INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6” x 9” black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®
NOTE TO USERS

This reproduction is the best copy available.

UMI
RICE UNIVERSITY

SUPERCONVERGENT SECOND DERIVATIVE
RECOVERY METHODS

by

Christine Michelle Jefferson

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

APPROVED, THESIS COMMITTEE:

[Signatures]

Dr. John E. Akin, Chairman
Professor
Mechanical Engineering and Materials
Science

Dr. Chad M. Landis
Assistant Professor
Mechanical Engineering and Materials
Science

Dr. Michael B. Heeley
Assistant Professor
Jesse H. Jones Graduate School of
Management

Houston, Texas
April 2001
SUPERCONVERGENT SECOND DERIVATIVE RECOVERY METHODS

Christine Michelle Jefferson

Abstract

A method of evaluating super-convergent second derivative recovery methods is proposed. The second derivative and exact flux error of three super-convergent patch (SCP) options are compared to determine the more effective method. The three super-convergent patch options employed are: the element based patch (all neighbors), the element based patch (facing neighbors), and the nodal based patch (all element neighbors). To witness the behavior of the different SCP options, different boundary conditions and element types are evaluated. The value of each SCP option is further investigated by comparing the uniform mesh refinement with the more cost effective h-refinement method.
Acknowledgments

First of all, I would like to thank God for all my many blessings.

I would like to show my deepest appreciation for my advisor, Dr Akin, for his time, commitment, and patience. I also offer my thanks to the members of my committee Dr. Michael B. Heeley and Dr. Chad Landis.

Last but not least, I would like to give thanks to all of my friends for their love and support. Special thanks to Powtache Williams, Victor Udoewa, and Marlon Stewart for their encouragement. And to my family members who have given me the love that I need; Carol, A.J., and my niece, Ariel C'nae Johnson, who always make me smile.
# Table of Contents

Abstract ii
Acknowledgments iii
List of Figures viii

1 Introduction 1
1.1 History 1
1.1.1 Error Estimation 1
1.2 Thesis Objective 2

2 Literature Survey 4
2.1 Error Estimation 4
2.1.1 Kelly 4
2.1.2 Zienkiewicz and Zhu 4
2.2 Patch Recovery 5
2.2.1 Zienkiewicz and Zhu 5
2.2.2 Aalto 5
2.3 First Derivative 6
2.3.1 Zienkiewicz and Zhu 6
2.3.2 Wilberg and Wahab 6
2.4 Second Derivative 6
2.4.1 Lakhany and Whiteman 6
3 Derivative Recovery and Error Estimation for Elliptic Problems

3.1 Introduction .................................................. 8
3.2 Error Estimates ............................................... 13
3.3 Super-Convergent Patch Recovery ............................ 14
3.4 Hierarchical Error Indicator ................................. 18
3.5 Flux Balancing Error Estimates .............................. 22

4 Super-Convergent Patch Recovery ............................. 24

4.1 Patch Implementation ......................................... 24
4.2 SCP Nodal Flux Averaging ...................................... 29
4.3 Computing the SCP Element Error Estimates ................. 33
4.4 H-Adaptivity .................................................. 37
4.5 Second Derivative Recovery .................................... 37

5 Two-Dimensional Whiteman Square ............................ 41

5.1 Full Whiteman Square for T3 Elements ....................... 44
  5.1.1 Geometry .................................................. 44
  5.1.2 Analysis ................................................... 44
5.2 Full Whiteman Square for Q4 Elements ....................... 47
  5.2.1 Geometry .................................................. 47
  5.2.2 Analysis ................................................... 47
5.3 Full Whiteman Square for T6 Elements ....................... 50
  5.3.1 Geometry .................................................. 50
  5.3.2 Analysis ................................................... 50
5.4 One-Fourth Whiteman Square for T3 Elements ............... 52
  5.4.1 Geometry .................................................. 52
5.4.2 Analysis ........................................ 52
5.5 One-Fourth Whiteman Square for Q4 Elements .......... 54
  5.5.1 Geometry ..................................... 54
  5.5.2 Analysis ..................................... 54
5.6 One-Fourth Whiteman Square for T6 Elements .......... 56
  5.6.1 Geometry ..................................... 56
  5.6.2 Analysis ..................................... 56
5.7 One-Eighth Whiteman Square for T3 Elements .......... 58
  5.7.1 Geometry ..................................... 58
  5.7.2 Analysis ..................................... 58
5.8 One-Eighth Whiteman Square for Q4 Elements .......... 60
  5.8.1 Geometry ..................................... 60
  5.8.2 Analysis ..................................... 60
5.9 One-Eighth Whiteman Square for T6 Elements .......... 62
  5.9.1 Geometry ..................................... 62
  5.9.2 Analysis ..................................... 62
5.10 Results ........................................ 64
  5.10.1 Element Comparison .............................. 64
  5.10.2 Boundary Condition Comparison .................... 67

6 H-Adaptive Refinement of the Lakhany Whiteman Square 70
  6.1 Full Model ..................................... 71
  6.2 One-Eighth Model ................................ 76

7 Conclusion and Future Work ................................ 80
  7.1 Conclusion ..................................... 80
  7.2 Future Work .................................... 81
Bibliography
List of Figures

3.1 Smoothing flux values on a node based patch ........... 15
3.2 Examples of element based and node based patches .......... 17
3.3 Overlapping patches give multiple node estimates ........... 17

5.1 Whiteman Square Boundary Conditions ................. 42
5.2 Whiteman Square Analytic Solution ................. 43
5.3 Initial Mesh h=0.5 ........................................... 44
5.4 Uniform Mesh Refinement of Full Model, T3 Element .......... 45
5.5 Log-Log Error Norm Plots for T3 Elements .......... 46
5.6 Initial Mesh h=0.5 ........................................... 47
5.7 Mesh Full Model, Q4 Element .................................. 48
5.8 Log-Log Error Norm Plots for Q4 Elements ........ 49
5.9 Initial Mesh h=0.5 ........................................... 50
5.10 Mesh Full Model, T6 Element .................................. 51
5.11 Log-Log Error Norm Plots for T6 Elements .......... 51
5.12 Mesh Quarter Model, T3 Element ......................... 53
5.13 Log-Log Error Norm Plots for T3 Elements of 1/4 Model .... 53
5.14 Mesh Quarter Model, Q4 Element ......................... 54
5.15 Log-Log Error Norm Plots for Q4 Elements of 1/4 Model .... 55
5.16 Mesh Quarter Model, T6 Element ......................... 56
5.17 Log-Log Error Norm Plots for T6 Elements of 1/4 Model .......... 57
5.18 Mesh Eighth Model, T3 Element .................................. 59
5.19 Log-Log Error Norm Plots for T3 Elements of 1/8 Model ........ 59
5.20 Mesh Eighth Model, Q4 Element ................................. 60
5.21 Log-Log Error Norm Plots for Q4 Elements of 1/8 Model ........ 61
5.22 Mesh Eighth Model, T6 Element ................................. 62
5.23 Log-Log Error Norm Plots for T6 Elements of 1/8 Model ........ 63
5.24 Comparison Plot of Elements Log-Log plot Full Model .......... 64
5.25 Comparison Plot of Elements Log-Log plot Quarter Model ...... 65
5.26 Comparison Plot of Elements Log-Log plot Eighth Model ....... 66
5.27 Comparison Plot of Boundary Conditions Log-Log plot with T3
   Elements .............................................................. 67
5.28 Comparison Plot of Boundary Conditions Log-Log plot with Q4
   Elements .............................................................. 68
5.29 Comparison Plot of Boundary Conditions Log-Log plot with T6
   Elements .............................................................. 69

6.1 Error plot .............................................................. 70
6.2 Initial H-Adaptive Mesh average h = 0.5 .......................... 71
6.3 H-Adaptive Refinement - Full Model .............................. 72
6.4 Final H-Adaptive Refined Mesh .................................... 72
6.5 Uniform Mesh Refinement .......................................... 74
6.6 H-Adaptive Mesh Refinement ....................................... 74
6.7 Comparison of Uniform Mesh and H-Adaptive Mesh for Full Model . 75
6.8 Initial Mesh of One-Eighth Model ................................. 76
6.9 H-Adaptive Mesh Refinement of One-Eighth Model ............... 77
6.10 Final H-Refinement Mesh of One-Eighth Model .................................................. 77
6.11 Uniform Mesh Refinement of One-Eighth Model .................................................. 78
6.12 H-Adaptive Mesh Refinement of One-Eighth Model ............................................ 78
6.13 Comparison of Uniform Refinement to H-Adaptive Refinement of
One-Eighth Model ........................................................................................................... 79
Chapter 1

Introduction

1.1 History

In the late 1940s, the finite element method was a numerical analysis technique which started as a structural analysis tool. As the power of computers increased the finite element method has grown into a widely used engineering tool. Today the finite element method is used in many engineering disciplines including aerospace, automotive, biomedical, electrical, hydraulic and nuclear engineering. It is used to analyze many problems from classical static structural problems to heat transfer and dynamics. The finite element method is a numerical technique for obtaining an approximate solution of partial differential equations with appropriate boundary conditions. These differential equations are then converted to a set of linear algebraic equations. The finite element method is important, since these equations can be assembled and solved on a computer simply.

1.1.1 Error Estimation

Error estimators provide information about the global quality of the solution and the distribution of error in the domain. The finite element method takes complex or difficult engineering problem and divides the problem into smaller elements (finite elements). A simple approximation is made using the smaller elements (local). The local approximate solutions are then used to obtain a global approximation of the solution. As the number of elements is increased, the finite element solution converges to the true solution. The solution may get better, however the cost of the solution
increases. In order to reduce the cost and increase the accuracy of the solution, error estimation is important. Effective error estimation can reveal where the elements need refinement therefore reducing the cost.

1.2 Thesis Objective

There are a few practical methods for obtaining error estimates. One of the most popular methods is based on the use of a "super-convergent patch". The purpose of this thesis is to compare the error norms for gradients and second derivative of three different choices for the super-convergent patch. The super-convergent patch choices compared are:

- Element Based Patch - All Neighbors
- Element Based Patch - Facing Neighbors
- Nodal Based Patch - All Element Neighbors

To witness the behavior of second derivatives recovered from the super-convergent patch, different models and elements are evaluated.

1. Models

   (a) Lakhany-Whiteman Square (3 different boundary conditions)
      i. Full Model - 4 essential boundary conditions
      ii. Quarter Model - 2 essential and 2 natural boundary conditions
      iii. Eighth Model - 3 essential and 1 natural boundary conditions

2. Elements

   (a) Linear triangular T3
(b) Bilinear quadrilateral Q4

(c) Quadratic triangle T6

The models are also evaluated for different mesh refinements.

- Uniform mesh refinement

- H-Adaptive mesh refinement

At the conclusion of the thesis, the trade-off of using different models, elements, and super-convergent patches will be determined. These trade-offs will allow the user to determine the optimal evaluation method based on their time and restraints.
Chapter 2

Literature Survey

2.1 Error Estimation

2.1.1 Kelly

In the past, it was possible to determine the rate of convergence of a finite element analysis by continuing to uniformly refine the mesh. Since this approach requires multiple analyses, the error analysis is computationally expensive. Kelly [12] discusses the theory and methods of deriving error estimates for second-order problems. The study estimates the error and shows where degrees of freedom should be added to improve the finite element mesh. Posteriori error estimates involve local computations instead of global ones. Kelly derived the error estimates for one- and two-dimensional problems, for higher order elements, and for the p-version of the finite element method in which the order of the elements is increased on a fixed mesh. The work also discusses how the error can be reduced to an acceptable level by identifying error indication and error estimation. The magnitude of the error is measured using the energy norm. This effectivity index is the ratio of the predicted energy norm of the error to the exact value of this norm.

2.1.2 Zienkiewicz and Zhu

Zienkiewicz and Zhu [36] gave numerical evidence of the effectiveness of a-posteriori error estimator for finite element analysis. The Zienkiewicz and Zhu estimator is popular, since it is easy to incorporate into existing finite element codes. By analyzing wider classes of estimators, it is shown that some estimators are asymptotically exact.
Zienkiewicz and Zhu analyzed three numerical examples to show the reliability of the error estimators. The effectivity index and the relative error where calculated during the evaluation of the examples. Here the effectivity index is the ratio of the estimated to the true error. Their examples show that the effectivity index tends towards unity, as the mesh is refined. The error estimator is asymptotically exact, since the effectivity index converges to unity. The results show that the error can be effectively and economically estimated by the projection technique.

2.2 Patch Recovery

2.2.1 Zienkiewicz and Zhu

Zienkiewicz and Zhu [36, 37] developed an effective way of getting a posteriori error estimate for a finite element analysis. In their procedure, the derivative quantities are improved using a suitable smoothing procedure. The error is then approximated as the difference between the smoothed and the original derivatives. The error estimation is dependent on the efficiency and accuracy of the smoothing technique, therefore the rate of convergence of the smoothed derivatives should be higher than that of the classical finite element derivatives. Zienkiewicz and Zhu developed the patch recovery procedure as a derivative smoothing procedure.

2.2.2 Aalto

Aalto [14] proposed a patch recovery procedure which improved on the Blacker and Belytchko[7] procedure that used information from residual error in the field equations to improve the accuracy of a patch recovery procedure. Aalto reduced the number of local equations that are solved within each patch using a number of local equations that are solved within each patch using a local least square fitting technique.
2.3 First Derivative

2.3.1 Zienkiewicz and Zhu

Zienkiewicz and Zhu presented a local recovery technique to recover the derivatives of the finite element solution. The recovered solutions are used to develop the post-processing type of energy norm and maximum norm error estimators. Numerical examples were used to demonstrate the superconvergence of the recovered derivative of the finite element solution and the effectivity and convergence of the proposed error estimators.

2.3.2 Wilberg and Wahab

Wilberg and Wahab [24] proposed a technique for determining first order derivates at nodal points based on derivatives in superconvergent points. Wilberg and Wahab claim that this technique improves on the superconvergent patch recovery technique developed by Zienkiewicz and Zhu. An improvement in the recovered derivatives is obtained using a procedure based on a least-squares fit of a higher-order stress distribution governed by the stresses at the superconvergent points obtained from the finite element calculation. The study concluded that the new technique using one order higher polynomial resulted in an increase in accuracy when compared to the Zienkiewicz and Zhu method.

2.4 Second Derivative

2.4.1 Lakhany and Whiteman

Lakhany and Whiteman [23] computed the second order derivatives locally over each element using the midpoint recovery technique. The gradient of the finite element solution is recovered then the recovery techniques are repeated to obtain the super-convergent second order derivatives. These derivatives are approximations to the
corresponding second derivatives of the weak solution when sufficient regularity of the weak solution is assumed. Their mathematical procedure required the use of a mesh of uniform right triangle linear elements and thus is not practical for most engineering shapes. The resulting second derivative estimates are quite similar to a classical finite difference stencil for computing second derivatives. We will use the same test geometry, but our procedure is not limited to linear triangles or uniform meshes.
Chapter 3

Derivative Recovery and Error Estimation for Elliptic Problems

3.1 Introduction

Having obtained a finite element solution, we would like to be able to estimate the error in that solution and, perhaps, have the analysis program correct itself. Currently, that is a practical option for elliptic partial differential equations. Here we will outline the basic method and notation of that class of error estimation. Consider a problem posed by the PDE written as

\[ L \phi + Q = 0 \quad \text{in } \Omega \]  \hspace{1cm} (3.1)

with the essential boundary condition \( \phi = \phi_o \) on boundary \( \Gamma_o \), and a prescribed traction \( t = t_o \) on the boundary \( \Gamma_t \) with \( \Gamma = \Gamma_o \cup \Gamma_t \). Here \( L \) is a linear differential operator that can be written in the symmetric form

\[ L \equiv S^T E S \] \hspace{1cm} (3.2)

where \( S \) is a lower order operator and \( E \) contains material information. The gradient quantities of interest are denoted as

\[ \epsilon \equiv S \phi \] \hspace{1cm} (3.3)

and the flux quantities, \( q \), by
\[ q = \pm E \epsilon. \quad (3.4) \]

On the boundary, \( \Gamma \), of \( \Omega \) we are often interested in a traction, \( t \), defined in terms of the fluxes by

\[ t = G \, q \quad (3.5) \]

where \( G \) is usually defined in terms of the components of the normal vector.

For example, in isotropic conduction \( \phi \) is the temperature, \( Q \), in internal volumetric heat source, \( E = k \, I \), where \( k \) is the thermal conductivity, and \( S \) is simply the gradient

\[ S = \nabla = \left\{ \begin{array}{c} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{array} \right\} \]

so that \( L \) becomes the Laplacian

\[ L = \nabla^T k \nabla = k \left( \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \right). \]

Here \( \epsilon \) is the gradient vector

\[ \epsilon = \nabla \phi, \quad \epsilon^T = \left[ \begin{array}{ccc} \frac{\partial \phi}{\partial x} & \frac{\partial \phi}{\partial y} & \frac{\partial \phi}{\partial z} \end{array} \right] \]

and the Fourier Law defines the heat flux vector

\[ q = -k \, I \nabla \phi = -k \, \nabla \phi, \quad q^T = [q_x \quad q_y \quad q_z]. \]

Likewise, for \( G = n \) the boundary traction is the normal heat flux:

\[ t = nq = q_x n_x + q_y n_y + q_z n_z = q_n = -k \frac{\partial \phi}{\partial n}. \]

For the one-dimensional case of heat conduction these all reduce to scalars with

\[ S = \partial/\partial x, \quad E = k, \quad \epsilon = \partial \phi/\partial x, \quad q = q_x = -k \partial \phi/\partial x. \]
and the governing differential equation $L\phi + Q = 0$ becomes

$$\frac{\partial}{\partial x} \left( k \frac{\partial \phi}{\partial x} \right) + Q = 0$$

in $\Omega$ with $\phi = \phi_0$ on $\Gamma_0$. While on the boundary $\Gamma_t$, the traction $t = q_n = -k n_x \partial \phi / \partial x$, and has an assigned value of $q_n = t_0$.

Likewise, for a problem in planar elasticity, $\phi$ and $\epsilon$ become the displacement components $\phi = [u \ v]^T$ and strain components $\epsilon = [\epsilon_x \ \epsilon_y \ \gamma]$, respectively, which are related by the differential operator

$$S = \begin{bmatrix}
\frac{\partial}{\partial x} & 0 \\
0 & \frac{\partial}{\partial y} \\
\frac{\partial}{\partial y} & 0
\end{bmatrix}.$$

The corresponding fluxes or stresses are $q = \sigma \equiv [\sigma_x \ \sigma_y \ \tau]^T$ which are related to the strains, $\epsilon$, by the symmetric "stress-strain" matrix, $E$ (without a minus sign). The source $Q$ generalizes to the body force vector $Q = X = [X_x \ X_y]^T$.

Finally, the surface traction vector $t = T = [T_x \ T_y]$ is related to the surface stresses, $\sigma$, and the components of the outward unit normal vector, $n$, by

$$G = \begin{bmatrix}
n_x & 0 & 0 \\
0 & n_y & 0 \\
0 & 0 & n_x
\end{bmatrix}.$$

In a finite element solution we seek an approximation $\hat{\phi}$ which, in turn, approximates the gradient and flux terms, $\hat{\epsilon}$ and $\hat{q}$. The standard interpolation gives

$$\phi \approx \hat{\phi} = N(x) \Phi^* \ x \ in \ \Omega^*$$  \hspace{1cm} (3.6)

with a corresponding gradient estimate

$$\epsilon \approx \hat{\epsilon} = S N(x) \Phi^* \equiv B^*(x) \Phi^*$$  \hspace{1cm} (3.7)

for $x$ in $\Omega^*$. Likewise, the flux approximation is $\sigma \approx \hat{\sigma} = E^* B^*(x) \Phi^*$. In this notation the element square matrix and source vector are
\[ K^e = \int_{\Omega^e} B^{eT} E^e B^e \, d\Omega \]  

(3.8)

and

\[ F_Q^e = \int_{\Omega^e} Q^e N^{eT} \, d\Omega \]  

(3.9)

and the boundary traction contribution, if any, is

\[ F_{q_b}^b = \int_{\Gamma_b} q_{bn}^b N^{bT} \, d\Gamma. \]  

(3.10)

When the element degrees of freedom subset, \( \Phi^e \subset \Phi \), have been computed, the local errors in an element domain are

\[ e_\varphi(x) \equiv \varphi(x) - \hat{\varphi}(x) \]  

(3.11)

\[ e_\varepsilon(x) \equiv \varepsilon(x) - \hat{\varepsilon}(x), \quad x \in \Omega^e \]  

(3.12)

\[ e_\sigma(x) \equiv \sigma(x) - \hat{\sigma}(x). \]  

(3.13)

These quantities can be either positive or negative so we will mainly be interested in their absolute value or some normalized measure of them. We will employ integral norms for our error measures. In finite elements we often use the inner product defined as

\[ <u, v> = \int_{\Omega} u(x) v(x) \, d\Omega \]  

(3.14)

which possesses a natural norm defined as

\[ ||\varphi||^2 = <\varphi, \varphi> = \int_{\Omega} \varphi(x) \varphi(x) \, d\Omega. \]  

(3.15)

This is also called the \( L_2 \) norm, since it involves the integral of the square of the argument. We wish to minimize the error in the solution, \( \varepsilon_\varphi \). However, for elliptical problems it can be shown that this corresponds to minimizing the error energy norm, or other related measures. Error estimates commonly employ one of the following norms:
1. The error energy norm \( \| e \| \) defined as

\[
\| e \| = \left[ \int_{\Omega} (e - \hat{e})^T E (e - \hat{e}) \, d\Omega \right]^{\frac{1}{2}}
\]

\[
= \left[ \int_{\Omega} (\varepsilon - \hat{\varepsilon})^T (\sigma - \hat{\sigma}) \, d\Omega \right]^{\frac{1}{2}} = \left[ \int_{\Omega} \varepsilon^T e_\sigma \, d\Omega \right]^{\frac{1}{2}}
\]  

(3.16)

\[
= \left[ \int_{\Omega} (\sigma - \hat{\sigma}) \, E^{-1}(\sigma - \hat{\sigma}) \, d\Omega \right]^{\frac{1}{2}}.
\]

2. The \( L_2 \) flux or stress error norm

\[
\| e_\sigma \|_{L_2} = \left[ \int_{\Omega} (\sigma - \hat{\sigma})^T (\sigma - \hat{\sigma}) \, d\Omega \right]^{\frac{1}{2}} = \left[ \int_{\Omega} e_\sigma^T e_\sigma \, d\Omega \right]^{\frac{1}{2}}.
\]

(3.17)

3. The root mean square stress error, \( \Delta \sigma \), given by

\[
\Delta \sigma = \| e_\sigma \|_{L_2} / \Omega^{\frac{1}{2}}.
\]

(3.18)

4. In general, any of these norms is the sum of the corresponding individual element norms:

\[
\| \phi \|^2 = \sum_{\varepsilon} \| \phi \|^2_{\varepsilon}
\]

(3.19)

where

\[
\| \phi \|^2_{\varepsilon} = \int_{\Omega^\varepsilon} \phi^2 \, d\Omega
\]

(3.20)

and the domain is the union of all of the element domains, \( \Omega = \bigcup \Omega^\varepsilon \).

A relative percentage error can be defined as \( \eta = \| e \| / \| \phi \| \times 100 \) which represents a weighted root mean square percentage error in the stresses. We can compute a similar estimate relative to the \( L_2 \) norms. In most of the literature on the subject of error estimators there is a discussion of the effectivity index, \( \Theta \). It is simply the ratio of the estimated error divided by the exact error. Usually an analytical solution
is employed to compute the exact error (and to assign the problem source, \( Q \), and boundary conditions, \( \phi_0 \)), but sometimes very high precision numerical results are used. Clearly, one should search for methods where the effectivity index is very close to unity. Some methods employ a constant, determined by numerical experiment, to increase their effectivity index to near unity for a specific element type.

### 3.2 Error Estimates

In general, we do not know the exact strain, \( \epsilon \), or stress values, \( \sigma \), in Eqns. 3.3 and 3.4. We do have piecewise continuous estimates for the element strains, \( \hat{\epsilon} \), and stresses, \( \hat{\sigma} \). Unlike the solution, \( \phi \), these estimates are generally discontinuous between elements. For homogeneous domains (homogeneous \( E \)), we expect the exact \( \epsilon \) and \( \sigma \) to be continuous. At the interface of two different homogeneous materials (\( E_1 \) and \( E_2 \)), we expect the gradients, \( \epsilon \), to be discontinuous and the fluxes, \( \sigma \), to be at least partially continuous. In most elliptical problems, we expect the normal flux component to be continuous, but the tangential component along the interface may be discontinuous. In a homogeneous domain a continuous estimate of \( \epsilon \) and \( \sigma \) should be more accurate than would be the piecewise continuous \( \hat{\epsilon} \) and \( \hat{\sigma} \). Denote such continuous approximations by \( \epsilon^* \) and \( \sigma^* \), respectively. That is, \( \hat{\sigma} \) is discontinuous across element boundaries, while the \( \sigma^* \) are constructed to be continuous across those boundaries. Then, within an element, the error estimators with good accuracy are

\[
e_\epsilon \approx \epsilon^*(x) - \hat{\epsilon}(x) \quad e_\sigma \approx \sigma^*(x) - \hat{\sigma}(x) .
\]

(3.21)

There are various procedures for obtaining nodal values of the strains, \( \epsilon^* \), or stresses, \( \sigma^* \), that will yield a continuous solution over the domain. Probably the most common is simply an averaging based on the number and/or size of elements contributing to a node. The continuous nodal stresses are obtained by averaging
the values from surrounding elements. However, this simple averaging process does not have any mathematical foundation relative to the original problem and can not be used as part of an effective error stimator. A precise mathematical procedure for computing the nodal values directly was given early in the development of finite element methods by Oden, et al [17, 10]. However, that "Conjugate Stress" approach required the assembly of element contributions and solving a system of equations equal in size to the number of nodes in the system. More recently for elliptical problems it has been shown that a Super-Convergent Patch (SCP) of elements provides a way to recover accurate continuous nodal fluxes or nodal gradients that can be used in an error estimator.

3.3 Super-Convergent Patch Recovery

Zienkiewicz and Zhu [36, 37, 40] developed the concept of utilizing a local patch of elements, sampled at their super-convergent points, to yield a smooth set of least square fit nodal gradients or fluxes. As noted earlier, the super-convergent points of an element are the special interior locations where the gradients of the element are most accurate. That is, those gradient locations match those of polynomials of one or more degrees higher. Numerous minor improvements to their original process have shown the SCP recovery process to be a practical way to get continuous nodal fluxes, $\sigma^*$. They have demonstrated numerically that one can generate superconvergence estimates for $\sigma^*$ at a node by employing patches of elements surrounding the node. These concepts are illustrated in Fig.3.1.

A local least squares fit is generated over the patch of elements in the following way. Assume a polynomial approximation of the form

$$\sigma^* = P(\xi, \eta) \ a$$  \hspace{1cm} (3.22)
Interpolated Solution: $u_h = N(x) U$ at node points, •
Element Flux: $q_h = E B(x) U$ at Gauss points, *
Least Squares Patch Fit of Flux, $F_p$, at patch points, o
Interpolated Node Flux in Patch: $q_p = N(x) F_p$ at nodes in patch, •
Element Flux Error Estimate: $e_q = q_p - q_h$

Figure 3.1: Smoothing flux values on a node based patch
where \( \mathbf{P} \) denotes a polynomial (in a local parametric coordinate system selected for each patch) that is of the same degree and completeness that was used to approximate the original solution, \( \mathbf{u}_h \). That is, \( \mathbf{P} \) is similar or identical to \( \mathbf{H} \). Here \( \mathbf{a} \) represents nodal values of the flux. Recall that \( \hat{\sigma} \) was computed using the derivatives of \( \mathbf{H} \). To compute the estimate for \( \sigma^e \) at the nodes inside the patch, we minimize the function

\[
F(\mathbf{a}) = \sum_{j=1}^{n} \left( \sigma^e_j - \hat{\sigma}_j \right)^2 \rightarrow \min
\]

where \( n \) is the total number of integration points (or super-convergent points) used in the elements that define the patch and \( \sigma_j \) is the flux evaluated at point \( \mathbf{x}_j \). Substituting the two different interpolation functions gives

\[
F(\mathbf{a}) = \sum_{e=1}^{NPE} \sum_{j=1}^{QE} \left[ \mathbf{P}_j \mathbf{a} - \mathbf{E}^e \mathbf{B}_j^e \mathbf{u}^e \right]^2
\]

where \( NPE \) denotes the number of elements in the patch and \( QE \) is the number of integration points used to form \( \hat{\sigma}^e \). That is, we are seeking a least squares fit through the

\[
n = \sum_{e=1}^{NPE} \sum_{j=1}^{QE}
\]

data points to compute the unknown coefficients, \( \mathbf{a} \), which is a rectangular matrix of flux components at each node of the patch. The standard least squares minimization gives the local algebraic problem \( \mathbf{S} \mathbf{a} = \mathbf{C} \) where

\[
\mathbf{S} = \sum_{e=1}^{NPE} \sum_{j=1}^{QE} \mathbf{P}_j^T(\xi_j, \eta_j) \mathbf{P}(\xi_j, \eta_j), \quad \mathbf{C} = \sum_{e=1}^{NPE} \sum_{j=1}^{QE} \mathbf{P}_j^T \mathbf{E}^e \mathbf{B}_j^e \mathbf{U}^e.
\]

This is solved for the coefficients \( \mathbf{a} \) of the local patch fit. It is the cost of solving this small system of equations, for each patch, that we must pay in order to obtain the continuous nodal values for the fluxes. To avoid ill-conditioning common to least squares, the local patch fitting parametric space \( (\xi, \eta) \) is mapped to enclose the patch of elements while using a constant Jacobian for the patch. The use of the constant
Jacobian is the key to the efficient conversion of the physical stress location, $x_j$, to the corresponding patch location, $\xi_j$. Here the implementation actually employs a diagonal constant Jacobian to map the patch onto the physical domain.

![Diagram](image)

**Figure 3.2:** Examples of element based and node based patches

![Diagram](image)

**Figure 3.3:** Overlapping patches give multiple node estimates

Zhu [32] has verified numerically that the derivatives estimated in this way have an accuracy of at least order $O(h^{p+1})$, where $h$ is the size of the element and $p$ is
the degree of the interpolation, $N$, used for the solution. There is a theorem that states that if the $\sigma^*$ are super convergent of order $O(h^{p+\alpha})$ for $\alpha > 0$, then the error estimator will be asymptotically exact. That is, the effectivity index will approach unity as $\theta \to 1$. This means that we have the ability to accurately estimate the error and, thus, to get the maximum accuracy for a given number of degrees of freedom. There is not yet a theoretical explanation for the "hyperconvergent" convergence (two orders higher) reported in some of the SCP numerical studies. It may be because the least square fit does not go exactly through the given Barlow points. Thus, they are really sampling nearby. It has been shown that derivative sampling points for a cubic are at $\pm 0.577$, while those for the quartic are at $\pm 0.707$. Therefore, the patch least squares smoothing may effectively be picking up those quartic derivative estimates and jumping to a higher degree of precision. It is also possible to make other logical choices for selecting the elements that will constitute a patch. Figure 3.2 shows two types of element based patches as well as the above node based patch. The implementation of the SCP recovery method will be given after considering other error indicator techniques.

### 3.4 Hierarchical Error Indicator

Zienkiewicz and Morgan [35] have given a detailed study of how hierarchical interpolation functions can be employed to compute an error estimate. Here we will outline this approach in one-dimension. They define the error norm as

$$ ||e||_E^2 = -\int_\Omega er d\Omega $$

where the error is $e = \phi - \hat{\phi}$ and $r$ is the residual error on the interior of the domain

$$ L\hat{\phi} + q = r \neq 0. $$
Now we enrich the current approximate solution \( \hat{\phi} \) to get a more accurate (higher degree) approximation by adding the next hierarchical bubble function \( \phi^* = \hat{\phi} + H_b a_b \) where \( a_b \) is the next unknown hierarchical degree of freedom. If we take this as representing the correction solution (\( \phi \approx \phi^* \)), then we have \( e^e = H_b^e a_b \) and

\[
||e^e||_E = a_b \int_{\Omega^e} H_b^{eT} r^e \, d\Omega.
\]

If one can estimate the degree of freedom \( a_b \), then we have an error indicator. If it is the only new dof, and if the hierarchical functions are orthogonal, the new system equations are

\[
\begin{bmatrix}
S & 0 \\
0 & s_{bb}
\end{bmatrix}
\begin{bmatrix}
a \\
a_b
\end{bmatrix} =
\begin{bmatrix}
C \\
c_b
\end{bmatrix}
\]

where \( S \) and \( C \) were the previous system matrices, and \( s_{bb} \) and \( c_b \) are the new element (and system) stiffness and source terms, respectively. From this diagonal system, we compute the new term \( a_b = c_b / s_{bb} \), that is,

\[
c_b = \int_{\Omega^e} H_b^T q^e \, d\Omega
\]

or from the internal residual definition and the above orthogonality,

\[
c_b = \int_{\Omega^e} H_b^T (r - L\hat{\phi}) \, d\Omega = \int_{\Omega^e} H_b^T r^e \, d\Omega.
\]

Therefore, this error indicator simplifies to \( ||e^e||_E = a_b c_b = \frac{c_b}{s_{bb}} \).

In the following we will use this approach on a one-dimensional sample problem. We will see that the effectivity index is only about one-half, which is unacceptably far from the desired value of unity. While we could introduce a "fudge factor" constant of two, it is wiser to search for a method, like the SCP recovery, that would yield an effectivity index that is always much closer to unity. Consider the Zienkiewicz and
Morgan (Z-M) hierarchical error estimator for their Example 8.1 of [35] expanded to consider the local element errors and flux balances. The model problem is

\[-\frac{d^2\phi}{dx^2} + Q = 0 \quad x \in ]0,L[ \quad \phi(0) = 0, \quad \phi(L) = 0\]

with the exact solution, for constant $Q$, of $\phi = Q(x - L)x/2$, so $\phi' = Q(2x - L)/2$.

Using the Galerkin approximation:

\[\int_0^L \phi Q \, dx - \int_0^L \phi \phi_{xx} \, dx = \int_0^L \phi Q \, dx - \phi_0 \phi_x \bigg|_0^L + \int_0^L \phi_{xx} \, dx = 0\]

or finally

\[\int_0^L \phi_{xx} \, dx = -\int_0^L \phi Q \, dx + \phi_0 \phi_x \bigg|_0^L\]

Splitting the domain into elements and using our interpolations $\phi_h = H^e u^e$ this reduces to the matrix form:

\[\sum_e u^e^T K^e u^e = -\sum_e u^e^T F_Q^e + u(L) \phi_x(L) - u(0)\phi_x(0)\]

with the typical element matrices defined (with $E = I$) as

\[K^e = \int_{L^e} H^e_{xx} H^e_{xx} \, dx, \quad F_Q^e = \int_{L^e} H^e_{xx} Q^e \, dx.\]

Recall for an initial linear interpolation with constant coefficients

\[K^e = \frac{1}{L^e} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad F_Q^e = \frac{Q^e L^e}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}\]

First, consider a trivial single element solution. By inspection, $L^e = L$ so that

\[
\frac{1}{L^e} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \frac{Q^e L^e}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} -\phi_x(0) \\ +\phi_x(L) \end{bmatrix}
\]

but $u_1 = u_2 = 0$ from the boundary conditions. There are no unknown degrees of freedom to compute so we go directly to the flux recovery and error estimates. Solving for the flux gives $\phi_x(0) = -Q L^e/2$ and $\phi_x(L) = Q L^e/2$ as the two necessary nodal
flux values. Checking we see that a useless solution has still give nodal fluxes that are exact as \( L^e = L \). The recovered nodal flux resultants are exact despite the fact that the single element solution is trivial, i.e., \( \phi_h = H^e \ u^e = H^e \ O^e = 0 \) (which is exact at nodes). The single element solution is useless in estimating the solution value. In the energy norm the error measure is

\[
||e||^2 = -\int_L (\phi - \phi_h) (-\phi_{h,x} + Q_h) \ dx = -\int_L e \cdot r \ dx
\]

where \( r \) is the interior residual. For the single element case \( \phi_h = 0 \) and \( \phi_{h,x} = 0 \) so \( r = Q \) and \( e = \phi \). Thus,

\[
||e||^2_{\text{exact}} = -\int_L Q^2(x - L) \ x/2 \ dx = -\frac{Q^2}{2} \int (x^2 - Lx) \ dx = \frac{Q^2 L^3}{12}.
\]

To compute an error indicator, we add a quadratic hierarchical term to the linear element so \( \phi_h^* = \phi_h + u^e_5 \ H^e_3 \) where \( H^e_3(x) = x(L-x) \) in global space, or \( H^e_3(r) = r(1-r) \) in a local unit coordinate space. The Z-M error indicator is

\[
||e^e||^2 = \left[ \int_{L^e} H^e_3 r \ dx \right]^{2} / K^e_{33}, \quad K^e_{33} = \int_{L^e} H^e_3 \left[ -H^e_3'' \right] \ dx
\]

is the new hierarchical stiffness term, and \( e^e = \phi^*_h - \phi_h \). Here

\[
K^e_{33} = \int_{L^e} r \ (1-r) \left[ -\frac{1}{L^2} (-2) \right] \ dx = \frac{1}{3L^e}
\]

and

\[
I^e = \int_{L^e} H^e_5 R^e \ dx = \int_{L^e} r \ (1-r) \ Q^e \ dx = Q^e L^e/6
\]

so

\[
||e^e||^2 = \frac{[Q^e L^e/6]^2}{(1/3L^e)} = \frac{3Q^e^2 L^e^3}{36} = \frac{Q^e^2 L^e^3}{12}
\]

which happens to be exact for one element. Thus, we see we can get error estimates for every crude finite element models.
3.5 Flux Balancing Error Estimates

Recently, Ainsworth and Oden [1, 2] have developed a local patch error estimator that is very well justified through detailed functional analysis, is robust, easy and economical to implement, and gives very accurate local error estimates for any order interpolation functions. That is, it usually produces an effectivity index that is very close to unity and is much more reliable than other methods known to the author. By using a dual variational formulation, they have proved that this estimator provides an upper bound estimate of the true error. The Ainsworth-Oden flux balancing method uses a local patch of elements for each master node. A typical patch includes all elements connected, or constrained, to the master node. The goal is to choose a linear averaging function $\alpha_{KL}$ between each pair of adjacent elements, $K$ and $L$, such that the residual internal error, $r$, and inter-element gradient jumps, $R$, are in equilibrium; that is,

$$\int_{\Omega} r \, d\Omega + \int_{\Gamma} R \, d\Gamma = 0.$$ 

They provide a detailed procedure for implementing this method, including pseudocode for the flux-splitting algorithm. The equilibrium fluxes are used to compute the local error estimator.

Once the fluxes are in equilibrium, the error, $e = u - \hat{u}$, is bounded above by the norm

$$\| e \|^2 \leq \frac{1}{\beta^2} \sum_{e=1}^{NE} \| \phi^e \|^2$$

where $\beta > 0$ is a constant depending on the norm selected ($\beta = 1$ for the standard energy norm), and $\phi$ is obtained by solving the element local Neumann problem

$$a^e(\phi, w) = L^e(w) - B^e(\hat{u}, w) +$$

$$+ \int_{\Gamma_e} w n^e \cdot \left[(1 - \alpha_{KL}(s)) \, k^e \nabla \hat{u} \big|_{\Gamma_e} + \alpha_{KL}(s) \, k^f \nabla \hat{u} \big|_{\Gamma_f} \right] \, d\Gamma.$$
The examples by Ainsworth and Oden show this procedure to be accurate and economical. The effectivity index, $\Theta$, is usually very near unity as desired, and is usually above 0.9 for even crude initial mesh calculations. While this is also a recommended method, we choose to implement SCP recovery due to its simplicity.
Chapter 4

Super-Convergent Patch Recovery

4.1 Patch Implementation

Since the super-convergent patch (SCP) recovery method is relatively easy to understand and is accurate for a wide range of problems, it was selected for implementation in the educational program MODEL. Its implementation is designed for use with most of the numerically integrated 1-D, 2-D, and 3-D elements in the MODEL library. Most of the literature on the SCP recovery methods are limited to a single element type and a single patch type. The present version is somewhat more general in allowing a mixture of element shapes in the mesh and a mesh that is either linear, quadratic, or cubic in its polynomial degree.

This represents actually the third version of the SCP algorithm and is a simplification of the first two. The method was originally developed for a $p$-adaptive code where all the elements could have a different polynomial degree on each element edge [16]. That version was then extended to an object-oriented F90 $p$-adaptive program that also included equilibrium error contributions as suggested by Wiberg [25] and others [21]. Including the $p$-adaptive features and the object-oriented features made the data base more complicated and required more planning and programming than desired. However, the version given here has shown to be robust and useful.

The SCP recovery process is clearly heuristic in nature, so some arbitrary choices need to be made in the implementation. We begin by defining a "patch" to be a local group of elements surrounding at least one interior node or being adjacent
to a boundary node. The original research in SCP recovery methods used patches sequentially built around each node in the mesh. Later it was widely recognized that one could use a patch for every element in the mesh. Therefore, three types of patches will be defined here:

1. Node-based patch: An adjacent group of elements associated with a particular node.

2. Element-based patch: All elements adjacent to a particular element.

3. Face-based patch: This subset of the element-based patch includes only the adjacent elements that share a common face with the selected element. For two-dimensional elements this means that they share a common edge.

Any of these three definitions of a patch requires that one have a mesh "neighbors list". That is, we will need a list of elements adjacent to each node, or a list of elements adjacent to each element, or the subset list of facing element neighbors. These can be expensive lists to create, but are often needed for other purposes and are sometime supplied by a mesh generation code or an equation re-ordering program. Here several routines are included for creating the lists and printing them. Normally, since those items are of unknown variable lengths, they would be stored in a linked list data structure. Here, for simplicity, they have been placed in rectangular arrays and padded with trailing zeros. This wastes a little storage space but keeps the general code simpler. The primary routines for the node-based patches are the routines COUNT.L.AT.NODE and FORM.L.ADJACENT.NODE, while the element-based patches use the routines COUNT.ELEMS.AT.ELEM and FORM.ELEMS.AT.EL.

In MODEL the default is to use an element-based patch. Having selected a patch type, we should now give consideration to the kind of data that will be needed for the SCP recovery. There are two main segments in the process:
1. Averaging the patch and system nodal fluxes.

2. Using the system nodal fluxes in the calculation of an error estimate.

The whole basis of the SCP recovery is that there are special locations within an element where we can show that the derivatives are most accurate or exact for a given polynomial degree. We refer to such locations as element super-convergent points. They are sometimes called Barlow points. The derivation of the locations generally shows them to coincide with the Gaussian quadrature points. Here we will assume that the minimum number of quadrature points needed to properly form the element matrices have locations that correspond to the element super-convergent points, or are reasonably close to them. Thus, as we process each element to build its square matrix, we will want to save, at each quadrature point, its physical location in space and the differential operator matrix, \( B \), that will allow the accurate gradients to be computed from the local nodal solution. Looking ahead to the error estimation or other post-processing, we know that at times we will also want to have the constitutive matrix, \( E \), so we will also save it. Note that we are allowing for different, but compatible, element shapes in the mesh (and patches) and they would require different integration rules.

Now we should look ahead to how the above data are to be recovered in the SCP section of the code. The main observation is that, for an unstructured mesh, the element numbers for the elements adjacent to a particular node or element are unpredictable. While we have a straight-forward way to save the above data in a sequential fashion, we need to recover the element data in a random fashion. Thus, we either need to build a database that allows random access or we must decide to re-compute the data in each element of each patch. The latter is too expensive, so we must select the new database option. We could select linked lists, or a tree structure, but there is a simpler way. Fortran has always had a feature known as
a "direct access" file that allows the user to randomly recover or change data. The actual data structure employed is left up to the group that writes the compiler, and is mainly hidden from the user. However, the user must declare the "record number" of the data set to be recovered or changed. Likewise, the record number of each data set must be given as the data are saved to the random access file. This means that some logical way will be needed to create a unique number for each record at any quadrature point in the mesh.

For a mesh with a single element type and a single integration rule, we could write a simple equation for the record number. Here we are allowing a mixture of element types and quadrature rules, so we store the record number at each quadrature point in an integer array sized for the maximum number of elements and the maximum number of quadrature points per element. The record numbers are created sequentially as the element matrices are integrated. A function, SCP_RECORD_NUMBER, is supplied for randomly recovering the record number at any integration point in any element. Like any other file used in a program, a random access file must be opened. It is opened as a DIRECT access file of UNFORMATTED, or binary, records to minimize storage. We must also declare the length of the data records. It is actually hardware-dependent, so F90 includes an intrinsic function, INQUIRE (IOLENLENGTH), that will compute the record length given a list of variables and arrays to be included in each record. The unit number assigned to the random access file holding the SCP records is given the variable name U_SCPR.

As mentioned above the SCP process can be used to determine the average nodal fluxes and to use them to compute the element error estimates. The general outline of the process is as follows:

1. Preliminary

   (a) Build a list of element neighbors.
(b) Open the sequential file unit U.FLUX to receive element data related to flux calculations. Those data can also be used for optional post-processing.

(c) Compute the record length necessary to store the coordinates and flux components at a point.

(d) Open the random access file unit U.SCPR that will receive the quadrature point coordinates and flux components.

2. Element Matrices Generation Loop

(a) For each element save its number of integration points to file unit U.FLUX.

(b) Within the numerical integration loop of the element save the arrays XYZ, E, and B at each point so that the gradients and flux components can be found at the point.

(c) When all elements have been processed rewind the file U.FLUX to its beginning.

3. Flux Calculations and Saving Them for Averaging

After the solution has been obtained it is possible to compute the flux (and gradient) components within each element so that they can be smoothed to nodal values. The element flux calculation is done in POST_PROCESS_GRADS. First, the SCP record number is set to zero. Next, each element is processed in a loop:

(a) Recover the element type;

(b) Extract the nodal degrees of freedom of the element;

(c) Read the number of quadrature points in the element from U.FLUX;
(d) Quadrature Point Loop For each integration point of the element of the element, in routine LIST_ELEM_FLUXES, recover the XYZ, E, and B arrays from U_FLUX. Multiply B by the element dof to get the gradients or strains at the point, and then multiply those by the constitutive array, E, to get the fluxes, or stresses at the point. The element and quadrature point numbers are then printed along with their coordinates and flux, or stress, components. Lastly, the SCP database is updated by incrementing the record number by one, and then writing the coordinates and flux component arrays to the random access file, U_SCPR, as that record is to be later recovered in the patch smoothing process.

4.2 SCP Nodal Flux Averaging

Having developed the above database on unit U_SCPR we can now average the flux components at every node in each patch and then average them for each node in the mesh. Here we assume an element based patch system for calculating the averages. In subroutine CALC_SCL_AVE_NODE_FLUXES we loop over every element and carry out the least squares fit in its associated patch. Looking ahead to that process recall that we must select a polynomial, P, to be used in the patch. We must make a choice for that function. We might select a complete polynomial of a given degree, or a Serendipity polynomial of a given edge degree, etc. In the current implementation the default is to select that polynomial to be exactly the same as the polynomial used to interpolate the element for which the patch is being constructed. This means that we will select a constant Jacobian patch "element" that has its local axes parallel to the global axes and completely surrounds the standard elements that make up the patch. This is easily done by searching for the maximum and minimum components
of all of the element nodes in the patch. It would be easy to allow the user to select a patch type and degree through a keyword control input.

The least squares flux averaging process is:

1. Zero the nodal flux array and the counter for each node.

2. Loop over each element in the mesh:

   (a) Extract its element neighbors to define the patch

   (b) Find the spatial "box" that bounds the patch

   (c) Find the number of quadrature points in the patch (i.e. sum the count in each element of the patch).

   (d) Determine the element type and thus the patch "element" shape (line, triangle, hexahedron, etc.) and the patch polynomial degree.

   (e) Allocate storage for the least squares fit arrays.

   (f) Set the fit matrix row number to zero.

   (g) For each element in the patch loop over the following steps:

      i. Find its type and quadrature rule

      ii. Loop over each of its quadrature points

         A. Increment the row number by one.

         B. Use the SCP.RECORD.NUMBER function to recover the record number for that point.

         C. Read the physical coordinates and flux components from random access file U.SCPR by using that record number.

         D. Use the constant Jacobian of the patch to convert the physical location to the corresponding non-dimensional coordinates in the
patch. Note that this helps reduce the numerical ill-conditioning that is common in a least squares fit process.

E. Evaluate the patch interpolation polynomial at the local point (by utilizing the standard element interpolation library). Insert it into the left hand side of this row of the coefficient matrix.

F. Substitute the flux components into the right hand side data matrix, in the same row. Of course the number of columns on the right hand side is the same as the number of flux components. This is also the size of the patch result matrix, $\mathbf{a}$, to be computed.

G. Having completed the loop over all the elements in this patch we now have the rectangular arrays cited in Eq. (3.23) but we have not computed their actual matrix roducts as shown in Eq. (3.24). While that equation is the standard way to describe a least squares fit we do not actually use that process. Instead, we try to avoid possible numerical ill-conditioning by using an equivalent but more powerful process called the singular value decomposition solution (in subroutine SVDC\_FACTOR) to recover the local nodal flux values (with subroutine SVDC\_BACK\_SUBST) for the current patch nodes. However, we want smooth flux values at the actual nodes of the elements, not values at the patch nodes. Thus, for the elements in question we need to interpolate the patch results to the system nodes in the current patch.

H. Loop over nodes in this patch:
   - Use the constant patch Jacobian to convert the node coordinates to non-dimensional coordinates of the patch
   - Evaluate the patch interpolation matrix, $\mathbf{H}$.
• Compute the flux components at the node by the matrix product of $\mathbf{H}$ and $\mathbf{a}$.

• Increment the nodal counter for patch contributions by one and scatter the node flux components to the system flux nodal array.

iii. Optional Improvement of the Solution At this stage in the SCP process one can use the least square smoothed gradients in this patch to get a locally improved solution value estimate at all of the patch's interior nodes. However, one may want to just do so for the parent element.

The algorithm is a form of the Loubignac [22] iterative process:

A. Read or reform element matrices, $\mathbf{S}_e$ and $\mathbf{C}_e$, here;

B. Use a higher order quadrature rule to form the equilibrating vector

$$\mathbf{V}_e = \int_{\Omega_e} \mathbf{B}^* (\sigma^* - \hat{\sigma}) d\Omega;$$

C. Assemble the elements in the patch into a local linear system:

$$\mathbf{S}^* \phi_{\text{new}} = \mathbf{C} - \mathbf{V};$$

D. Apply the previous solution as essential boundary conditions at all nodes on the patch boundary;

E. Solve for the new interior node values (always a small system but especially small for a patch of elements around a single node as in original ZZ patch paper). Call them $\phi_{\text{c}}^*$.

F. Compute norm of $\phi_{\text{a}} - \phi_{\text{c}}^*$ to use as an additional term in the final error estimator.

3. Final nodal flux average. Having processed every patch for the mesh each node has now received as many nodal flux estimates as there were patches that
contained that node. We finalize the nodal flux values by simply dividing each node's flux components sums by that integer counter, print the result, and save them in array SCP_AVERAGES for use by the element error estimator or other user-defined post-processing.

4.3 Computing the SCP Element Error Estimates

For a homogeneous domain, or sub-domain, the above nodal averaging process provides a continuous flux approximation that should be much closer to the true solution than the element discontinuous fluxes. Thus, it is reasonable to base the element error estimator on the average nodal fluxes from the SCP process. Basically, we will want to integrate the difference between the spatial distributions of the two flux estimates so that we can calculate the error norms of interest in each element. Then we will sum those scalar values over all elements so that we can establish the relative errors and how they compare to the allowed value specified by the user. Of course, we will evaluate the element norms by numerical integration. This will require a higher order quadrature rule than the one needed to evaluate the element square matrix because the interpolation function, \( P \) (which is usually \( H \)), is of higher polynomial degree than the \( B \) matrix (which contains the derivatives of \( H \)) used in forming the element square matrix.

Subroutine SCP_ERROR_ESTIMATES implements the major steps outlined below.

1. Preliminary Setup

   (a) Initialize all of the norms to zero.

   (b) If the mesh has a constant constitutive matrix, \( E \), then recover it and invert it for later use in calculating the energy norm.
2. Loop over all elements in the mesh:

(a) Recover the element type (shape, number of nodes, quadrature rule for $B$, etc.)

(b) Determine the quadrature rule to integrate the $P$ array (here the $H$ array), allocate storage for that array to be pre-computed at each quadrature point, and then fill those arrays.

(c) Extract the element's node numbers, coordinates, and $dof$.

(d) At each node on the element gather the continuous nodal flux components from the system SCP averages, $a_e \subset a$.

(e) Numerical integration loop over the element to form element norms and increment system norms:

i. Recover the $H$ array at the point and its local derivatives.

ii. Obtain the physical coordinates, Jacobian and its inverse.

iii. If the $L_2$ norm of the solution is desired, interpolate for the value at the point. Increment the $L_2$ norm integral. If the exact solution has been provided, then compute its norm also.

iv. Compute the physical gradients of the original finite element solution. Form the $B$ matrix at the point for the current application.

$$\epsilon = B \phi$$

If the constitutive array, $E$, is smoothly varying, then we could evaluate it at this point (and compute its inverse). Otherwise, we employ the $E$ matrix saved in the preliminary setup. Now we recover the standard element flux estimate by matrix multiplication

$$\tilde{\sigma} = E \epsilon.$$
We can increment the $L_2$ norm of this term if desired.

v. Now we are ready to recover the continuous flux values and approximate the stress error. We simply carry out the matrix product of the interpolation functions, $\mathbf{H}$, and the nodal fluxes, $\mathbf{a}$.

\[
\mathbf{\sigma'} = \mathbf{H} \mathbf{a}
\]

The difference between these components and those from the previous step are formed to define the stress error

\[
\mathbf{e}_\sigma = \mathbf{\sigma'} - \mathbf{\hat{\sigma}}.
\]

If desired the square of this term (its dot product with itself) is obtained for its increment to the $L_2$ stress norm.

vi. In this implementation we always use the flux error to compute the error energy norm, so at this stage we form the related triple matrix product, $\mathbf{e}_\sigma^T \mathbf{E}^{-1} \mathbf{e}_\sigma$, and increment the quadrature point contribution to the element norm

\[
\|\mathbf{e}_\sigma\|^2 = \sum_{q}^{\text{NQ}} (\sigma^* - \hat{\sigma}_q) \mathbf{E}_q^{-1} (\sigma^* - \hat{\sigma}_q).
\]

If the user has supplied an expression for the exact flux components, then they are evaluated at the physical coordinates, and the corresponding $L_2$ and error energy norms are updated for later comparisons and to find the effectivity index.

Having incremented all of the element norms, they are complete at the end of this quadrature loop for the current element. The active element norm values are then added to the current values of the corresponding system norms. At times we also want to use the element and system volume measures so that we can get some norm volumetric averages. Thus, the determinant of the Jacobian at the above points are
also used to obtain the element volumes so that they are available for these optional calculations.

Upon completing the loop over all elements we have the element norms, the element volume, the system norms, and the system volume. The allowed error energy is obtained from the product of the strain energy norm and the user input value of the allowed percentage error, \( \eta \) (keyword input value scp.allow.error.%). That number is used, in turn, to evaluate two allowed error densities in dividing by the square root of the number of elements and the square root of the volume to yield mean element and volumetric references, respectively. One of these reference values will be used to rank the relative error measures in each element. The system norm values are printed along with the two allowable reference values for the energy error and the system volume. For each element we will list the element error norm, its percentage of the strain energy norm, and a refinement parameter for that element. Here, the refinement parameter is based on the volumetric error density, so for element \( j \) the refinement parameter is

\[
\xi_j = \left( \frac{\|e_j\|/\sqrt{\Omega_j}}{\eta \|e\|/\sqrt{\Omega}} \right)
\]

or

\[
\xi_j = \frac{\|e_j\|}{\eta \|e\|} \left( \frac{\Omega}{\Omega_j} \right)^{\frac{1}{2}}.
\]  

(4.1)

Here it is informative to note that for a uniform mesh all the \( NE \) element volumes are constant with a value of

\[
\Omega_j = \Omega/NE
\]

and the refinement indicator becomes the same as originally employed by Zienkiewicz and Zhu:
\[ \xi_j = \frac{||e_j||}{\eta||e|| \sqrt{NE}}. \]  

By combining such an indicator with interpolation error analysis, one can predict the desired element size or polynomial degree. For each element, \( i \), we define the ratio \( \xi \) to indicate needed refinement when \( \xi_i > 1 \) and de-refinement when \( \xi_i < 1 \).

### 4.4 H-Adaptivity

If \( h_i \) is the current element size, then asymptotic convergence rate estimates suggest that the new element size should be smaller than \( h = h_i/\xi^{1/p} \) where \( p \) is the polynomial order of the interpolation for \( \phi \).

### 4.5 Second Derivative Recovery

There are times when one is also interested in the estimates of the second derivatives of the solution. Examples include the application of the Streamline Upwind Petrov Galerkin (SUPG) method for advection-diffusion problems and the inclusion of "stabilization" (or governing PDE residual) terms in the solution of the Navier-Stokes equations. If one is employing high-order interpolation elements, one could proceed with direct estimates of the second derivatives at the element level. Of course, we would expect a decrease in accuracy compared to the element gradient estimates. Assuming the use of parametric elements, we need to employ the Jacobian. Recall that the Jacobian defines the mapping from the parametric space the physical space. In two dimensions:

\[
\begin{bmatrix}
\frac{\partial}{\partial r} \\
\frac{\partial}{\partial s}
\end{bmatrix} = \begin{bmatrix}
\frac{\partial x}{\partial r} & \frac{\partial y}{\partial r} \\
\frac{\partial x}{\partial s} & \frac{\partial y}{\partial s}
\end{bmatrix} \begin{bmatrix}
\frac{\partial}{\partial x} \\
\frac{\partial}{\partial y}
\end{bmatrix}.
\]
Continuing this process to relate second parametric derivatives to second physical derivatives involves products and derivatives of the Jacobian array. For a two-dimensional mapping:

\[
\begin{bmatrix}
\frac{\partial^2 x}{\partial r^2} & \frac{\partial^2 x}{\partial r \partial s} & \frac{\partial^2 x}{\partial s^2} \\
\frac{\partial^2 y}{\partial r^2} & \frac{\partial^2 y}{\partial r \partial s} & \frac{\partial^2 y}{\partial s^2}
\end{bmatrix}
\begin{bmatrix}
\frac{\partial}{\partial x} \\
\frac{\partial}{\partial y}
\end{bmatrix}
+ \begin{bmatrix}
\left(\frac{\partial x}{\partial r}\right)^2 & \left(\frac{\partial x}{\partial s}\right)^2 & 2\frac{\partial x}{\partial r}\frac{\partial x}{\partial s} \\
\left(\frac{\partial y}{\partial r}\right)^2 & \left(\frac{\partial y}{\partial s}\right)^2 & 2\frac{\partial y}{\partial r}\frac{\partial y}{\partial s}
\end{bmatrix}
\begin{bmatrix}
\frac{\partial^2 x}{\partial r^2} & \frac{\partial^2 x}{\partial r \partial s} & \frac{\partial^2 x}{\partial s^2} \\
\frac{\partial^2 y}{\partial r^2} & \frac{\partial^2 y}{\partial r \partial s} & \frac{\partial^2 y}{\partial s^2}
\end{bmatrix}
\begin{bmatrix}
\frac{\partial}{\partial x} \\
\frac{\partial}{\partial y}
\end{bmatrix}
\]

(4.3)

For a constant Jacobian the first rectangular matrix on the right is zero. Otherwise, the second derivatives are clearly more sensitive to a variable Jacobian. In each case, we must invert the square matrices to obtain the first and second physical derivatives. If the Jacobian is not constant, then the effect of a distorted element would be amplified by the product terms in the square matrix of Eq. (4.3), as well as by the second derivatives in the rectangular array that multiplies the physical gradient term. Using this analytic form for the second derivative of an approximate finite element solution is certainly questionable. Clearly, for a linear interpolation element with a constant Jacobian, the result would be identically zero. Thus, we will actually only use this form as a tool to estimate regions of high second-derivative error, as compared to the values obtained from a patch recovery technique.

In the present software the second derivative calculations and associated output are activated by a global logical constant SCP_2ND_DERIV which is set to true by including an input keyword control of scp_2nd_deriv in the data file. If the norm of the estimated second derivative error is required, then the square matrix of Jacobian product terms is computed in subroutine JACOBIAN_PRODUCTS and is given the name P_AJ. Its associated inverse matrix is called P_AJ_INV. The second parametric (local) derivatives of the interpolation functions, H, are denoted as D2LH and the corresponding second physical (global) derivatives, from Eq. (4.3), are denoted
by D2GH Each has N_2.DER rows and a column for each interpolation function. For one-, two- and three-dimensional problems, N_2.DER has a value of 1, 3, or 6, respectively.

Since the analytic estimate will usually be compared to a smoothed patch estimate, the second derivative terms are also computed in VAL.SCP.FIT.AT.PATCH.NODES if SCP.2ND.DERIV is true. After the element fluxes have been processed to give continuous nodal values on a patch (as described earlier), the physical gradients of the patch interpolations, SCP.DGH are invoked to evaluate the gradients of the fluxes (i.e., the second derivatives) at all of the mesh nodes in the patch. They are then also scattered to the system array, SCP.AVERAGES for later averaging over all patch contributions. Sometimes one may want to bias estimates on the boundary of the domain before scattering its contribution to the system averages.

After the flux components and their gradients (second derivatives) have been averaged, they are listed at the nodes and passed to the routine SCP.ERROR.ESTIMATES. There the second derivative values are only used to calculate various measures or norms for them and their estimated error. While the corresponding cross derivatives like $\partial^2/\partial x \partial y$ and $\partial^2/\partial y \partial x$ should be equal, they generally will differ due to various numerical approximations. All cross derivatives are computed, but only average values are used in estimating errors in the second derivatives. The larger full set of second derivatives at the nodes of an element are gathered and placed in an array called DERIV2.LT. If cross derivative estimates exist, they are averaged and placed in a smaller array called DERIV2.AVE that has the N_2.DER second derivatives at each element node. They will be interpolated to give the average second derivatives at the quadrature points used to evaluate the norms. By analogy to the first derivative norms, the N_2.DER interpolated second derivatives at a point are called DERIV2.SCP. The values computed directly from Eq. (4.3) are called
DERIV2.HAT. Their values and differences are used to define norms and error estimates at the element level (ELEM.H2.NORM and ELEM.H2.ERROR) and at the system level (GLOBAL.H2.NORM and GLOBAL.H2.ERROR). If an exact solution is available for comparison, called FLUX.GRAD it is used to compute corresponding exact values (EXACT.H2.NORM and EXACT.H2.ERROR). The various norm and error measures are listed, as are the SCP averaged and exact second derivatives at the system nodes. For plotting or other use, the last two items are saved to external files called pt.ave.grad.flux.tmp and pt.ex_grad.flux.tmp respectively.
Chapter 5

Two-Dimensional Whiteman Square

The two-dimensional Poisson equation is solved here for the Lakhany and Whiteman[23] square test problem. It is evaluated for three different sets of boundary conditions. By taking advantage of the various symmetry and anti-symmetry aspects of a geometry, a complex geometry can be reduced to a more economical geometrical model for analysis. The problem was designed by Lakhany and Whiteman to test the recovery of second and third derivatives.

The exact solution is:

\[ u(x, y) = (1 - x^2) \cdot \sin(\pi \cdot y) \]

which gives \( u=0 \) on the square boundary from \((-1, -1)\) to \((+1, +1)\). Substituting into the Poisson equation:

\[ (u, xx + u, yy) = -(2 + \pi^2 \cdot (1 - x^2)) \cdot \sin(\pi \cdot y) \]

so the term on the right defines the source per unit area, \( Q \), that drives the problem.
Figure 5.1: Whiteman Square Boundary Conditions
Figure 5.2: Whiteman Square Analytic Solution
5.1 Full Whiteman Square for T3 Elements

5.1.1 Geometry

The analysis begins with the complete square with a width and height equivalent to 2 units, where the width ranges from -1 to +1 in the x direction and from -1 to +1 in the y direction, see Figure 5.1. The full model has four essential boundary condition regions and no natural boundary conditions. The first example uses the triangular T3 elements, therefore the maximum number of nodes per element is 3.

5.1.2 Analysis

![Figure 5.3: Initial Mesh h=0.5](image)

The original number of nodes in the mesh is 25, the number of elements in the system is 32, and the length per element is \( h = 0.50 \) for the initial mesh. (Figure 5.3) The example is run to obtain the maximum number of element neighbors, exact second derivative error, and exact flux error of three different superconvergent patches. The three superconvergent patch options used to compare the error are: the element
based patch (all neighbors), the element based patch (facing neighbors), and the nodal based patch (all element neighbors).

For a uniform \( h \) refinement the mesh at each stage is refined by dividing the mesh in half in the x and y direction. After the first refinement the number of nodes in the mesh will be 81, the number of elements in the system will be 128, and the length of each element \( h = 0.25 \). Three more refinements of the mesh yields the (node, element) combinations of (289, 512) at \( h = 0.125 \), (1089, 2048) at \( h = 0.0625 \), and (4225, 8192) at \( h = 0.03125 \), see Figure 5.4.

Figure 5.4: Uniform Mesh Refinement of Full Model, T3 Element
The previous steps are repeated to acquire the error norms for the gradients and the second derivative for each patch type option at each value of h. These values are plotted on a log-log graph (Figure 5.5) to compare the effectiveness of each patch as the mesh is refined.

Figure 5.5: Log-Log Error Norm Plots for T3 Elements
5.2 Full Whiteman Square for Q4 Elements

5.2.1 Geometry

The analysis uses the same geometry as the T3 element, however the quadrilateral Q4 element has a maximum number of four nodes per element.

5.2.2 Analysis

The number of nodes in the mesh is 25, the number of elements in the system is 16, and the length per element is \( h = 0.50 \) for the initial mesh. The example is run to obtain exact second derivative error and exact flux error of three different superconvergent patch type options. The same three superconvergent patches stated previously are used to compare the error.

![FE Mesh Geometry: 16 Elements, 25 Nodes](image)

**Figure 5.6: Initial Mesh \( h=0.5 \)**

The mesh is refined by dividing the mesh in half in the x and y direction. The refinement will bring the number of nodes in the mesh to 81 and the number of elements in the system to 64, while the length of each element \( h = 0.25 \). Three more
refinements of the mesh yields the (node, element) combinations of (289,256) at $h = 0.125$, (1089, 1024) at $h = 0.0625$, and (4225, 4096) at $h = 0.03125$.

![Diagram of FE Mesh Geometry: 256 Elements, 260 Nodes]

Figure 5.7: Mesh Full Model, Q4 Element

The previous steps are repeated to acquire the error for each patch at each value of $h$. These values are plotted on a log-log graph, see (Figure 5.8) to compare the effectiveness of each patch, as the mesh is refined.
Figure 5.8: Log-Log Error Norm Plots for Q4 Elements
5.3 Full Whiteman Square for T6 Elements

5.3.1 Geometry

The analysis uses the same geometry as the T3 element, however the T6 elements have the maximum number of six nodes per element.

5.3.2 Analysis

The analysis for the T6 element is similar to the two previous examples, and the length between each node is 0.5. Similarly, the example is run to obtain the exact second derivative error and exact flux error of three different superconvergent patches. The element based patch (all neighbors), the element based patch (facing neighbors), and the nodal based patch are plotted on a log-log graph to compare the effect of each patch as the mesh is refined.

Figure 5.9: Initial Mesh h=0.5
Figure 5.10: Mesh Full Model, T6 Element

Figure 5.11: Log-Log Error Norm Plots for T6 Elements
5.4 One-Fourth Whiteman Square for T3 Elements

5.4.1 Geometry

The analysis of the quarter model of the Lakhany and Whiteman test has a width and height equivalent to 1 unit, where the width ranges from 0 to +1 in the x direction and from 0 to +1 in the y direction, see Figure 5.1. The model has two essential and two natural boundary conditions. The first example uses the T3 elements, therefore the maximum number of nodes per element is three.

5.4.2 Analysis

Similarly to the full Whiteman square model, the number of nodes in the mesh is 25 and the number of elements in the system is 32. The significance of the quarter model is that the length per element is reduced to \( h = 0.25 \) for the initial mesh. This phenomenon hints that the quarter model should converge to the true solution at a lower cost.

Like the previous model, the example is run to obtain the exact second derivative error and exact flux error of three different superconvergent patches.

The mesh is refined by dividing the mesh in half in the x and y direction, see Figure 5.12. The refinement will bring the number of nodes in the mesh to 81 and the number of elements in the system to 128, while the length of each element \( h = 0.125 \). It is obvious that the number of nodes and elements will remain the same while the length of each element is reduced. Three more refinements of the mesh will now yield the (node, element) combinations of \((289, 512)\) at \( h = 0.0625 \), \((1089, 2048)\) at \( h = 0.03125 \), and \((4225, 8192)\) at \( h = 0.015625 \).

The previous steps are repeated to acquire the error for each patch at each value of \( h \), see Figure 5.13. These values are plotted on a log-log graph to compare the effectiveness of each patch, as the mesh is refined.
Figure 5.12: Mesh Quarter Model, T3 Element

Figure 5.13: Log-Log Error Norm Plots for T3 Elements of 1/4 Model
5.5 One-Fourth Whiteman Square for Q4 Elements

5.5.1 Geometry

The analysis uses the same geometry as the T3 element for the quarter model, however the Q4 element is employed.

5.5.2 Analysis

The number of nodes in the mesh is 25, the number of elements in the system is 16, and the length per element is $h = 0.25$ for the initial mesh. Similarly to the previous element, the values of number of nodes and elements corresponds with the full model, while the length of each element $h$ corresponds with the quarter model length of element T3.

![Mesh Quarter Model, Q4 Element](image)

(First Refinement)  
(Second Refinement)

**Figure 5.14: Mesh Quarter Model, Q4 Element**

The refinements of the mesh yields the (node, element) combinations of $(31, 64)$ at $h = 0.125$, $(289, 256)$ at $h = 0.0625$, $(1089, 1024)$ at $h = 0.03125$, and $(4225, 4096)$ at $h = 0.015625$ see Figure 5.14.
The previous steps are repeated to acquire the error for each patch at each value of h. These values are plotted on a log-log graph (Figure 5.15) to compare the effectiveness of each patch, as the mesh is refined.

Figure 5.15: Log-Log Error Norm Plots for Q4 Elements of 1/4 Model
5.6 One-Fourth Whiteman Square for T6 Elements

5.6.1 Geometry

The analysis uses the same geometry as the T3 element, however the T6 elements have the maximum number of nodes per element is six.

5.6.2 Analysis

The analysis for the T6 element is similar to the two previous examples with an $h$ of 0.25.

![Mesh Quarter Model, T6 Element](image)

Figure 5.16: Mesh Quarter Model, T6 Element

The refinements of the mesh (Figure 5.16) yields the (node, element) combinations of $(25, 8)$ at $h = 0.25$, $(81, 32)$ at $h = 0.125$, $(289, 125)$ at $h = 0.0625$, $(1089, 512)$ at $h = 0.03125$, and $(4225, 2048)$ at $h = 0.015625$. The number of nodes corresponds to node numbers of the quarter model of the T3 and Q4 elements, while the number of elements and the length of the element correspond to elements and $h$ of the full model with linear elements T3 and Q4.
Similarly, the example is run to obtain the maximum number of element neighbors, exact second derivative error, and exact flux error of three different superconvergent patches. The element based patch, all neighbors; the element based patch, facing neighbors; and the nodal based patch, all element neighbors are plotted on a log-log graph (Figure 5.17) to compare the effect of each patch as the mesh is refined.

Figure 5.17: Log-Log Error Norm Plots for T6 Elements of 1/4 Model
5.7 One-Eighth Whiteman Square for T3 Elements

5.7.1 Geometry

The analysis of the eighth model of the Lakhany and Whiteman test has a width equivalent to 1 unit and a height equivalent to 0.5 unit, where the value ranges from 0 to +1 in the x direction and from 0.50 to +1 in the y direction, see Figure 5.1. The model has three essential and one natural boundary conditions. The first example uses the T3 elements, therefore the maximum number of nodes per element is 3.

5.7.2 Analysis

In the eighth Whiteman Square model, the number of nodes in the mesh is 15 and the number of elements in the system is 16. Similar to the quarter model the length per element is reduced to \( h = 0.25 \) for the initial mesh. This allows the comparison of the trade off between using a slightly more expensive element versus the cheaper geometry.

Like the previous model, the example is run to obtain the exact second derivative error and exact flux error of three different superconvergent patches.

The refinements of the mesh (Figure 5.18) yields the (node, element) combinations of \((45, 64)\) at \( h = 0.125 \), \((153, 256)\) at \( h = 0.0625 \), \((561, 1024)\) at \( h = 0.03125 \), and \((2145, 4096)\) at \( h = 0.015625 \).

The previous steps are repeated to acquire the error for each patch at each value of \( h \). The element based patch, all neighbors, the element based patch, facing neighbors; and the nodal based patch, all element neighbors are plotted on a log-log graph (Figure 5.19) to compare the effectiveness of each patch as the mesh is refined for the quarter model.
Figure 5.18: Mesh Eighth Model, T3 Element

Figure 5.19: Log-Log Error Norm Plots for T3 Elements of 1/8 Model
5.8 One-Eighth Whiteman Square for Q4 Elements

5.8.1 Geometry

The analysis uses the same geometry as the T3 element for the eighth model, however the Q4 element is applied.

5.8.2 Analysis

The number of nodes in the mesh is 15, the number of elements in the system is 8, and the length per element is \( h = 0.25 \) for the initial mesh. The values of the number of nodes and the length of each element \( h \) correspond with the eighth model of element T3.

![Figure 5.20: Mesh Eighth Model, Q4 Element](image)

The refinements of the mesh (Figure 5.20) yields the (node, element) combinations (45, 32) at \( h = 0.125 \), (153, 128) at \( h = 0.0625 \), (561, 512) at \( h = 0.03125 \), and (2145, 2048) at \( h = 0.015625 \).
The previous steps are repeated to acquire the error norms for each patch type option at each value of $h$. These values are plotted on a log-log graph (Figure 5.21) to compare the effectiveness of each patch, as the mesh is refined.

![Graph showing log-log error norm plots for Q4 elements of 1/8 model](image)

**Figure 5.21: Log-Log Error Norm Plots for Q4 Elements of 1/8 Model**
5.9 One-Eighth Whiteman Square for T6 Elements

5.9.1 Geometry

The analysis uses the same geometry as the T3 element and the T6 element is used.

5.9.2 Analysis

The analysis for the T6 element is similar to the two previous examples, however the length of each element, \( h \), is 0.25 for the initial mesh.

![First Refinement](image1.png)  ![Second Refinement](image2.png)

(First Refinement)  (Second Refinement)

Figure 5.22: Mesh Eighth Model, T6 Element

The refinements of the mesh (Figure 5.22) yields the (node, element) combinations of \((15, 4)\) at \( h = 0.25 \), \((45, 16)\) at \( h = 0.125 \), \((153, 64)\) at \( h = 0.0625 \), \((561, 256)\) at \( h = 0.03125 \), and \((2145, 1024)\) at \( h = 0.015625 \).

Similarly, the example is run to obtain the exact second derivative error and exact flux error of three different superconvergent patches. The superconvergent patches are plotted on a log-log graph (Figure 5.23) to compare the effect of each patch as the mesh is refined.
Figure 5.23: Log-Log Error Norm Plots for T6 Elements of 1/8 Model
5.10 Results

The results of the examples in the chapter can be examined using two methods. The first method is to compare the different element shapes, while the second method compares the different model boundary conditions.

5.10.1 Element Comparison

The element comparison plots (Figures 5.24, 5.25, 5.28) show that the T6 element would be the best element to use. Since the T6 element has a higher polynomial order, the plots of the T6 element have a steeper slope. Despite the fact that the T6 element is more expensive, the trade off for the better error estimate is preferred.

Figure 5.24: Comparison Plot of Elements Log-Log plot Full Model
Figure 5.25: Comparison Plot of Elements Log-Log plot Quarter Model
Figure 5.26: Comparison Plot of Elements Log-Log plot Eighth Model
5.10.2 Boundary Condition Comparison

The boundary condition comparison plot reveals a great deal of important information. The error is proportional to some constant times the size, \( h \), raised to the power of the slope of the log-log plot. The constant is a function of the shape and boundary conditions. The boundary condition plots (Figures 5.27, 5.28, 5.29) compare the error versus the size, \( h \).

![Comparison Plot of Boundary Conditions Log-Log plot with T3 Elements](image)

Figure 5.27: Comparison Plot of Boundary Conditions Log-Log plot with T3 Elements
Figure 5.28: Comparison Plot of Boundary Conditions Log-Log plot with Q4 Elements
Figure 5.29: Comparison Plot of Boundary Conditions Log-Log plot with T6 Elements
Chapter 6

H-Adaptive Refinement of the Lakhany Whiteman Square

The uniform mesh used in the previous chapter is not very cost effective, therefore adaptive mesh refinement is used to generate the mesh. H-refinement, p-refinement, r-refinement, and hp-refinement are four methods of adaptive refinement. The h-refinement changes the size of the elements h, the p-refinement effects the polynomial order, the r-refinement relocates the nodes, and the hp refinement is a combination of the first two methods. The Lakhany Whiteman [23] square is examined using the refinement parameter discussed in chapter 4 and the h-refinement method.

![Figure 6.1: Error plot](image)

The energy norm error estimation averaged at the nodes is shown in Figure 6.1. Using the h-refinement method, it is expected that a finer mesh will appear in areas with the largest amount of error.
6.1 Full Model

Properties, such as number of holes, number of regions, number of materials in region, number of segments, and segment types, are given in the data file for the adaptive mesh generation program. For the Lakhany Whiteman square the number of segments is four, and each segment type is a line. There are zero holes, and one region with only one material in each region. The four lines for the square go through the points (-1,-1), (+1,-1), (+1,+1), and (-1,+1). The program will refine the mesh in areas that contain the most error, while the areas with less error will be de-refined. The output file returns the new size, h, for the elements.

![Initial H-Adaptive Mesh average h = 0.5](image)

**Figure 6.2: Initial H-Adaptive Mesh average h = 0.5**

The initial mesh created is shown in Figure (6.2) with an average size, h, of 0.5. The mesh is refined four additional times, and the second and third refinements are shown in Figure (6.3). The h-adaptive process is witnessed by examining the mesh in Figures 6.1 and 6.4, since the element size h is smaller near the area with the larger error.

The log-log plot of the error norm of the h-adaptive mesh are shown in Figure 6.6. The figures show the error norm versus the number of equations. Figure 6.7 is used to evaluate the effectiveness of the uniform refinement versus the h-adaptive refinement.
(Second Refinement)  

(Third Refinement)  

Figure 6.3: H-Adaptive Refinement - Full Model

Figure 6.4: Final H-Adaptive Refined Mesh
Figure 6.5: Uniform Mesh Refinement

Figure 6.6: H-Adaptive Mesh Refinement
Figure 6.7: Comparison of Uniform Mesh and H-Adaptive Mesh for Full Model
6.2 One-Eighth Model

The one-eighth model is evaluated to see the effects of a model with different boundary conditions and the h-adaptive method. The model contains three sides with essential boundary conditions and one with natural boundary conditions. The initial model is shown in Figure (6.8).

![FE Mesh Geometry: 28 Elements, 20 Nodes](image)

**Figure 6.8: Initial Mesh of One-Eighth Model**

Similarly to the full model, the h-adaptive method is applied to acquire the meshes shown in Figures (6.9, 6.10).

The log-log plot of the error norm versus the number of equations is shown in Figure (6.12). The same method stated above of comparing the effectiveness of the model is used with the one-eighth model, Figure (6.13). As expected, the one-eighth model is a better model, but the more expensive patch is better. The comparison plots show that despite using a more effective refinement method, the more expensive element patch is still the better super-convergent patch option.
Figure 6.9: H-Adaptive Mesh Refinement of One-Eighth Model

Figure 6.10: Final H-Refinement Mesh of One-Eighth Model
Figure 6.11: Uniform Mesh Refinement of One-Eighth Model

Figure 6.12: H-Adaptive Mesh Refinement of One-Eighth Model
Figure 6.13: Comparison of Uniform Refinement to H-Adaptive Refinement of One-Eighth Model
Chapter 7

Conclusion and Future Work

7.1 Conclusion

At the conclusion of the research, a lot of important information is gathered. As expected when comparing different elements, the steeper slopes of the quadratic triangle show that higher polynomial orders are of course more accurate in predicting the second derivatives. Despite the fact that linear elements are more cost effective, the trade-off of the better element is desired.

The effects of the different boundary conditions were anticipated also. By inspection of the plots, the one-eighth model would be the best choice for the model.

For those problems that must be formulated with linear elements, and thus no second derivatives, the evaluation of the patch choices proves important when second derivatives must be recovered.

Since the h-refinement method is more cost effective than the uniform refinement method, the super-convergent patch options were evaluated. The more expensive element patch is found to be the better choice, despite the expectation of the h-refinement method to improve the super-convergent patch options. The second derivative accuracy is less than what was hoped for. It may be practical to improve their accuracy by increasing the degree of the patch polynomial, or by adding an additional layer of elements to the perimeter of the patch.
7.2 Future Work

Some aspects of the research were expected; however there were some new observable occurrences. Some suggestions for future work are below:

- The phenomenon of the nodal patch and facing patch exchanging position is an incident that needs future work.

- The patch options need to be evaluated for different problems, such as problems with curved boundaries.

- One-dimensional and three-dimensional problems need to be analyzed to see the behavior of the super-convergent patch options, also.
Bibliography


[33] Zhang, Z. and Zhu, J.Z., Superconvergence Of The Derivative Patch Recovery Technique And A Posteriori Error Estimation: In IMA Volumes in Mathematics


