	9 (3)	4442665	9 (4)	6039743	9 (3)
16	5635327	11 (4)	8821463	10 (4)	12010188	11 (4)
32	11204214	12 (4)	17564640	10 (4)	23930867	11 (4)
64	22301910	14 (4)	34983543	13 (4)	47693190	13 (4)
128	44408605	11 (4)	69696605	12 (4)	95026759	11 (4)
256	88369503	11 (3)	138790801	11 (3)	189363322	11 (4)

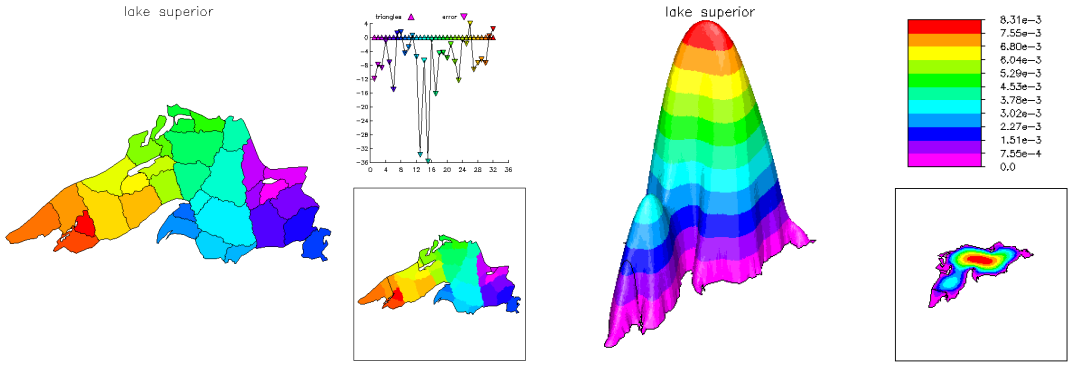


Figure 6.15: The load balance (left) and solution (right) in the case $N_P = 800K$, $P = 32$.

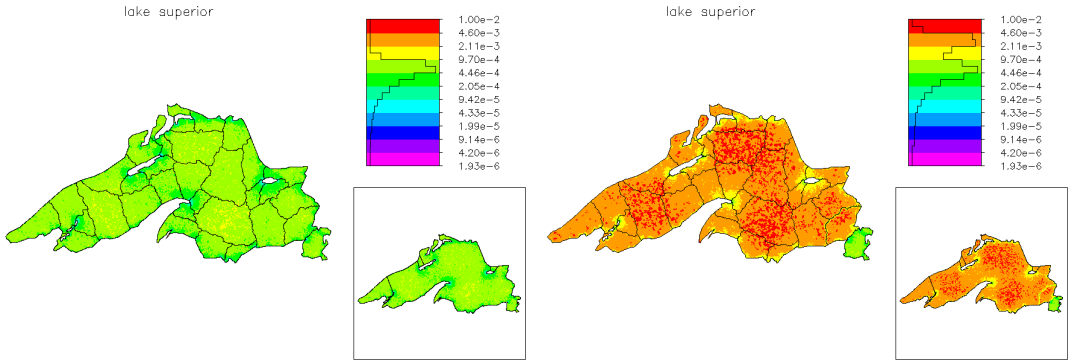


Figure 6.16: The mesh density for the global mesh (left) and for one of the local meshes (right) in the case $N_P = 800K$, $P = 32$.

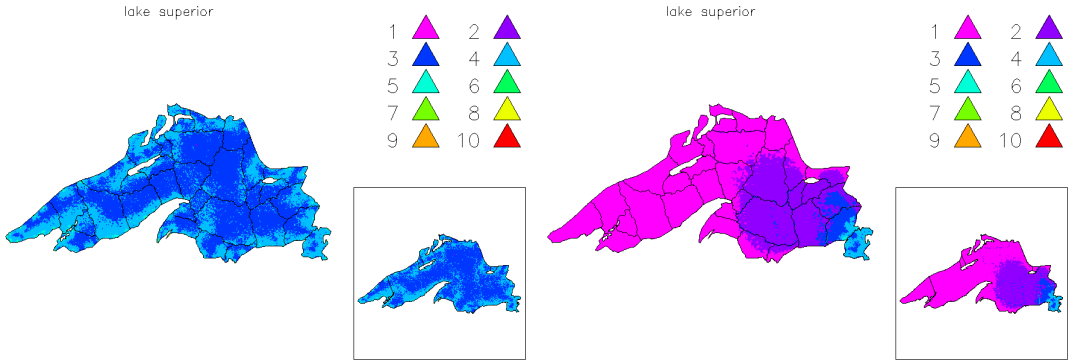


Figure 6.17: The degree density for the global mesh (left) and for one of the local meshes (right) in the case $N_P = 800K$, $P = 32$.

size from approximately $750K$ to $189M$. These global meshes were regularized to be h-conforming and p-conforming, and a global DD solve was made as in the first experiment. As in the first experiment, the usual convergence criteria was replaced by (6.7) in order to illustrate the dependence of the convergence rate on N and P. The results are summarized in Table 6.11.

For the original paradigm the size of the global mesh is predicted by

$$N \approx PN_P - (P-1)N_c. \tag{6.9}$$

Similar to equation (6.8), equation (6.9) only predicts an upper bound, as it does not account for refinement outside of Ω_i, needed to keep the mesh conforming and for other reasons. For example, for N_c=50K, N_P=800K, $P = 256$, (6.9) predicts $N \approx 192050000$ when actually $N = 189363322$. For the case $N_P = 800K$, $P = 32$, the solution and the load balance is shown in Figure 6.15. The mesh density and degree density of the global mesh and one local mesh are shown in Figure 6.16 and Figure 6.17. As expected, both the mesh density and the degree density are high in the local region and much lower elsewhere in the local mesh.

Appendix A

Barycentric Coordinates

Barycentric coordinates are coordinates defined by the vertices of a simplex (a triangle, tetrahedron, etc).

Definition A.1. *Let $P \in \mathbb{R}^2$ be a point with Cartesian coordinates (x, y) and $\triangle ABC \in \mathbb{R}^2$ be a triangle with vertices $v_k = (x_k, y_k), k = 1, 3$. Then (c_1, c_2, c_3) is said to be the barycentric coordinates of P with respect to $\triangle ABC$ if and only if*

$$\begin{cases} c_1 + c_2 + c_3 & = 1 \\ c_1 x_1 + c_2 x_2 + c_3 x_3 & = x \\ c_1 y_1 + c_2 y_2 + c_3 y_3 & = y \end{cases} . \tag{A.1}$$

Sometimes equation (A.1) is also written in the form of

$$\begin{pmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 1 \\ x \\ y \end{pmatrix} . \tag{A.2}$$

Figure A.1 shows some special barycentric coordinates on a triangle. Barycentric coordinates with respect to a triangle are also known as area coordinates, because the barycentric coordinates of P with respect to $\triangle ABC$ are proportional to the signed areas of $\triangle PBC, \triangle PCA$, and $\triangle PAB$. Here the signed area of a triangle is defined as in equation (3.2). With this property, one could use barycentric coordinates to determine if a point belongs to the interior of a triangle. Figure A.2 shows different portions of the plane with signs of associated barycentric coordinates.

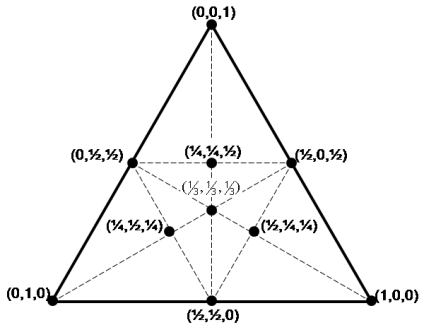


Figure A.1: Barycentric coordinates.

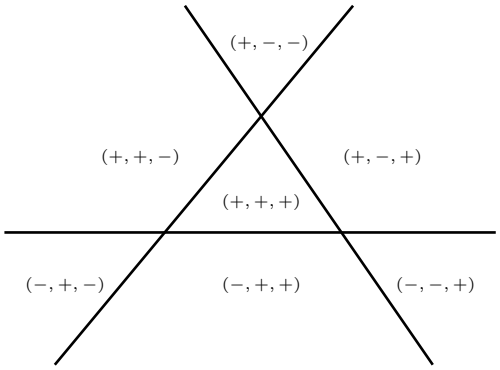


Figure A.2: Signs of barycentric coordinates.

In addition, a point with barycentric coordinates (c_1, c_2, c_3) with respect to $\triangle ABC$ can also be thought of as the barycenter or the center of mass of masses equal to c_1, c_2, c_3 attached at vertices A, B, C respectively. This is actually the origin of the term "barycentric" introduced by August Ferdinand Mobius in 1827.

Obviously, with the last two equations of (A.1), we could easily convert barycentric coordinates of a point to its Cartesian coordinates. In order to do the reverse, we substitute the first equation of (A.1) into the last two. After some algebra, we get

$$\begin{cases} c_1 = \frac{(y_2-y_3)(x-x_3)+(x_3-x_2)(y-y_3)}{(x_1-x_3)*(y_2-y_3)-(x_2-x_3)*(y_1-y_3)}, \\ c_2 = \frac{(y_3-y_1)(x-x_3)+(x_1-x_3)(y-y_3)}{(x_1-x_3)*(y_2-y_3)-(x_2-x_3)*(y_1-y_3)}, \\ c_3 = 1 - c_1 - c_2 \end{cases} . \tag{A.3}$$

Fortunately, we rarely need to use this equation.

Now we prove some results related to barycentric coordinates.

Proposition A.2. *Let (c_1, c_2, c_3) be the barycentric coordinates of a point P with respect to $\triangle ABC$. Then $c_i, i = 1, \ldots, 3$ is the ratio between the distance of P to the ith edge and the length of the ith altitude.*

Proof. It is sufficient to show the result for $i = 1$.

Assume that vertices of $\triangle ABC$ have Cartesian coordinates as in Definition A.1. Then the equation for edge BC is

$$\frac{x - x_2}{x_3 - x_2} - \frac{y - y_2}{y_3 - y_2} = 0$$

This implies that the distance from $A = (x_1, y_2)$ to BC is

$$k \left| \frac{x_1 - x_2}{x_3 - x_2} - \frac{y_1 - y_2}{y_3 - y_2} \right|, \tag{A.4}$$

where

$$k = \left(\frac{1}{(x_3 - x_2)^2} + \frac{1}{(y_3 - y_2)^2} \right)^{\frac{1}{2}}.$$

Let (x_P, y_P) be the Cartesian coordinates of P. According to equation (A.1)

$$\begin{cases} x_P = c_1 x_1 + c_2 x_2 + c_3 x_3 \\ y_P = c_1 y_1 + c_2 y_2 + c_3 y_3 \end{cases}.$$

Then, the distance from P to BC is

$$\begin{aligned}
& k \left| \frac{c_1 x_1 + c_2 x_2 + c_3 x_3 - x_2}{x_3 - x_2} - \frac{c_1 y_1 + c_2 y_2 + c_3 y_3 - y_2}{y_3 - y_2} \right| \\
=& k \left| \frac{c_1 x_1 + (1 - c_2) x_2 + c_3 x_3}{x_3 - x_2} - \frac{c_1 y_1 + (1 - c_2) y_2 + c_3 y_3}{y_3 - y_2} \right| \\
=& k \left| \frac{c_1 x_1 - (c_1 + c_3) x_2 + c_3 x_3}{x_3 - x_2} - \frac{c_1 y_1 - (c_1 + c_3) y_2 + c_3 y_3}{y_3 - y_2} \right| \\
=& k \left| \frac{c_1 (x_1 - x_2)}{x_3 - x_2} + c_3 - \left(\frac{c_1 (y_1 - y_2)}{y_3 - y_2} + c_3 \right) \right| \\
=& k\, c_1 \left| \frac{x_1 - x_2}{x_3 - x_2} - \frac{y_1 - y_2}{y_3 - y_2} \right|
\end{aligned} \tag{A.5}$$

From equation (A.4) and equation (A.5), we have c_1 as the ratio between the distance from P to BC and the first altitude. □

The following results follow immediately after Proposition A.2.

Corollary A.3. *With respect to* $\triangle ABC$

(i) If the i*th barycentric coordinates of* P *and* P' *are the same, then* PP' *is parallel to the* i*th edge.*

(ii) The set of points with the i*th barycentric coordinate equal* c_i *is the line that is parallel to the* i*th edge and has distance to that edge equalling* $c_i h_i$*. Here* h_i *is the length of the* i*th altitude and we assume that the distances are negative for lines in the half plane (defined by the* i*th edge) not containing the* i*th vertex.*

(iii) i*th edge is the set of points with* i*th barycentric coordinate* $c_i = 0$*.*

Remark A.4. *With respect to* $\triangle ABC$*, if we consider the* i*th barycentric coordinate* c_i *as a function of* x *and* y*, then* c_i *has the same formula with the* i*th linear nodal basis function associated with* $\triangle ABC$*. Therefore, barycentric coordinates are sometimes used to refer to linear nodal basis functions and vice versa. In this dissertation, we use these two terms interchangeably.*

Appendix B

Numerical Quadrature

Numerical quadrature is used in many (if not almost every) scientific program, especially the ones related to finite element methods. There are several reasons for its popularity. Firstly, the integrand may be known only at certain points, such as those obtained by sampling. Secondly, a formula for the integrand may be known, but its integral can not be written as an explicit formula. An example of such integrand is $f(x) = exp(-x^2)$. Lastly and most importantly, numerical quadrature provides programs with more flexibility, and easy implementation. With numerical quadrature, even if the problem is changed completely, we only need to redefine the way we calculate the integrand and the part of code using numerical quadrature itself remains unchanged.

For finite element methods with quadrilateral or hexahedral elements, one could use quadrature formulas derived from tensor products of one dimensional Gaussian quadrature rules. In our case, however, we use elements of triangular shape. Therefore we need special quadrature rules designed for triangular integral domains.

Let t be a domain of triangular shape. A quadrature rule $\mathcal{R}$ on t is defined as a set of point and weight pairs:

$$\mathcal{R} = \{(p_i, w_i), i = 1, \ldots, n\}$$

such that for any function $f(x)$ defined on t, its integral on t can be approximated

by

$$\int_t f(x)dx \approx |t| \sum_{i=1}^{n} w_i f(p_i).$$

Here $|t|$ is the area of t, n the number of points, p_i the quadrature points, and w_i the associated weights.

In finite element methods, functions are integrated on each element. Those functions sometimes are defined element by element. They might have jumps or might not even be defined on part of element boundary. Therefore we only use quadrature rules with quadrature points that are inside the integral domain. In addition, it is also common to use rules with positive weight only.

In dealing with triangular domains, it is more convenient to use barycentric coordinates (see A). In order for a quadrature point to be inside the integral domain, its barycentric coordinates need to be in the interval $(0, 1)$.

A quadrature rule R is said to be symmetric if it is invariant under permutations of the barycentric coordinates. That is, if (c_1, c_2, c_3) is a quadrature point of $\mathcal{R}$ associated with weight w, then for any permutation (i_1, i_2, i_3) of $(1, 2, 3)$, the point$(c_{i_1}, c_{i_2}, c_{i_3})$ is also a quadrature point of R with the same weight w. For symmetric quadrature rules of triangular domains, quadrature points can be divided into separated symmetry orbits, each of which contains all the points generated by permuting the barycentric coordinates of a single point. These symmetry orbits can be classified into three different permutation stars described in detail in Table B.1.

Table B.1: Permutation stars on a triangle.

permutation star	barycentric coordinates	number of points
$S_3(\frac{1}{3})$	$(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$	1
$S_{21}(a)$	$(a, a, 1-2a)$	3
$S_{111}(a, b)$	$(a, b, 1-a-b)$	6

A quadrature rule is said to be of (accuracy) order p if it is exact for all polynomials of degree equal or less than p. In PLTMG, when we use elements of high degree, it is critical that we have quadrature rules with high accuracy order. For an element of degree p, the associated error functions are polynomials of degree $p+1$.

Therefore, in order to guarantee sufficient accuracy we need to use quadrature rules of order at least $2(p+1)$ (in computing norm of errors, the integrands are polynomials of degree $2(p+1)$ since they are square of error functions).

Currently, PLTMG incorporates quadrature rules of order $2, \ldots, 22$. This allows elements in PLTMG to have degrees up to 10. The quadrature rules used in PLTMG come from the paper [59] of Lin-bo Zhang, Tao Cui and Hui Liu.

Index

Bibliography

[1] Milton Abramowitz and Irene A. Stegun (eds.), *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, Dover Publications Inc., New York, 1992, Reprint of the 1972 edition. MR MR1225604 (94b:00012)

[2] I. Babuška and M. R. Dorr, *Error estimates for the combined h and p versions of the finite element method*, Numer. Math. **37** (1981), no. 2, 257–277. MR MR623044 (82h:65080)

[3] I. Babuška and B. Q. Guo, *Approximation properties of the h-p version of the finite element method*, Comput. Methods Appl. Mech. Engrg. **133** (1996), no. 3-4, 319–346. MR MR1399640 (98k:73063)

[4] I. Babuška, B. Q. Guo, and E. P. Stephan, *On the exponential convergence of the h-p version for boundary element Galerkin methods on polygons*, Math. Methods Appl. Sci. **12** (1990), no. 5, 413–427. MR MR1053063 (91i:65174)

[5] I. Babuška and B.Q. Guo, *The hp version of the finite element method for domains with curved boundaries*, SIAM Journal on Numerical Analysis (1988), 837–861.

[6] I. Babuška, R. B. Kellogg, and J. Pitkäranta, *Direct and inverse error estimates for finite elements with mesh refinements*, Numer. Math. **33** (1979), no. 4, 447–471. MR MR553353 (81c:65054)

[7] I. Babuška and W. C. Rheinboldt, *Error estimates for adaptive finite element computations*, SIAM J. Numer. Anal. **15** (1978), no. 4, 736–754. MR MR0483395 (58 #3400)

[8] I. Babuška, EP Stephan, and B.Q. Guo, *The hp version of the boundary element method with geometric mesh on polygonal domains*, (1989).

[9] I. Babuška and M. Suri, *The h-p version of the finite element method with quasi-uniform meshes*, RAIRO Modél. Math. Anal. Numér. **21** (1987), no. 2, 199–238. MR MR896241 (88d:65154)

[10] ———, *The optimal convergence rate of the p-version of the finite element method*, SIAM J. Numer. Anal. **24** (1987), no. 4, 750–776. MR MR899702 (88k:65102)

[11] I. Babuška, B. A. Szabo, and I. N. Katz, *The p-version of the finite element method*, SIAM J. Numer. Anal. **18** (1981), no. 3, 515–545. MR MR615529 (82j:65081)

[12] Ivo Babuška and Theofanis Strouboulis, *The finite element method and its reliability*, Numerical Mathematics and Scientific Computation, The Clarendon Press Oxford University Press, New York, 2001. MR MR1857191 (2002k:65001)

[13] Randolph E. Bank, *Multigraph users' guide - version 1.0*, Tech. report, Department of Mathematics, University of California at San Diego, 2001.

[14] ———, *A domain decomposition solver for a parallel adaptive meshing paradigm*, Domain Decomposition Methods in Science and Engineering XVI (Olof B. Widlund and David E. Keyes, eds.), Lecture Notes in Computational Science and Engineering, vol. 55, Springer-Verlag, 2006, pp. 3–14.

[15] Randolph E. Bank, *Some variants of the Bank-Holst parallel adaptive meshing paradigm*, Comput. Vis. Sci. **9** (2006), no. 3, 133–144. MR MR2271791

[16] Randolph E. Bank, *Some variants of the Bank-Holst parallel adaptive meshing paradigm*, Computing and Visualization in Science **9** (2006), 133–144.

[17] ———, *PLTMG: A software package for solving elliptic partial differential equations, users' guide 10.0*, Tech. report, Department of Mathematics, University of California at San Diego, 2007.

[18] Randolph E. Bank and Michael Holst, *A new paradigm for parallel adaptive meshing algorithms*, SIAM J. Sci. Comput. **22** (2000), no. 4, 1411–1443 (electronic). MR MR1797889 (2002g:65117)

[19] ———, *A new paradigm for parallel adaptive meshing algorithms*, SIAM Rev. **45** (2003), no. 2, 291–323 (electronic), Reprinted from SIAM J. Sci. Comput. **2**2 (2000), no. 4, 1411–1443 [MR1797889]. MR MR2010380

[20] Randolph E. Bank and Peter K. Jimack, *A new parallel domain decomposition method for the adaptive finite element solution of elliptic partial differential equations*, Concurrency and Computation: Practice and Experience **13** (2001), 327–350.

[21] Randolph E. Bank, Peter K. Jimack, Sarfraz A. Nadeem, and Sergei V. Nepomnyaschikh, *A weakly overlapping domain decomposition preconditioner*

for the finite element solution of elliptic partial differential equations, SIAM J. on Scientific Computing **23** (2002), 1817–1841.

[22] Randolph E. Bank and Shaoying Lu, *A domain decomposition solver for a parallel adaptive meshing paradigm*, SIAM J. on Scientific Computing **26** (2004), 105–127 (electronic).

[23] Randolph E. Bank and Hieu T. Nguyen, *Domain decomposition and hp finite elements*, Domain decomposition methods in science and engineering XIX, Lect. Notes Comput. Sci. Eng., Springer, Berlin, to appear.

[24] Randolph E. Bank, Andrew H. Sherman, and Alan Weiser, *Refinement algorithms and data structures for regular local mesh refinement*, Scientific computing (Montreal, Que., 1982), IMACS Trans. Sci. Comput., I, IMACS, New Brunswick, NJ, 1983, pp. 3–17. MR MR751598

[25] Randolph E. Bank and R. Kent Smith, *Mesh smoothing using a posteriori error estimates*, SIAM J. Numer. Anal. **34** (1997), no. 3, 979–997. MR MR1451110 (98m:65162)

[26] Randolph E. Bank and Alan Weiser, *Some a posteriori error estimators for elliptic partial differential equations*, Math. Comp. **44** (1985), no. 170, 283–301. MR MR777265 (86g:65207)

[27] Randolph E. Bank and Jinchao Xu, *Asymptotically exact a posteriori error estimators. I. Grids with superconvergence*, SIAM J. Numer. Anal. **41** (2003), no. 6, 2294–2312 (electronic). MR MR2034616 (2004k:65194)

[28] ———, *Asymptotically exact a posteriori error estimators. II. General unstructured grids*, SIAM J. Numer. Anal. **41** (2003), no. 6, 2313–2332 (electronic). MR MR2034617 (2004m:65212)

[29] Randolph E. Bank, Jinchao Xu, and Bin Zheng, *Superconvergent derivative recovery for Lagrange triangular elements of degree p on unstructured grids*, SIAM J. Numer. Anal. **45** (2007), no. 5, 2032–2046 (electronic). MR MR2346369 (2009b:65293)

[30] Susanne C. Brenner and L. Ridgway Scott, *The mathematical theory of finite element methods*, third ed., Texts in Applied Mathematics, vol. 15, Springer, New York, 2008. MR MR2373954 (2008m:65001)

[31] Tony F. Chan, P. Ciarlet, Jr., and W. K. Szeto, *On the optimality of the median cut spectral bisection graph partitioning method*, SIAM J. Sci. Comput. **18** (1997), no. 3, 943–948. MR MR1443649 (98d:65044)

[32] P. G. Ciarlet, *Basic error estimates for elliptic problems*, Handbook of numerical analysis, Vol. II, Handb. Numer. Anal., II, North-Holland, Amsterdam, 1991, pp. 17–351. MR MR1115237

[33] L. Demkowicz, J. T. Oden, W. Rachowicz, and O. Hardy, *Toward a universal h-p adaptive finite element strategy, part 1. constrained approximation and data structure*, Computer Methods in Applied Mechanics and Engineering **77** (1989), no. 1-2, 79 – 112.

[34] L. Demkowicz, W. Rachowicz, and Ph. Devloo, *A fully automatic hp-adaptivity*, Proceedings of the Fifth International Conference on Spectral and High Order Methods (ICOSAHOM-01) (Uppsala), vol. 17, 2002, pp. 117–142. MR MR1910555

[35] Gene H. Golub and Charles F. Van Loan, *Matrix computations*, third ed., Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, 1996. MR MR1417720 (97g:65006)

[36] W. Gui and I. Babuška, *The h, p and h-p versions of the finite element method in 1 dimension*, Parts 1, 2, 3, Numerische Mathematik **49** (1986), no. 6, 577–683.

[37] Benqi Guo and Weiwei Sun, *The optimal convergence of the h-p version of the finite element method with quasi-uniform meshes*, SIAM J. Numer. Anal. **45** (2007), no. 2, 698–730 (electronic). MR MR2300293 (2008c:65325)

[38] B.Q. Guo, *The h- p version of the finite element method for elliptic equations of order2m*, Numerische Mathematik **53** (1988), no. 1, 199–224.

[39] B.Q. Guo and I. Babuška, *The h-p version of the finite element method - part 1: The basic approximation results*, Computational Mechanics **1** (1986), no. 1, 21–41.

[40] ———, *The h-p version of the finite element method - part 2: General results and applications*, Computational Mechanics **1** (1986), no. 3, 203–220.

[41] ———, *The h-p version of the finite element method - parts 1,2*, Computational Mechanics **1** (1986), no. 1, 21–41, 203–220.

[42] V. Heuveline and R. Rannacher, *Duality-based adaptivity in the hp-finite element method*, J. Numer. Math. **11** (2003), no. 2, 95–113. MR MR1987590 (2004m:65196)

[43] Yunqing Huang and Jinchao Xu, *Superconvergence of quadratic finite elements on mildly structured grids*, Math. Comp. **77** (2008), no. 263, 1253–1268. MR MR2398767 (2009h:65184)

[44] Bo Li, *Lagrange interpolation and finite element superconvergence*, Numer. Methods Partial Differential Equations **20** (2004), no. 1, 33–59. MR MR2020249 (2004m:65199)

[45] Joachim A. Nitsche and Alfred H. Schatz, *Interior estimates for Ritz-Galerkin methods*, Math. Comp. **28** (1974), 937–958. MR MR0373325 (51 #9525)

[46] J. T. Oden, L. Demkowicz, W. Rachowicz, and T. A. Westermann, *Toward a universal h-p adaptive finite element strategy, part 2. a posteriori error estimation*, Computer Methods in Applied Mechanics and Engineering **77** (1989), no. 1-2, 113 – 180.

[47] J. T. Oden, L. Demkowicz, T. Strouboulis, and P. Devloo, *Adaptive methods for problems in solid and fluid mechanics*, Accuracy estimates and adaptive refinements in finite element computations (Lisbon, 1984), Wiley Ser. Numer. Methods Engrg., Wiley, Chichester, 1986, pp. 249–280. MR MR879450 (88d:73010)

[48] W. Rachowicz, J. T. Oden, and L. Demkowicz, *Toward a universal h-p adaptive finite element strategy part 3. design of h-p meshes*, Computer Methods in Applied Mechanics and Engineering **77** (1989), no. 1-2, 181 – 212.

[49] Rolf Rannacher, *The dual-weighted-residual method for error control and mesh adaptation in finite element methods*, The mathematics of finite elements and applications, X, MAFELAP 1999 (Uxbridge), Elsevier, Oxford, 2000, pp. 97–116. MR MR1801971 (2001m:65153)

[50] M.-Cecilia Rivara, *Algorithms for refining triangular grids suitable for adaptive and multigrid techniques*, Internat. J. Numer. Methods Engrg. **20** (1984), no. 4, 745–756. MR MR739618 (85h:65258)

[51] María-Cecilia Rivara, *Selective refinement/derefinement algorithms for sequences of nested triangulations*, Internat. J. Numer. Methods Engrg. **28** (1989), no. 12, 2889–2906. MR MR1030410 (90j:57018)

[52] Ivo G. Rosenberg and Frank Stenger, *A lower bound on the angles of triangles constructed by bisecting the longest side*, Math. Comp. **29** (1975), 390–395. MR MR0375068 (51 #11264)

[53] Gilbert Strang, *Piecewise polynomials and the finite element method*, Bull. Amer. Math. Soc. **79** (1973), 1128–1137. MR MR0327060 (48 #5402)

[54] Martin Stynes, *On faster convergence of the bisection method for all triangles*, Math. Comp. **35** (1980), no. 152, 1195–1201. MR MR583497 (81j:51023)

[55] BA Szabo, *Mesh design for the p-version of the finite element method*, Computer Methods in Applied Mechanics and Engineering **55** (1986), no. 1-2, 197.

[56] Barna Szabó and Ivo Babuška, *Finite element analysis*, A Wiley-Interscience Publication, John Wiley & Sons Inc., New York, 1991. MR MR1164869 (93f:73001)

[57] Lars B. Wahlbin, *Local behavior in finite element methods*, Handbook of numerical analysis, Vol. II, Handb. Numer. Anal., II, North-Holland, Amsterdam, 1991, pp. 353–522. MR MR1115238

[58] S. Wandzura and H. Xiao, *Symmetric quadrature rules on a triangle*, Comput. Math. Appl. **45** (2003), no. 12, 1829–1840. MR MR1995755 (2004e:65026)

[59] Linbo Zhang, Tao Cui, and Hui Liu, *A set of symmetric quadrature rules on triangles and tetrahedra*, J. Comput. Math. **27** (2009), no. 1, 89–96. MR MR2493559 (2009k:65045)